

# CS4618 Artificial Intelligence I

Today: Evolutionary Algorithms  
Introduction to Probability

Thomas Jansen

October 19<sup>th</sup>

# Plans for Today

- 1 Evolutionary Algorithms  
Concrete EAs
- 2 Outlook  
Motivation Randomised Search Heuristics  
Upcoming Attractions
- 3 Introduction to Probability  
Probability Theory Basics
- 4 Random Variables and Expectations  
Introduction
- 5 Summary  
Summary & Take Home Message

# 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

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.     $y := \text{standard bit mutation}(x_t)$
4.     $t := t + 1$ ; 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

**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  $x_1, x_2, \dots, x_\mu \in S$  uniformly at random.
2. Repeat
3.    For  $i \in \{1, 2, \dots, \lambda\}$  do
4.        Select  $z \in \{x_1, x_2, \dots, x_\mu\}$  uniformly at random.
5.         $y_i :=$  standard bit mutation( $z$ )
6.    Select new  $x_1, x_2, \dots, x_\mu$  out of best  $x_1, x_2, \dots, x_\mu$   
        and  $y_1, y_2, \dots, y_\lambda$ .
7. Until 'decide to stop'
8. Output  $x_i$  with  $f(x_i) = \max\{f(x_j) \mid 1 \leq j \leq \mu\}$ .

$(\mu, \lambda)$  EA

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

 $(\mu, \lambda)$  EA

1. Choose  $x_1, x_2, \dots, x_\mu \in S$  uniformly at random.
2. Repeat
3.    For  $i \in \{1, 2, \dots, \lambda\}$  do
4.        Select  $z \in \{x_1, x_2, \dots, x_\mu\}$  uniformly at random.
5.         $y_i :=$  standard bit mutation( $z$ )
6.    Select new  $x_1, x_2, \dots, x_\mu$  out of best  $y_1, y_2, \dots, y_\lambda$ .
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

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

- only **most basic** variants considered, much more involved variants known and used
- extreme variants  $(1+1)$  EA,  $(\mu+1)$  EA,  $(1+\lambda)$  EA,  $(1, \lambda)$  EA are of special interest for understanding evolutionary algorithms
- steady state GA is  $(\mu+1)$  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?

## Observations

- optimisation framework  $(S, f: S \rightarrow \mathbb{R})$  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

- 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

## Definition

A **probability space** consists of

- a countable **sample space**  $\Omega$  and
- a **probability measure**  $\text{Prob}: 2^\Omega \rightarrow [0; 1]$

such that for the probability measure  $\text{Prob}$  the following holds.

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

**Notation**  $A \subseteq \Omega$  is called **event**

## Direct Consequences and Examples

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

### Example Dice

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

### Example Two Fair Coins

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

# Independence of Events

## Observation

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

## Definition

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

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

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

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

Is “completely independent” and “pairwise independent” different?

# Independence

## Example Two Fair Coins

Consider events

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

**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

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

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

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

**Observe**  $A_1, A_2, A_3$  **not** completely independent

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

$$= \text{Prob}(A_1) \cdot \text{Prob}(A_2) \cdot \text{Prob}(A_3)$$

# Working with Probability

as always    numbers especially important

## Definition

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

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**  $\mathbb{E}(X)$  is defined by

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

## One Fair Die

$X :=$  number of pips

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

**Observation**    Expected value need not be possible event.

# Summary & Take Home Message

## Things to remember

- evolutionary algorithms  $(\mu+\lambda)$  EA,  $(\mu, \lambda)$  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.