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

CS4618 Artificial Intelligence I

Today: Evolutionary Algorithms
Introduction to Probability

Thomas Jansen

October 19th

Plans for Today

Randomised Search Heuristics

Remember we have

- different variants of local search
  - first improvement
  - best improvement
  - random
- Metropolis algorithm
- simulated annealing
- general framework for evolutionary algorithms
- modules for evolutionary algorithms
  - selection: uniform, fitness-proportional, tournament, plus, comma
  - variation: crossover (uniform, k-point), mutation (standard bit mutation, b-bit mutation)

Randomised Search Heuristics

Remember we have

- different variants of local search
  - first improvement
  - best improvement
  - random
- Metropolis algorithm
- simulated annealing
- general framework for evolutionary algorithms
- modules for evolutionary algorithms
  - selection: uniform, fitness-proportional, tournament, plus, comma
  - variation: crossover (uniform, k-point), mutation (standard bit mutation, b-bit mutation)

New some concrete evolutionary algorithms

(1+1) EA

(1+1) EA

Parameter mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$

(1+1) EA

Parameter mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$

(1+1) EA

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

(1+1) EA

Parameter mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$

(1+1) EA

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. $\quad y :=$ standard bit mutation(x_t)

(1+1) EA

Parameter mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$

(1+1) EA

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. $\quad y :=$ standard bit mutation(x_t)
4. $\quad t := t + 1$; If $f(y) \geq f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.

(1+1) EA

Parameter mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$

(1+1) EA

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

(1+1) EA

Parameter mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$

(1+1) EA

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

Facts

(1+1) EA

Parameter mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$

(1+1) EA

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

Facts

- simplest evolutionary algorithm

(1+1) EA

Parameter mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$

(1+1) EA

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

Facts

- simplest evolutionary algorithm
- useful for understanding the role of mutation

(1+1) EA

Parameter mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$

(1+1) EA

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

Facts

- simplest evolutionary algorithm
- useful for understanding the role of mutation
- similar to but different from local search and simulated annealing

$(\mu+\lambda)$ EA

$(\mu+\lambda)$ EA

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
population size $\mu \in \mathbb{N}$, offspring population size $\lambda \in \mathbb{N}$

$(\mu+\lambda)$ EA

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
population size $\mu \in \mathbb{N}$, offspring population size $\lambda \in \mathbb{N}$

$(\mu+\lambda)$ EA

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

$(\mu+\lambda)$ EA

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
population size $\mu \in \mathbb{N}$, offspring population size $\lambda \in \mathbb{N}$

$(\mu+\lambda)$ EA

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

$(\mu+\lambda)$ EA

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
population size $\mu \in \mathbb{N}$, offspring population size $\lambda \in \mathbb{N}$

$(\mu+\lambda)$ EA

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. Select $z \in \{x_1, x_2, \ldots, x_\mu\}$ uniformly at random.

$(\mu+\lambda)$ EA

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
population size $\mu \in \mathbb{N}$, offspring population size $\lambda \in \mathbb{N}$

$(\mu+\lambda)$ EA

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. Select $z \in \{x_1, x_2, \ldots, x_\mu\}$ uniformly at random.
5. $y_i :=$ standard bit mutation(z)

$(\mu+\lambda)$ EA

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
population size $\mu \in \mathbb{N}$, offspring population size $\lambda \in \mathbb{N}$

$(\mu+\lambda)$ EA

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. Select $z \in \{x_1, x_2, \ldots, x_\mu\}$ uniformly at random.
5. $y_i :=$ standard bit mutation(z)
6. Select new $x_1, x_2, \ldots, x_\mu$ out of best $x_1, x_2, \ldots, x_\mu$ and $y_1, y_2, \ldots, y_\lambda$.

$(\mu+\lambda)$ EA

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
population size $\mu \in \mathbb{N}$, offspring population size $\lambda \in \mathbb{N}$

$(\mu+\lambda)$ EA

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. Select $z \in \{x_1, x_2, \ldots, x_\mu\}$ uniformly at random.
5. $y_i :=$ standard bit mutation(z)
6. Select new $x_1, x_2, \ldots, x_\mu$ out of best $x_1, x_2, \ldots, x_\mu$ and $y_1, y_2, \ldots, y_\lambda$.
7. Until 'decide to stop'
8. Output x_i with $f(x_i) = \max\{f(x_j) \mid 1 \leq j \leq \mu\}$.

(μ, λ) EA

(μ, λ) EA

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
population size $\mu \in \mathbb{N}$, offspring population size $\mu \in \mathbb{N}$
$\lambda \geq \mu$

(μ, λ) EA

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
population size $\mu \in \mathbb{N}$, offspring population size $\mu \in \mathbb{N}$
$\lambda \geq \mu$

(μ, λ) EA

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

(μ, λ) EA

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
population size $\mu \in \mathbb{N}$, offspring population size $\mu \in \mathbb{N}$
$\lambda \geq \mu$

(μ, λ) EA

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

(μ, λ) EA

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
population size $\mu \in \mathbb{N}$, offspring population size $\mu \in \mathbb{N}$
$\lambda \geq \mu$

(μ, λ) EA

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. Select $z \in \{x_1, x_2, \ldots, x_\mu\}$ uniformly at random.

(μ, λ) EA

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
population size $\mu \in \mathbb{N}$, offspring population size $\mu \in \mathbb{N}$
$\lambda \geq \mu$

(μ, λ) EA

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. Select $z \in \{x_1, x_2, \ldots, x_\mu\}$ uniformly at random.
5. $y_i :=$ standard bit mutation(z)

(μ, λ) EA

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
population size $\mu \in \mathbb{N}$, offspring population size $\mu \in \mathbb{N}$
$\lambda \geq \mu$

(μ, λ) EA

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. Select $z \in \{x_1, x_2, \ldots, x_\mu\}$ uniformly at random.
5. $y_i :=$ standard bit mutation(z)
6. Select new $x_1, x_2, \ldots, x_\mu$ out of best $y_1, y_2, \ldots, y_\lambda$.

(μ, λ) EA

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
population size $\mu \in \mathbb{N}$, offspring population size $\mu \in \mathbb{N}$
$\lambda \geq \mu$

(μ, λ) EA

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. Select $z \in \{x_1, x_2, \ldots, x_\mu\}$ uniformly at random.
5. $y_i :=$ standard bit mutation(z)
6. Select new $x_1, x_2, \ldots, x_\mu$ out of best $y_1, y_2, \ldots, y_\lambda$.
7 Until 'decide to stop'
8. Output x_i with $f(x_i) = \max\{f(x_j) \mid 1 \leq j \leq \mu\}$.

Steady State Genetic Algorithm

Steady State Genetic Algorithm

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
crossover probability $p_c \in [0, 1]$, population size $\mu \in \mathbb{N}$

Steady State Genetic Algorithm

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
crossover probability $p_c \in [0, 1]$, population size $\mu \in \mathbb{N}$

Steady State GA

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

Steady State Genetic Algorithm

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
crossover probability $p_c \in [0, 1]$, population size $\mu \in \mathbb{N}$

Steady State GA

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

Steady State Genetic Algorithm

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
crossover probability $p_c \in [0, 1]$, population size $\mu \in \mathbb{N}$

Steady State GA

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

Steady State Genetic Algorithm

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
crossover probability $p_c \in [0, 1]$, population size $\mu \in \mathbb{N}$

Steady State GA

1. Choose $x_1, x_2, \ldots, x_\mu \in S$ uniformly at random.
2. Repeat
3. With probability p_c select $z_1, z_2 \in \{x_1, x_2, \ldots, x_\mu\}$ u. a. r.
   $z :=$ crossover(z_1, z_2)
   Else select $z \in \{x_1, x_2, \ldots, x_\mu\}$ uniformly at random.
4. $y :=$ standard bit mutation(z)

Steady State Genetic Algorithm

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
crossover probability $p_c \in [0, 1]$, population size $\mu \in \mathbb{N}$

Steady State GA

1. Choose $x_1, x_2, \ldots, x_\mu \in S$ uniformly at random.
2. Repeat
3. With probability p_c select $z_1, z_2 \in \{x_1, x_2, \ldots, x_\mu\}$ u. a. r.
   $z :=$ crossover(z_1, z_2)
   Else select $z \in \{x_1, x_2, \ldots, x_\mu\}$ uniformly at random.
4. $y :=$ standard bit mutation(z)
5. Select new $x_1, x_2, \ldots, x_\mu$ out of best $x_1, x_2, \ldots, x_\mu, y$.

Steady State Genetic Algorithm

Parameters mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$
crossover probability $p_c \in [0, 1]$, population size $\mu \in \mathbb{N}$

Steady State GA

1. Choose $x_1, x_2, \ldots, x_\mu \in S$ uniformly at random.
2. Repeat
3. With probability p_c select $z_1, z_2 \in \{x_1, x_2, \ldots, x_\mu\}$ u. a. r.
   $z :=$ crossover(z_1, z_2)
   Else select $z \in \{x_1, x_2, \ldots, x_\mu\}$ uniformly at random.
4. $y :=$ standard bit mutation(z)
5. Select new $x_1, x_2, \ldots, x_\mu$ out of best $x_1, x_2, \ldots, x_\mu, y$.
7 Until 'decide to stop'
8. Output x_i with $f(x_i) = \max\{f(x_j) \mid 1 \leq j \leq \mu\}$.

Remarks about Evolutionary Algorithms

Remarks about Evolutionary Algorithms

- only most basic variants considered, much more involved variants known and used

Remarks about Evolutionary Algorithms

- only most basic variants considered, much more involved variants known and used
- extreme variants (1+1) EA, (μ+1) EA, (1+λ) EA, (1, λ) EA are of special interest for understanding evolutionary algorithms

Remarks about Evolutionary Algorithms

- only most basic variants considered, much more involved variants known and used
- extreme variants (1+1) EA, (μ+1) EA, (1+λ) EA, (1, λ) EA are of special interest for understanding evolutionary algorithms
- steady state GA is (μ+1) EA with crossover (both forms of crossover common)

Remarks about Evolutionary Algorithms

- only most basic variants considered, much more involved variants known and used
- extreme variants (1+1) EA, (μ+1) EA, (1+λ) EA, (1, λ) EA are of special interest for understanding evolutionary algorithms
- steady state GA is (μ+1) EA with crossover (both forms of crossover common)
- static parameter settings may be replaced by
  - dynamic settings (time-driven, like in simulated annealing)

Remarks about Evolutionary Algorithms

- only most basic variants considered, much more involved variants known and used
- extreme variants (1+1) EA, (μ+1) EA, (1+λ) EA, (1, λ) EA are of special interest for understanding evolutionary algorithms
- steady state GA is (μ+1) EA with crossover (both forms of crossover common)
- static parameter settings may be replaced by
  - dynamic settings (time-driven, like in simulated annealing)
  - adaptive settings (depending on the run)

Remarks about Evolutionary Algorithms

- only most basic variants considered, much more involved variants known and used
- extreme variants (1+1) EA, (μ+1) EA, (1+λ) EA, (1, λ) EA are of special interest for understanding evolutionary algorithms
- steady state GA is (μ+1) EA with crossover (both forms of crossover common)
- static parameter settings may be replaced by
  - dynamic settings (time-driven, like in simulated annealing)
  - adaptive settings (depending on the run)
  - self-adaptive settings (evolving the parameter settings during the run)

Why Randomised Search Heuristics?

Why Randomised Search Heuristics?

Observations

- optimisation framework $(S,\ f : S \to \mathbb{R})$ extremely general

Why Randomised Search Heuristics?

Observations

- optimisation framework $(S,\ f\colon S \to \mathbb{R})$ extremely general
- randomised search heuristics easy to implement

Why Randomised Search Heuristics?

Observations

- optimisation framework $(S,\ f\colon S \to \mathbb{R})$ extremely general
- randomised search heuristics easy to implement
- randomised search heuristics easy to apply

Why Randomised Search Heuristics?

Observations

- optimisation framework $(S,\ f\colon S \to \mathbb{R})$ extremely general
- randomised search heuristics easy to implement
- randomised search heuristics easy to apply
- $\rightsquigarrow$ attractive for difficult problem
  where no good problem-specific algorithm is available

Why Randomised Search Heuristics?

Observations

- optimisation framework $(S,\ f\colon S \to \mathbb{R})$ extremely general
- randomised search heuristics easy to implement
- randomised search heuristics easy to apply
- $\rightsquigarrow$ attractive for difficult problem
  where no good problem-specific algorithm is available

Note of Caution

Don't apply randomised search heuristics
if good problem-specific algorithm is known!

Things to Come

Things to Come

- performance assessment
  Which randomised search heuristic is best?

Things to Come

- performance assessment
  Which randomised search heuristic is best?
- general limitation
  What cannot be achieved by randomised search heuristics?

Things to Come

- performance assessment
  Which randomised search heuristic is best?
- general limitation
  What cannot be achieved by randomised search heuristics?
- understanding mutations
  How do local and global mutations differ?

Things to Come

- performance assessment
  Which randomised search heuristic is best?
- general limitation
  What cannot be achieved by randomised search heuristics?
- understanding mutations
  How do local and global mutations differ?
- understanding selection
  How do simulated annealing and plus-selection differ?

Things to Come

- performance assessment
  Which randomised search heuristic is best?
- general limitation
  What cannot be achieved by randomised search heuristics?
- understanding mutations
  How do local and global mutations differ?
- understanding selection
  How do simulated annealing and plus-selection differ?
- understanding parametrisation
  How do the parameter settings influence randomised search heuristics?

Probability Measure

Probability Measure

Definition

A probability space consits of

- a countable sample space Ω and
- a probability measure $\text{Prob}\colon 2^{\Omega} \to [0;1]$

such that for the probability measure Prob the following holds.

1. $\forall A \subseteq \Omega\colon \text{Prob}(A) \geq 0$
2. $\forall A_1, A_2, \cdots \subseteq \Omega\colon\ (\forall i \neq j\colon A_i \cap A_j = \emptyset)$
   $\Rightarrow \left(\text{Prob}\left(\bigcup_i A_i\right) = \sum_i \text{Prob}(A_i)\right)$
3. $\text{Prob}(\Omega) = 1$

Probability Measure

Definition

A probability space consits of

- a countable sample space Ω and
- a probability measure $\text{Prob}\colon 2^{\Omega} \to [0;1]$

such that for the probability measure Prob the following holds.

1. $\forall A \subseteq \Omega\colon \text{Prob}(A) \geq 0$
2. $\forall A_1, A_2, \cdots \subseteq \Omega\colon\ (\forall i \neq j\colon A_i \cap A_j = \emptyset)$
   $\Rightarrow \left(\text{Prob}\left(\bigcup_i A_i\right) = \sum_i \text{Prob}(A_i)\right)$
3. $\text{Prob}(\Omega) = 1$

Notation $A \subseteq \Omega$ is called event

Direct Consequences and Examples

- $\text{Prob}(\emptyset) = 0$

Direct Consequences and Examples

- $\text{Prob}(\emptyset) = 0$
- $\forall A \subseteq \Omega\colon \text{Prob}(A) + \text{Prob}\left(\overline{A}\right) = 1$

Direct Consequences and Examples

- $\text{Prob}(\emptyset) = 0$
- $\forall A \subseteq \Omega\colon \text{Prob}(A) + \text{Prob}\left(\overline{A}\right) = 1$
- $\forall A \subseteq \Omega\colon \text{Prob}(A) = \sum_{s \in A} \text{Prob}(\{s\})$

Direct Consequences and Examples

- $\text{Prob}(\emptyset) = 0$
- $\forall A \subseteq \Omega\colon \text{Prob}(A) + \text{Prob}\left(\overline{A}\right) = 1$
- $\forall A \subseteq \Omega\colon \text{Prob}(A) = \sum_{s \in A} \text{Prob}(\{s\})$
- $\forall A_1, A_2, \ldots\colon \text{Prob}\left(\bigcup_i A_i\right) \leq \sum_i \text{Prob}(A_i)$ (Union Bound)

Direct Consequences and Examples

- $\text{Prob}(\emptyset) = 0$
- $\forall A \subseteq \Omega\colon \text{Prob}(A) + \text{Prob}\left(\overline{A}\right) = 1$
- $\forall A \subseteq \Omega\colon \text{Prob}(A) = \sum_{s \in A} \text{Prob}(\{s\})$
- $\forall A_1, A_2, \ldots\colon \text{Prob}\left(\bigcup_i A_i\right) \leq \sum_i \text{Prob}(A_i)$ (Union Bound)

Example Dice

- $\Omega = \{1, 2, 3, 4, 5, 6\}$
- $\forall s \in \Omega\colon \text{Prob}(s) = \frac{1}{6}$

Direct Consequences and Examples

- $\text{Prob}(\emptyset) = 0$
- $\forall A \subseteq \Omega\colon \text{Prob}(A) + \text{Prob}\left(\overline{A}\right) = 1$
- $\forall A \subseteq \Omega\colon \text{Prob}(A) = \sum_{s \in A} \text{Prob}(\{s\})$
- $\forall A_1, A_2, \ldots\colon \text{Prob}\left(\bigcup_i A_i\right) \leq \sum_i \text{Prob}(A_i)$ (Union Bound)

Example Dice

- $\Omega = \{1, 2, 3, 4, 5, 6\}$
- $\forall s \in \Omega\colon \text{Prob}(s) = \frac{1}{6}$

Example Two Fair Coins

- $\Omega = \{\mathsf{HH}, \mathsf{HT}, \mathsf{TH}, \mathsf{TT}\}$
- $\forall s \in \Omega\colon \text{Prob}(s) = \frac{1}{4}$

Independence of Events

Observation

$$\forall A, B \subseteq \Omega\colon \operatorname{Prob}(A \cap B) \leq \min\{\operatorname{Prob}(A), \operatorname{Prob}(B)\}$$

Independence of Events

Observation

$$\forall A, B \subseteq \Omega\colon \operatorname{Prob}(A \cap B) \leq \min\{\operatorname{Prob}(A), \operatorname{Prob}(B)\}$$

Definition

Events $A_1, A_2, \ldots, A_n \subseteq \Omega$ are called (completely) independent

$$:\Leftrightarrow \forall I \subseteq \{1, 2, \ldots, n\}\colon \operatorname{Prob}\left(\bigcap_{i \in I} A_i\right) = \prod_{i \in I} \operatorname{Prob}(A_i)$$

Independence of Events

Observation

$\forall A, B \subseteq \Omega\colon \operatorname{Prob}(A \cap B) \leq \min\{\operatorname{Prob}(A), \operatorname{Prob}(B)\}$

Definition

Events $A_1, A_2, \ldots, A_n \subseteq \Omega$ are called (completely) independent

$:\Leftrightarrow \forall I \subseteq \{1, 2, \ldots, n\}\colon \operatorname{Prob}\left(\bigcap_{i \in I} A_i\right) = \prod_{i \in I} \operatorname{Prob}(A_i)$

Events $A_1, A_2, \ldots, A_n \subseteq \Omega$ are called pairwise independent

$:\Leftrightarrow \forall i_1 \neq i_2 \in \{1, 2, \ldots, n\}\colon \operatorname{Prob}(A_{i_1} \cap A_{i_2}) = \operatorname{Prob}(A_{i_1}) \cdot \operatorname{Prob}(A_{i_2})$

Independence of Events

Observation

$\forall A, B \subseteq \Omega\colon \operatorname{Prob}(A \cap B) \leq \min\{\operatorname{Prob}(A), \operatorname{Prob}(B)\}$

Definition

Events $A_1, A_2, \ldots, A_n \subseteq \Omega$ are called (completely) independent

$:\Leftrightarrow \forall I \subseteq \{1, 2, \ldots, n\}\colon \operatorname{Prob}\left(\bigcap_{i \in I} A_i\right) = \prod_{i \in I} \operatorname{Prob}(A_i)$

Events $A_1, A_2, \ldots, A_n \subseteq \Omega$ are called pairwise independent

$:\Leftrightarrow \forall i_1 \neq i_2 \in \{1, 2, \ldots, n\}\colon \operatorname{Prob}(A_{i_1} \cap A_{i_2}) = \operatorname{Prob}(A_{i_1}) \cdot \operatorname{Prob}(A_{i_2})$

Is "completely independent" and "pairwise independent" different?

Independence

Example Two Fair Coins

Consider events

- first coin lands heads $A_1 =$

Independence

Example Two Fair Coins

Consider events

- first coin lands heads $A_1 = \{\mathsf{HT}, \mathsf{HH}\}$

Independence

Example Two Fair Coins

Consider events

- first coin lands heads $A_1 = \{\mathsf{HT}, \mathsf{HH}\}$
- second coin lands heads $A_2 =$

Independence

Example Two Fair Coins

Consider events

- first coin lands heads $A_1 = \{\mathsf{HT}, \mathsf{HH}\}$
- second coin lands heads $A_2 = \{\mathsf{TH}, \mathsf{HH}\}$

Independence

Example Two Fair Coins

Consider events

- first coin lands heads $A_1 = \{\mathsf{HT}, \mathsf{HH}\}$
- second coin lands heads $A_2 = \{\mathsf{TH}, \mathsf{HH}\}$
- both coins land equally $A_3 =$

Independence

Example Two Fair Coins

Consider events

- first coin lands heads $A_1 = \{\mathsf{HT}, \mathsf{HH}\}$
- second coin lands heads $A_2 = \{\mathsf{TH}, \mathsf{HH}\}$
- both coins land equally $A_3 = \{\mathsf{TT}, \mathsf{HH}\}$

Independence

Example Two Fair Coins

Consider events

- first coin lands heads $A_1 = \{\mathsf{HT}, \mathsf{HH}\}$
- second coin lands heads $A_2 = \{\mathsf{TH}, \mathsf{HH}\}$
- both coins land equally $A_3 = \{\mathsf{TT}, \mathsf{HH}\}$

Observe $\text{Prob}(A_1) = \text{Prob}(A_2) = \text{Prob}(A_3) = \frac{1}{2}$

Independence

Example Two Fair Coins

Consider events

- first coin lands heads $A_1 = \{\mathsf{HT}, \mathsf{HH}\}$
- second coin lands heads $A_2 = \{\mathsf{TH}, \mathsf{HH}\}$
- both coins land equally $A_3 = \{\mathsf{TT}, \mathsf{HH}\}$

Observe $\text{Prob}(A_1) = \text{Prob}(A_2) = \text{Prob}(A_3) = \frac{1}{2}$

Observe A_1, A_2, A_3 pairwise independent

$\text{Prob}(A_1 \cap A_2) = \text{Prob}(\mathsf{HH}) = \frac{1}{4} = \text{Prob}(A_1) \cdot \text{Prob}(A_2)$

Independence

Example Two Fair Coins

Consider events

- first coin lands heads $A_1 = \{\mathsf{HT}, \mathsf{HH}\}$
- second coin lands heads $A_2 = \{\mathsf{TH}, \mathsf{HH}\}$
- both coins land equally $A_3 = \{\mathsf{TT}, \mathsf{HH}\}$

Observe $\text{Prob}(A_1) = \text{Prob}(A_2) = \text{Prob}(A_3) = \frac{1}{2}$

Observe A_1, A_2, A_3 pairwise independent

$$\text{Prob}(A_1 \cap A_2) = \text{Prob}(\mathsf{HH}) = \frac{1}{4} = \text{Prob}(A_1) \cdot \text{Prob}(A_2)$$
$$\text{Prob}(A_1 \cap A_3) = \text{Prob}(\mathsf{HH}) = \frac{1}{4} = \text{Prob}(A_1) \cdot \text{Prob}(A_3)$$

Independence

Example Two Fair Coins

Consider events

- first coin lands heads $A_1 = \{\mathsf{HT}, \mathsf{HH}\}$
- second coin lands heads $A_2 = \{\mathsf{TH}, \mathsf{HH}\}$
- both coins land equally $A_3 = \{\mathsf{TT}, \mathsf{HH}\}$

Observe $\text{Prob}(A_1) = \text{Prob}(A_2) = \text{Prob}(A_3) = \frac{1}{2}$

Observe A_1, A_2, A_3 pairwise independent

$$\text{Prob}(A_1 \cap A_2) = \text{Prob}(\mathsf{HH}) = \frac{1}{4} = \text{Prob}(A_1) \cdot \text{Prob}(A_2)$$
$$\text{Prob}(A_1 \cap A_3) = \text{Prob}(\mathsf{HH}) = \frac{1}{4} = \text{Prob}(A_1) \cdot \text{Prob}(A_3)$$
$$\text{Prob}(A_2 \cap A_3) = \text{Prob}(\mathsf{HH}) = \frac{1}{4} = \text{Prob}(A_2) \cdot \text{Prob}(A_3)$$

Independence

Example Two Fair Coins

Consider events

- first coin lands heads $A_1 = \{\mathsf{HT}, \mathsf{HH}\}$
- second coin lands heads $A_2 = \{\mathsf{TH}, \mathsf{HH}\}$
- both coins land equally $A_3 = \{\mathsf{TT}, \mathsf{HH}\}$

Observe $\text{Prob}(A_1) = \text{Prob}(A_2) = \text{Prob}(A_3) = \frac{1}{2}$

Observe A_1, A_2, A_3 pairwise independent

$$\text{Prob}(A_1 \cap A_2) = \text{Prob}(\mathsf{HH}) = \frac{1}{4} = \text{Prob}(A_1) \cdot \text{Prob}(A_2)$$
$$\text{Prob}(A_1 \cap A_3) = \text{Prob}(\mathsf{HH}) = \frac{1}{4} = \text{Prob}(A_1) \cdot \text{Prob}(A_3)$$
$$\text{Prob}(A_2 \cap A_3) = \text{Prob}(\mathsf{HH}) = \frac{1}{4} = \text{Prob}(A_2) \cdot \text{Prob}(A_3)$$

Observe A_1, A_2, A_3 not completely independent

$$\text{Prob}(A_1 \cap A_2 \cap A_3) = \text{Prob}(\mathsf{HH}) = \frac{1}{4} \neq \frac{1}{8}$$
$$= \text{Prob}(A_1) \cdot \text{Prob}(A_2) \cdot \text{Prob}(A_3)$$

Working with Probability

as always numbers especially important

Working with Probability

as always numbers especially important

Definition

For a probability space (Ω, Prob), a random variable X is a function $X : \Omega \to \mathbb{R}$.

Working with Probability

as always numbers especially important

Definition

For a probability space (Ω, Prob), a random variable X is a function $X : \Omega \to \mathbb{R}$.

The probability distribution of X is given by

$\forall x_i \in \mathbb{R}: \text{Prob}(X = x_i) = \text{Prob}(\{s \in \Omega : X(s) = x_i\})$.

Working with Probability

as always numbers especially important

Definition

For a probability space (Ω, Prob), a random variable X is a function $X : \Omega \to \mathbb{R}$.

The probability distribution of X is given by

$\forall x_i \in \mathbb{R}: \text{Prob}(X = x_i) = \text{Prob}(\{s \in \Omega : X(s) = x_i\})$.

The expected value $\mathrm{E}(X)$ is defined by

$\mathrm{E}(X) := \sum_{s \in \Omega} X(s) \cdot \text{Prob}(s)$.

Working with Probability

as always numbers especially important

Definition

For a probability space (Ω, Prob), a random variable X is a function $X : \Omega \to \mathbb{R}$.

The probability distribution of X is given by

$\forall x_i \in \mathbb{R} : \text{Prob}(X = x_i) = \text{Prob}(\{s \in \Omega : X(s) = x_i\})$.

The expected value $\text{E}(X)$ is defined by

$$\text{E}(X) := \sum_{s \in \Omega} X(s) \cdot \text{Prob}(s).$$

One Fair Die

$X :=$ number of pips

Working with Probability

as always numbers especially important

Definition

For a probability space (Ω, Prob), a random variable X is a function $X : \Omega \to \mathbb{R}$.

The probability distribution of X is given by

$\forall x_i \in \mathbb{R}: \text{Prob}(X = x_i) = \text{Prob}(\{s \in \Omega : X(s) = x_i\})$.

The expected value $\text{E}(X)$ is defined by

$\text{E}(X) := \sum_{s \in \Omega} X(s) \cdot \text{Prob}(s)$.

One Fair Die

$X :=$ number of pips

$\text{E}(X) = \sum_{i=1}^{6} i \cdot \frac{1}{6} = \frac{7}{2} = 3.5$

Working with Probability

as always numbers especially important

Definition

For a probability space (Ω, Prob), a random variable X is a function $X : \Omega \to \mathbb{R}$.
The probability distribution of X is given by
$\forall x_i \in \mathbb{R}: \text{Prob}(X = x_i) = \text{Prob}(\{s \in \Omega : X(s) = x_i\})$.
The expected value $\text{E}(X)$ is defined by
$\text{E}(X) := \sum_{s \in \Omega} X(s) \cdot \text{Prob}(s)$.

One Fair Die

$X :=$ number of pips
$\text{E}(X) = \sum_{i=1}^{6} i \cdot \frac{1}{6} = \frac{7}{2} = 3.5$

Observation Expected value need not be possible event.

Summary & Take Home Message

Things to remember

Summary & Take Home Message

Things to remember

- evolutionary algorithms $(\mu+\lambda)$ EA, (μ, λ) EA, steady state GA

Summary & Take Home Message

Things to remember

- evolutionary algorithms $(\mu+\lambda)$ EA, (μ, λ) EA, steady state GA
- applying randomised search heuristics (and not doing so)

Summary & Take Home Message

Things to remember

- evolutionary algorithms $(\mu+\lambda)$ EA, (μ, λ) EA, steady state GA
- applying randomised search heuristics (and not doing so)
- upcoming attractions

Summary & Take Home Message

Things to remember

- evolutionary algorithms $(\mu+\lambda)$ EA, (μ, λ) EA, steady state GA
- applying randomised search heuristics (and not doing so)
- upcoming attractions
- introduction to probability

Summary & Take Home Message

Things to remember

- evolutionary algorithms $(\mu+\lambda)$ EA, (μ, λ) EA, steady state GA
- applying randomised search heuristics (and not doing so)
- upcoming attractions
- introduction to probability
  - probability space

Summary & Take Home Message

Things to remember

- evolutionary algorithms $(\mu+\lambda)$ EA, (μ, λ) EA, steady state GA
- applying randomised search heuristics (and not doing so)
- upcoming attractions
- introduction to probability
  - probability space
  - events

Summary & Take Home Message

Things to remember

- evolutionary algorithms $(\mu+\lambda)$ EA, (μ, λ) EA, steady state GA
- applying randomised search heuristics (and not doing so)
- upcoming attractions
- introduction to probability
  - probability space
  - events
  - independence

Summary & Take Home Message

Things to remember

- evolutionary algorithms $(\mu+\lambda)$ EA, (μ, λ) EA, steady state GA
- applying randomised search heuristics (and not doing so)
- upcoming attractions
- introduction to probability
  - probability space
  - events
  - independence
  - random variable

Summary & Take Home Message

Things to remember

- evolutionary algorithms $(\mu+\lambda)$ EA, (μ, λ) EA, steady state GA
- applying randomised search heuristics (and not doing so)
- upcoming attractions
- introduction to probability
  - probability space
  - events
  - independence
  - random variable

Take Home Message

Summary & Take Home Message

Things to remember

- evolutionary algorithms $(\mu+\lambda)$ EA, (μ, λ) EA, steady state GA
- applying randomised search heuristics (and not doing so)
- upcoming attractions
- introduction to probability
  - probability space
  - events
  - independence
  - random variable

Take Home Message

- RSH are easy to implement and easy to apply.

Summary & Take Home Message

Things to remember

- evolutionary algorithms $(\mu+\lambda)$ EA, (μ, λ) EA, steady state GA
- applying randomised search heuristics (and not doing so)
- upcoming attractions
- introduction to probability
  - probability space
  - events
  - independence
  - random variable

Take Home Message

- RSH are easy to implement and easy to apply.
- Probability theory is not very complicated.