Automating Algorithm Design through Genetic Programming Hyper-Heuristics

Elsa Browning

Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

April 15, 2017
Morris, MN

What does the title mean?

- Reducing the human component in algorithm design

https://scratch.mit.edu/discuss/m/topic/200574/

What does the title mean?

- Reducing the human component in algorithm design
- More work at the beginning, more possibilities

https://scratch.mit.edu/discuss/m/topic/200574/

What does the title mean?

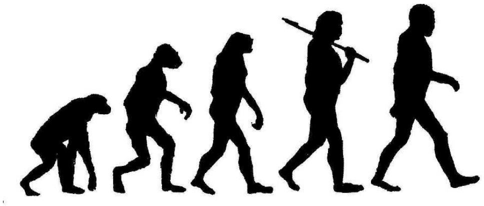
- Reducing the human component in algorithm design
- More work at the beginning, more possibilities
- Genetic programming hyper-heuristics as a method to the madness

https://scratch.mit.edu/discuss/m/topic/200574/

Outline

Outline

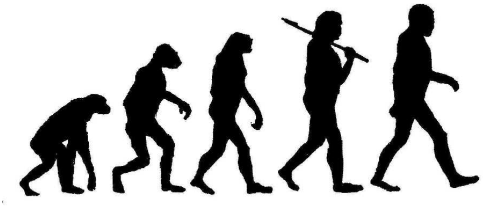
Evolutionary Computation



https://www.spigotmc.org/attachments/evolution-jpg.137048/

- Subfield of Artificial Intelligence

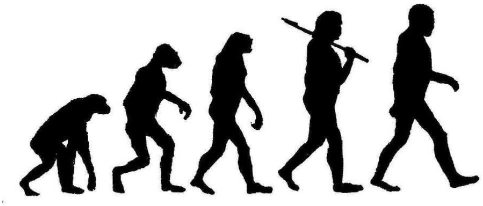
Evolutionary Computation



https://www.spigotmc.org/attachments/evolution-jpg.137048/

- Subfield of Artificial Intelligence
- Algorithms based on biological evolution

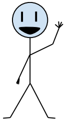
Evolutionary Computation



https://www.spigotmc.org/attachments/evolution-jpg.137048/

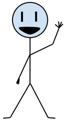
- Subfield of Artificial Intelligence
- Algorithms based on biological evolution
- Uses lots of terminology from biology, doesn't always mean same thing as term means in biology.

Evolutionary Computation – Terminology

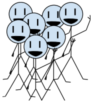


- **Individual** – a potential solution to a problem (or set of problems)

Evolutionary Computation – Terminology

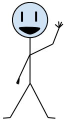


- **Individual** – a potential solution to a problem (or set of problems)

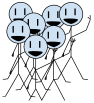


- **Population** – a group of individuals

Evolutionary Computation – Terminology



- **Individual** – a potential solution to a problem (or set of problems)

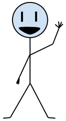


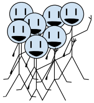
- **Population** – a group of individuals

- **Fit** – how well suited an individual is at solving a problem

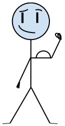
Evolutionary Computation – Terminology

- **Individual** – a potential solution to a problem (or set of problems)

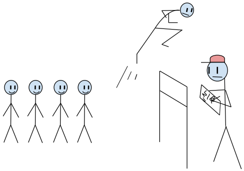




- **Population** – a group of individuals



- **Fit** – how well suited an individual is at solving a problem



- **Fitness Test** – a set of tests to determine how fit an individual is.

Evolutionary Computation – Terminology

- **Mutation** – an insertion, deletion, or small change in the code of an individual, creating a new individual

Evolutionary Computation – Terminology

- **Mutation** – an insertion, deletion, or small change in the code of an individual, creating a new individual
- **Sexual reproduction** – when two or more individuals are munged together to create a new individual

Evolutionary Computation – Terminology

- **Mutation** – an insertion, deletion, or small change in the code of an individual, creating a new individual
- **Sexual reproduction** – when two or more individuals are munged together to create a new individual
- **Generation** – a population of individuals

Evolutionary Computation – Terminology

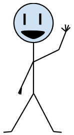
- **Mutation** – an insertion, deletion, or small change in the code of an individual, creating a new individual
- **Sexual reproduction** – when two or more individuals are munged together to create a new individual
- **Generation** – a population of individuals
- **Global optima** – best solution (or solutions) possible

Evolutionary Computation – Terminology

If individual A experiences a mutation to create individual B, then:

Evolutionary Computation – Terminology

If individual A experiences a mutation to create individual B, then:

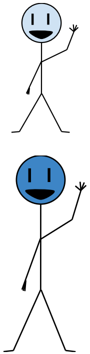


- **Parent** – Individual A

Evolutionary Computation – Terminology

If individual A experiences a mutation to create individual B, then:

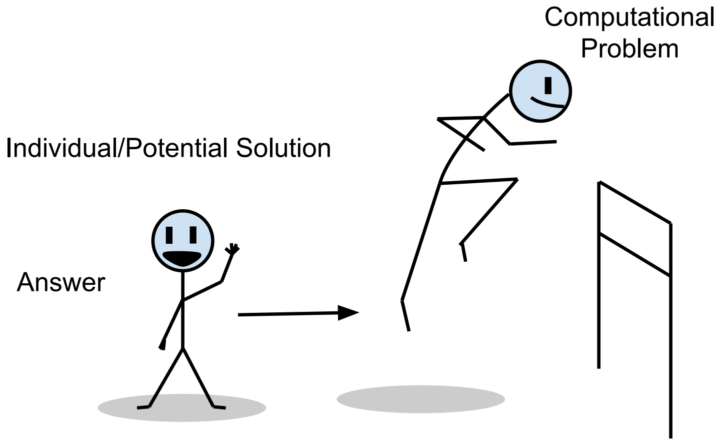
- **Parent** – Individual A
- **Child** – Individual B



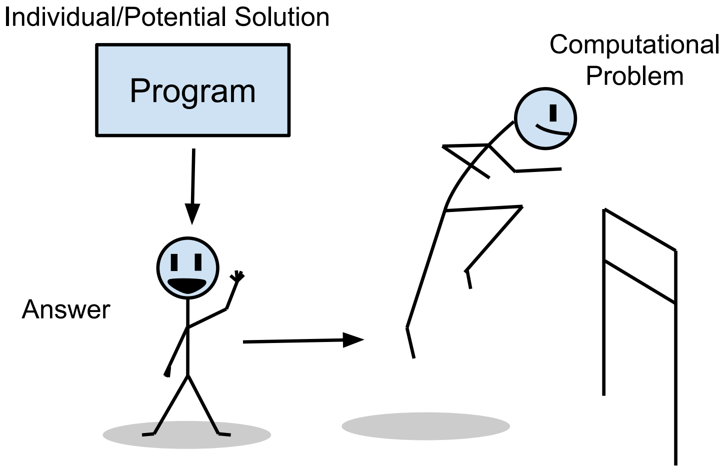
Genetic Programming

A family of algorithms in Evolutionary Computation that uses biological techniques to create programs to solve computational problems.

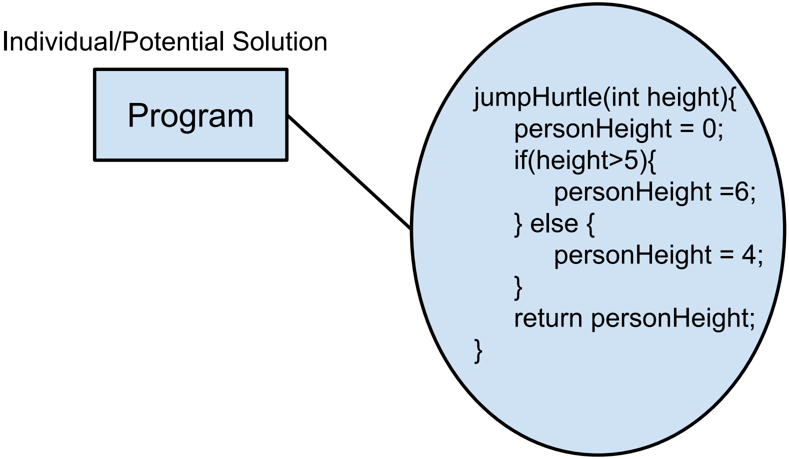
Genetic Programming – structure for evolution



Genetic Programming – structure for evolution

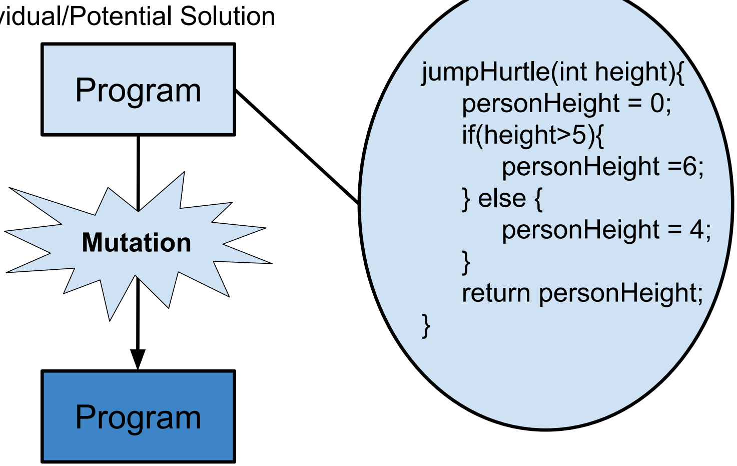


Genetic Programming – structure for evolution



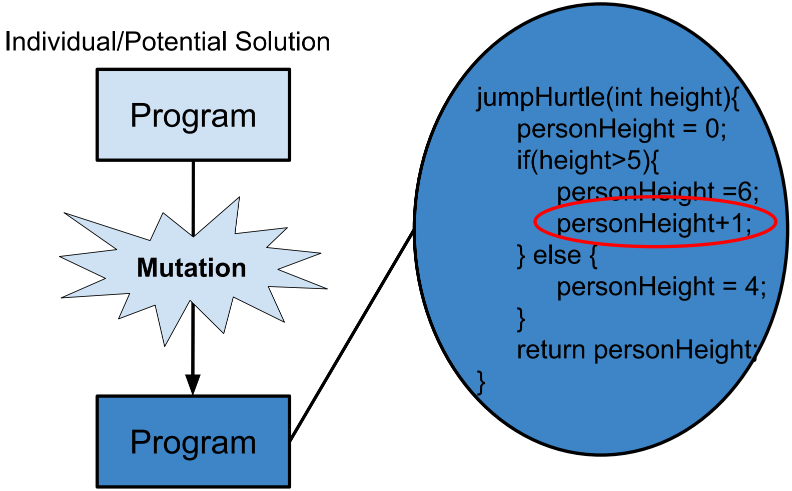
```
jumpHurtle(int height){
    personHeight = 0;
    if(height>5){
        personHeight =6;
    } else {
        personHeight = 4;
    }
    return personHeight;
}
```

Genetic Programming – structure for evolution



```
jumpHurtle(int height){
    personHeight = 0;
    if(height>5){
        personHeight =6;
    } else {
        personHeight = 4;
    }
    return personHeight;
}
```

Genetic Programming – structure for evolution



```
jumpHurtle(int height){
    personHeight = 0;
    if(height>5){
        personHeight =6;
        personHeight+1;
    } else {
        personHeight = 4;
    }
    return personHeight;
}
```

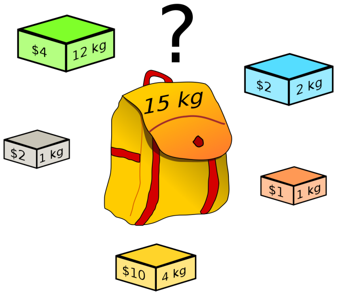
Outline

Heuristic

Heuristic – a function that ranks alternatives in a search algorithm at each branching step and uses that information to choose which branch to follow.

Heuristic

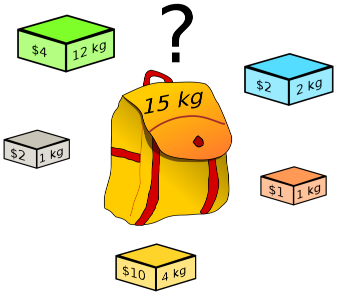
Heuristic – a function that ranks alternatives in a search algorithm at each branching step and uses that information to choose which branch to follow.



https://upload.wikimedia.org/wikipedia/commons/thumb/f/fd/Knapsack.svg/1200px-Knapsack.svg.png

Heuristic

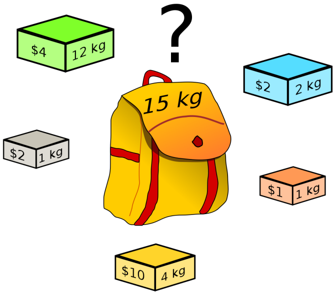
"if knapsack is not full, put largest valued item into knapsack. If this item would cause knapsack to be overweight, take the next highest valued item and put it into the knapsack. Repeat until all items are gone"



https://upload.wikimedia.org/wikipedia/commons/thumb/f/fd/Knapsack.svg/1200px-Knapsack.svg.png

Heuristic

“if knapsack is not full, put largest valued item into knapsack. If this item would cause knapsack to be overweight, take the next highest valued item and put it into knapsack. Repeat until all items are gone or knapsack is full"

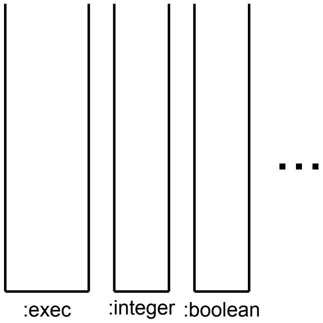


https://upload.wikimedia.org/wikipedia/commons/thumb/f/fd/Knapsack.svg/1200px-Knapsack.svg.png

Outline

Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

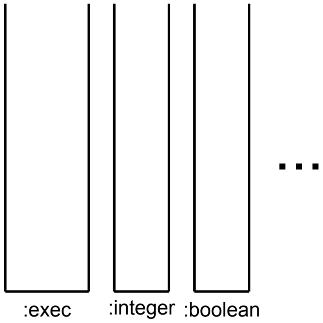


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_equal)
```

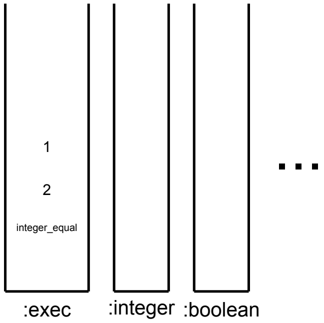


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_equal)
```

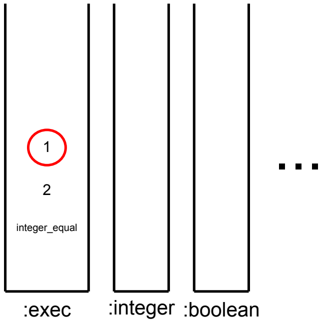


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_equal)
```

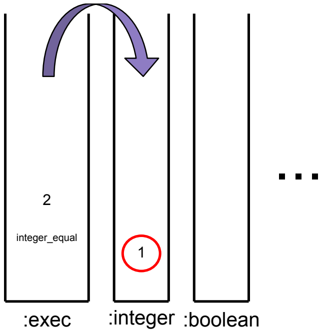


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_equal)
```

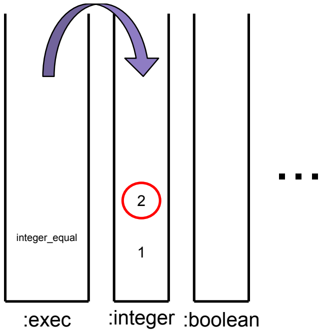


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_equal)
```

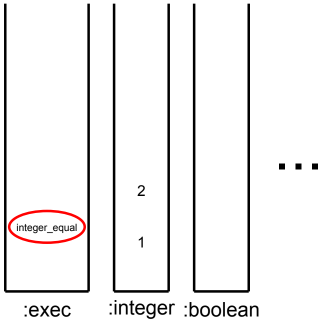


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_equal)
```

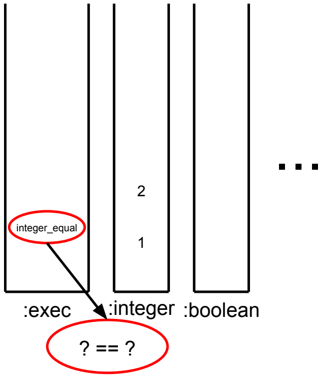


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_equal)
```

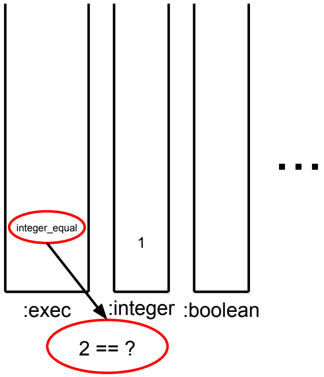


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_equal)
```

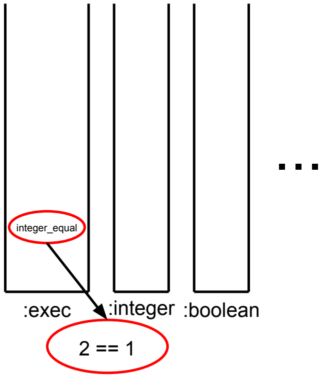


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_equal)
```

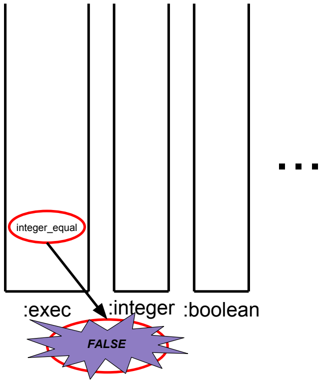


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_equal)
```

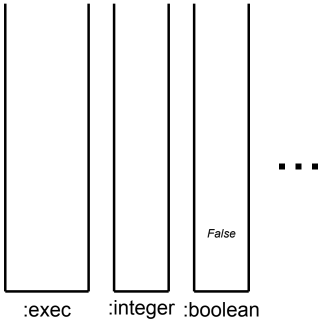


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_equal)
```



Outline

What is Autoconstruction?

- Autoconstruction is a type of GPHH

What is Autoconstruction?

- Autoconstruction is a type of GPHH
- In most GPHH, the individual programs are evolving, but everything else is specified by the engineer; in autoconstruction, evolution is evolving as well.

What is Autoconstruction?

- Autoconstruction is a type of GPHH
- In most GPHH, the individual programs are evolving, but everything else is specified by the engineer; in autoconstruction, evolution is evolving as well.
- Programs are responsible for evolving solutions *and* responsible for evolving their offspring.

AutoDoG

Outline