JDriven

MAKING DARWIN PROUD

Coding Evolutionary Algorithms

bas.knopper@jdriven.com @BWKnopper github.com/bknopper

JDriven

SPECIAL DAY

bas.knopper@jdriven.com @BWKnopper github.com/bknopper

JDriven

bas.knopper@jdriven.com @BWKnopper github.com/bknopper

JDriven

Let's Sing Happy Birthday!

Happy Birthday to You

Happy Birthday to You

Happy Birthday Dear **Olivier**

Happy Birthday to You.

bas.knopper@jdriven.com @BWKnopper github.com/bknopper

JDriven

bas.knopper@jdriven.com @BWKnopper github.com/bknopper

JDriven

MAKING DARWIN PROUD

Coding Evolutionary Algorithms

bas.knopper@jdriven.com @BWKnopper github.com/bknopper

Let me introduce myself…

- Bas W. Knopper
- Dutch
- Java Developer
  - JDriven (IT Services Company)
  - 10+ years Java exp
- JavaOne, J-Fall, GeeCon, JFokus Speaker
- AI enthousiast
  - Soft spot for Evolutionary Algorithms

bas.knopper@jdriven.com @BWKnopper github.com/bknopper

JDriven

What I would like to accomplish…

- Interest
- Understanding
- How & when
- Add to toolbox
- Attention

#JFokus #EvolutionaryAlgorithms @BWKnopper

bas.knopper@jdriven.com @BWKnopper github.com/bknopper

JDriven

Agenda

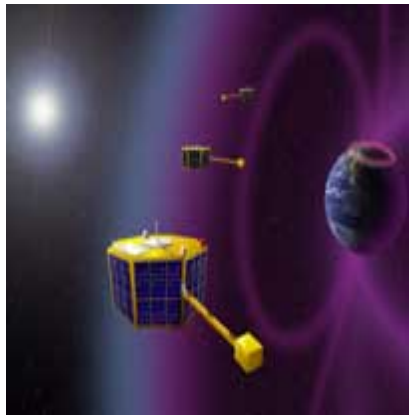
- Singing Happy Birthday (Thanks!)
- Introduction
- NASA
- Evolution Concepts
- Puzzle Solving Time: Traveling Salesman Problem
  - Evolutionary Algorithm Design
  - Plain Java Code
  - Demo!
- Frameworks
- Checklist

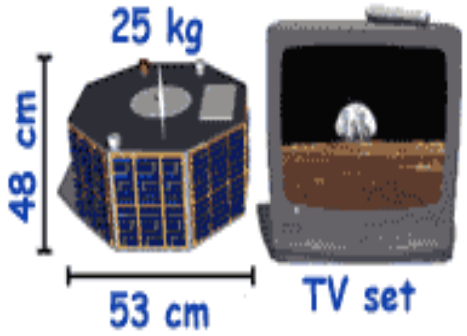
bas.knopper@jdriven.com @BWKnopper github.com/bknopper

NASA

- Space Technology 5 mission
  - launched March 22, 2006, and completed June 20, 2006

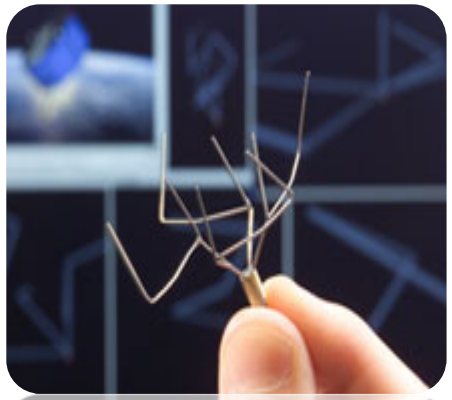
- Three full service 25-kilogram-class spacecraft

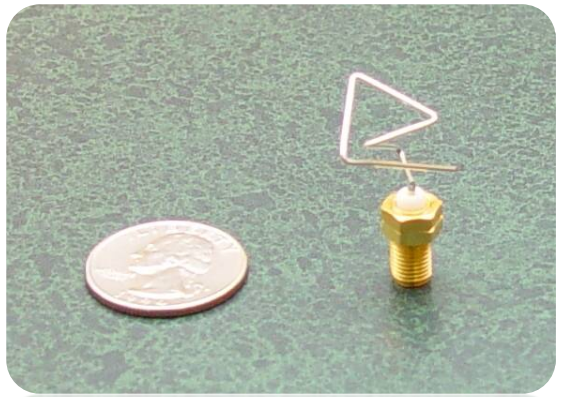




NASA Continued

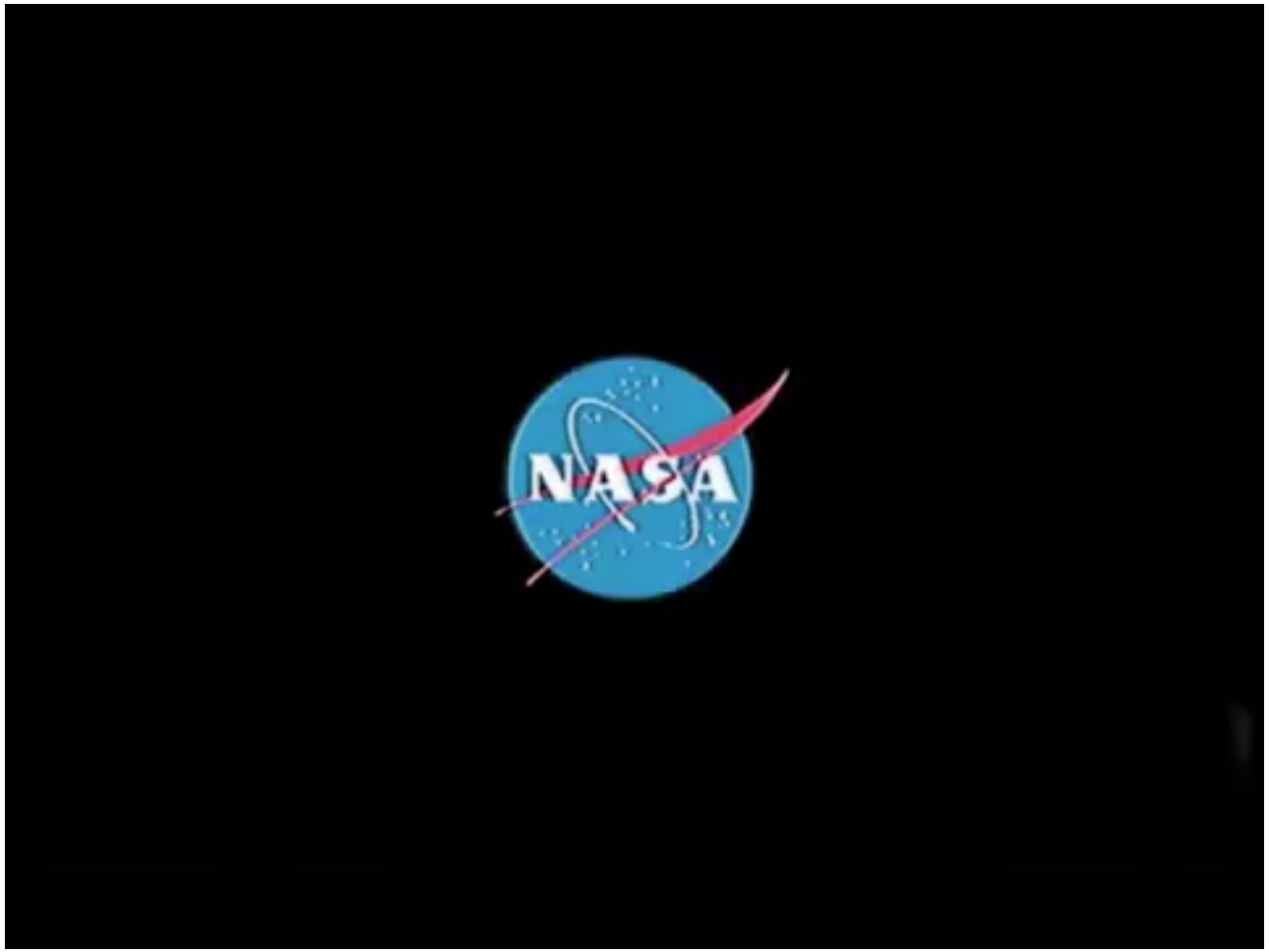
- Needs even smaller antenna
- That still functions according to spec
- Need for solution that's not easy to engineer
- So they used an EA that made these:





bas.knopper@jdriven.com @BWKnopper github.com/bknopper

JDriven



bas.knopper@jdriven.com @BWKnopper github.com/bknopper

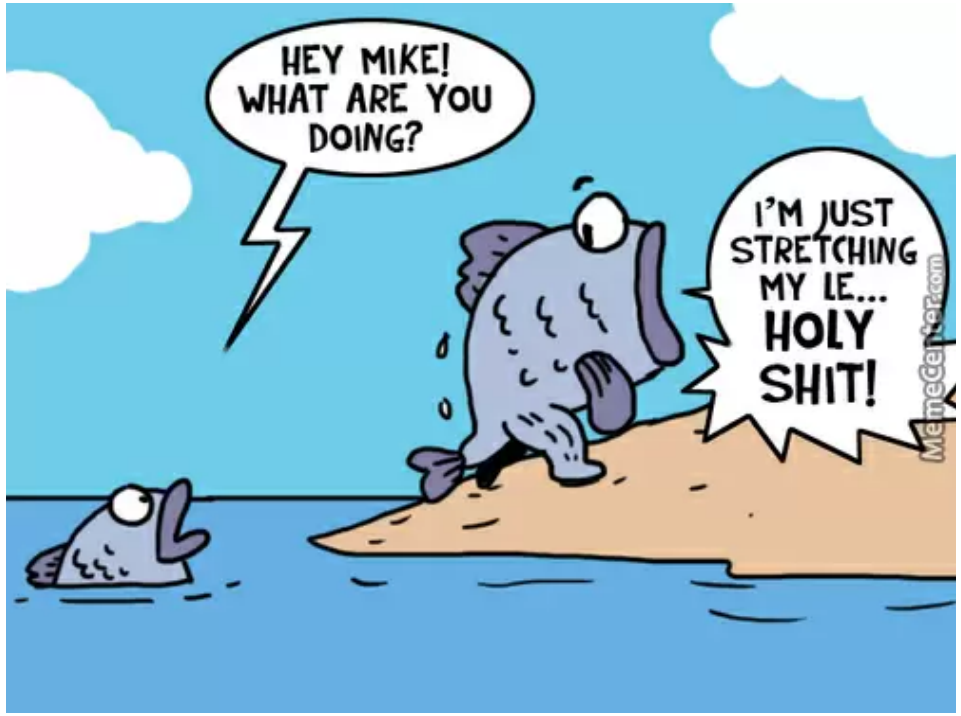
JDriven

Recap

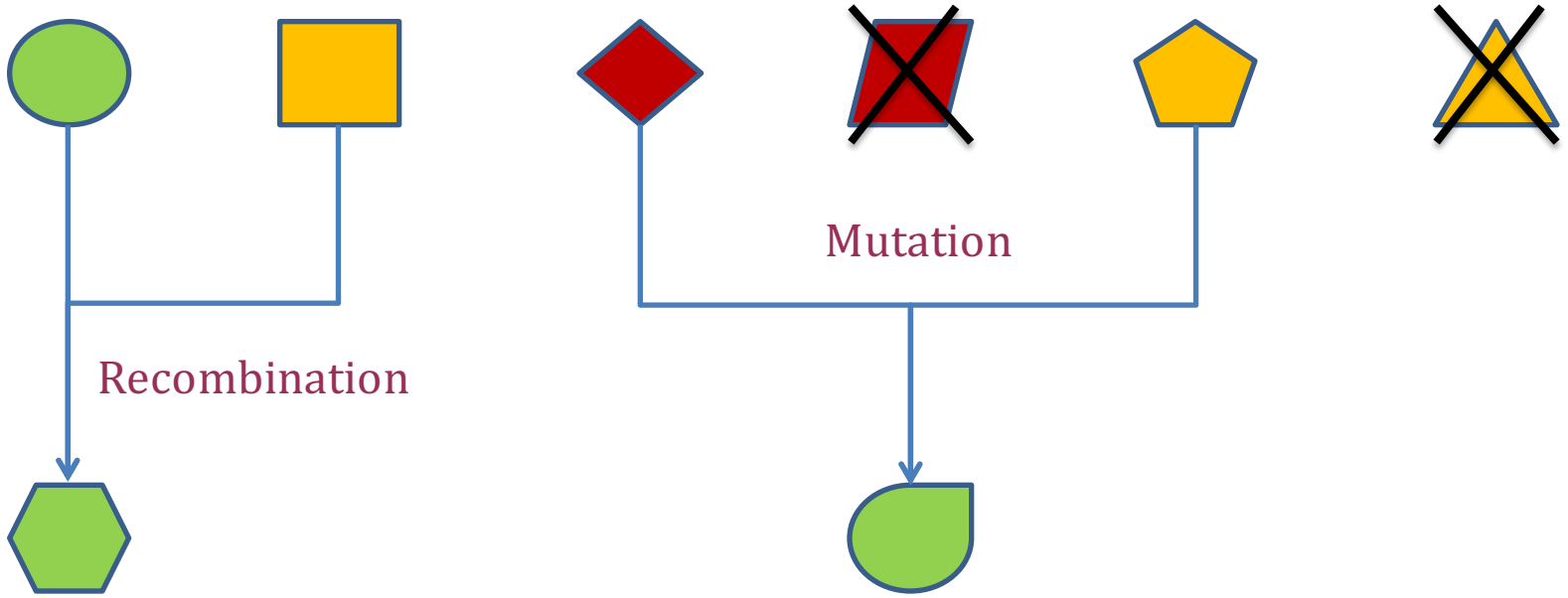
- Powerful example
- First evolved object to travel through space

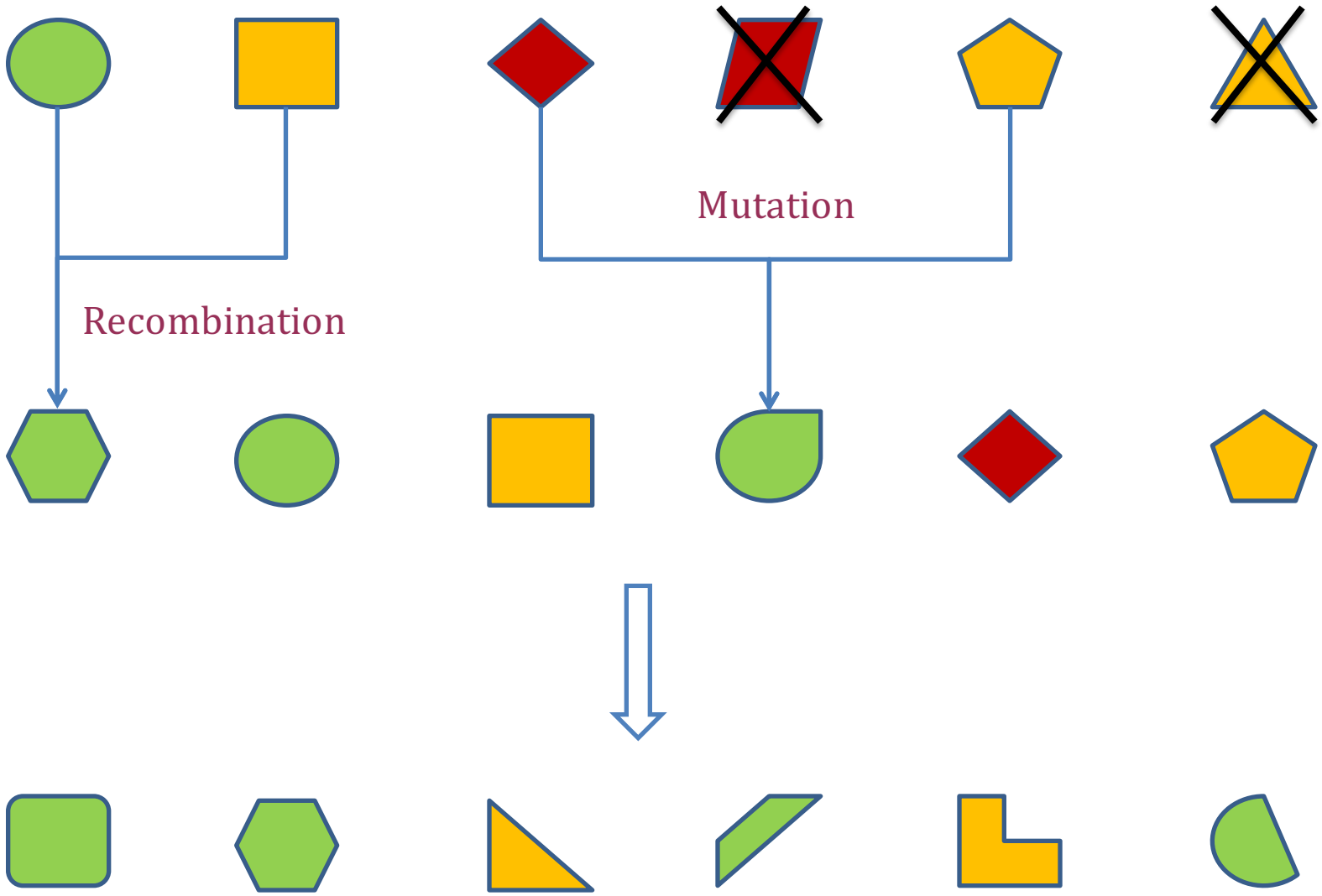
- How?

Evolution


HEY MIKE! WHAT ARE YOU DOING?
I'M JUST STRETCHING MY LE... HOLY SHIT!

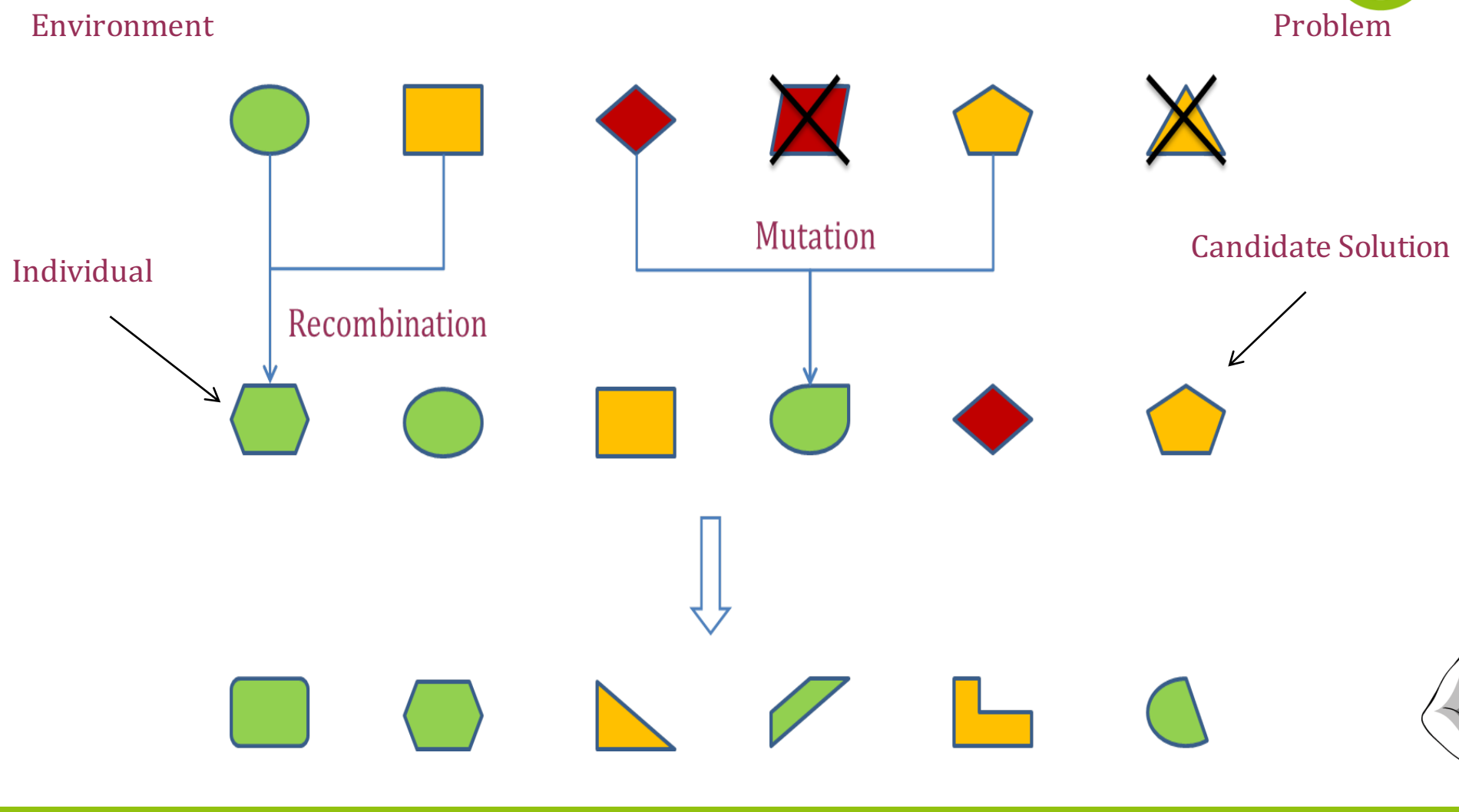
Evolution - “Survival of the fittest”


Recombination
Mutation



bas.knopper@jdriven.com
@BWKnopper
github.com/bknopper

From evolution to problem solving



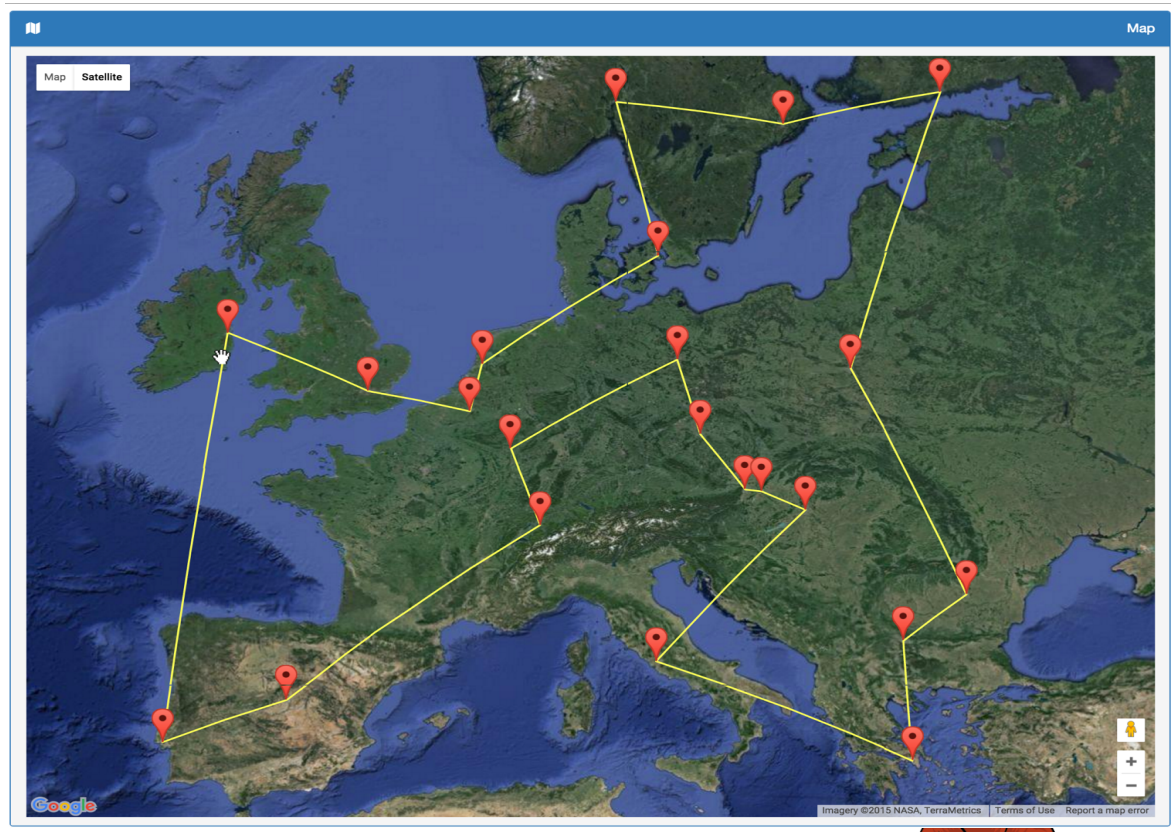
JDriven

Puzzle solving time!

- More down to earth example
  - Travelling Salesman Problem

Travelling Salesman Problem

- Given *n* cities
  - *n = number of cities to visit*
- Find (optimal) route for visiting all cities
  - Visit every city only once
  - Return to origin city
- Search space is huge
  - For 30 cities there are 30! ≈ 10^32 possible routes



That's 100.000.000.000.000.000.000.000.000.000.000 possible routes!

Brute force might not be the best solution...

bas.knopper@jdriven.com @BWKnopper github.com/bknopper

JDriven

Puzzle solving time!

- More down to earth example
  - Travelling Salesman Problem

- Use case to show you
  - Evolutionary Algorithm Design
  - Plain Java Code
  - Demo

EA Algorithm (Pseudocode)

INITIALISE population with random candidate solutions;

EA Algorithm (Pseudocode)

INITIALISE population with random candidate solutions;

Candidate Solution - Representation

- $n = 6$

- Label cities 1,2,3,4,5,6
- And base city 1

- Candidate Solution
  - Signifying the route

- for $n = 10$

- Enables
  - Calculating distance (fitness function)
  - Mutation
  - Recombination

EA Algorithm (Pseudocode)

INITIALISE population with random candidate solutions;

EA Algorithm (Pseudocode)

INITIALISE population with random candidate solutions;
EVALUATE each candidate;

Evaluation Function (Fitness Function)

- Summed distance:
  - d1,2 + d2,4 + … + d5,1
  - Used NASA WorldWind
- Minimize!

```
/**
 * Calculates the total distance of the whole route and stores it as this
 * candidate solution's fitness
 */
private void calculateFitness() {









}
```

```java
/**
 * Calculates the total distance of the whole route and stores it as this
 * candidate solution's fitness
 */
private void calculateFitness() {

   /* initialize total distance */
   double totalDistance = 0;









   /* calculate total distance */









   /* store totalDistance as this candidate solution's fitness */
   this.fitness = totalDistance;
}
```

```java
/**
 * Calculates the total distance of the whole route and stores it as this
 * candidate solution's fitness
 */
private void calculateFitness() {

   /* initialize total distance */
   double totalDistance = 0;

   /*
    * For all Cities in the route (except the last one) get the distance between this
    * City and the next and add it to the totalDistance
    */
   for (int i = 0; i < route.size() - 1; i++) {




   }

   /* store totalDistance as this candidate solution's fitness */
   this.fitness = totalDistance;
}
```

```java
/**
 * Calculates the total distance of the whole route and stores it as this
 * candidate solution's fitness
 */
private void calculateFitness() {

   /* initialize total distance */
   double totalDistance = 0;

   /*
    * For all Cities in the route (except the last one) get the distance between this
    * City and the next and add it to the totalDistance
    */
   for (int i = 0; i < route.size() - 1; i++) {

      City city = route.get(i);
      City nextCity = route.get(i + 1);
      totalDistance += city.calculateDistance(nextCity);
   }

   /* store totalDistance as this candidate solution's fitness */
   this.fitness = totalDistance;
}
```

bas.knopper@jdriven.com @BWKnopper github.com/bknopper

EA Algorithm (Pseudocode)

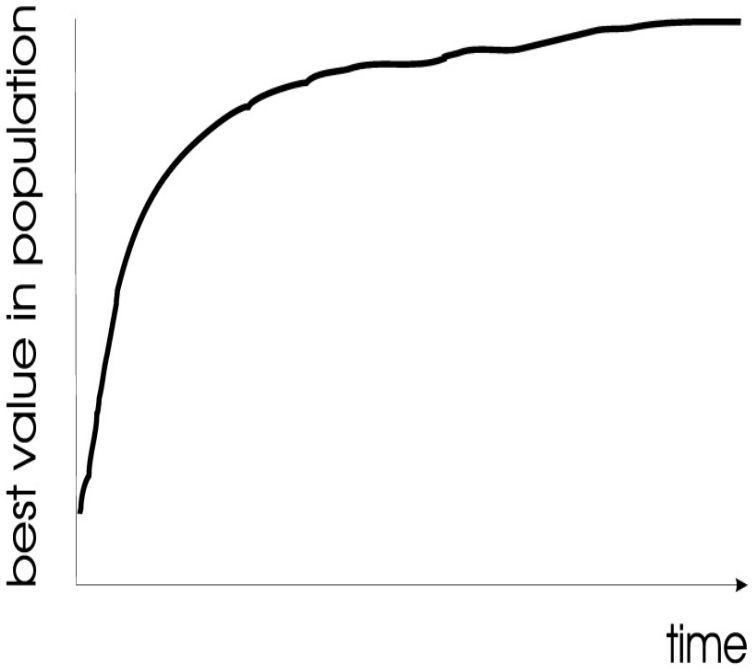
INITIALISE population with random candidate solutions;
EVALUATE each candidate;

EA Algorithm (Pseudocode)

```
INITIALISE population with random candidate solutions;
EVALUATE each candidate;
WHILE ( TERMINATION CONDITION is not satisfied ) {



}
```

Termination Condition

- EA's are stochastic
  - May never find optimum



Figures from "Introduction to Evolutionary Computing" by A.E. Eiben & J.E. Smith (Springer)

Termination Condition

- Combination
- Sure to terminate
  - Time
  - Max number of runs (generations)
- Goal
  - Fitness threshold
  - Fitness improvement stagnation

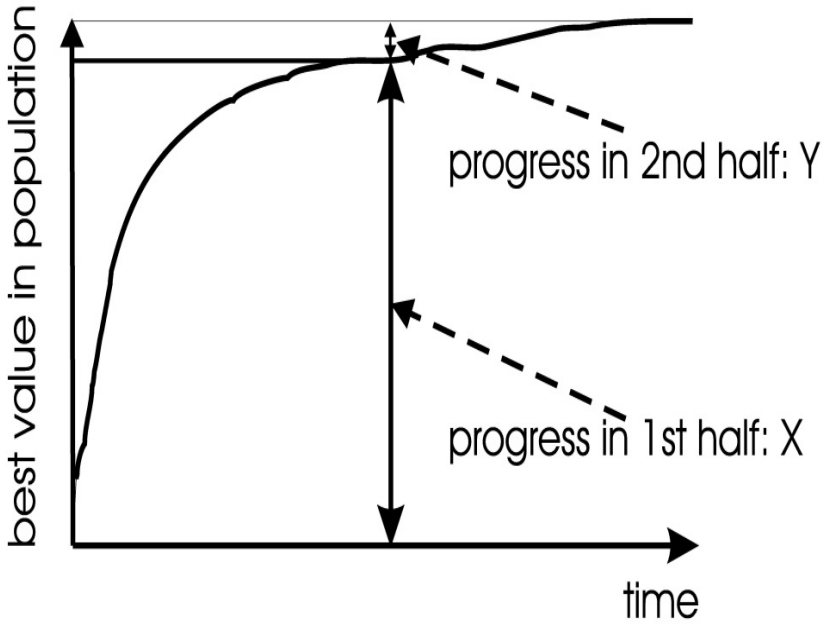


Figure from "Introduction to Evolutionary Computing" by A.E. Eiben & J.E. Smith (Springer)

EA Algorithm (Pseudocode)

```
INITIALISE population with random candidate solutions;
EVALUATE each candidate;
WHILE ( TERMINATION CONDITION is not satisfied ) {



}
```

EA Algorithm (Pseudocode)

```
INITIALISE population with random candidate solutions;
EVALUATE each candidate;
WHILE ( TERMINATION CONDITION is not satisfied ) {
   1 SELECT parents;



}
```

JDriven

Parent Selection

- Where *x* is the number of parents:
  - Pick *x* best
  - Random *x*
  - Best *x* out of random *y*

- In our example:
  - Best x out of random y

```
private List<CandidateSolution> parentSelection() {




}
```

```
private List<CandidateSolution> parentSelection() {

    List<CandidateSolution> tempPopulation = new ArrayList<>(population);
    List<CandidateSolution> randomCandidates = new ArrayList<>();




}
```


```
private List<CandidateSolution> parentSelection() {

    List<CandidateSolution> tempPopulation = new ArrayList<>(population);
    List<CandidateSolution> randomCandidates = new ArrayList<>();

    /* create parent pool */
    for(int i = 0; i < parentPoolSize; i++)
    {









    }



}
```

bas.knopper@jdriven.com @BWKnopper github.com/bknopper


```
private List<CandidateSolution> parentSelection() {

    List<CandidateSolution> tempPopulation = new ArrayList<>(population);
    List<CandidateSolution> randomCandidates = new ArrayList<>();

    /* create parent pool */
    for(int i = 0; i < parentPoolSize; i++)
    {
       /* select a random candidate solution from the temp population */
       int randomlySelectedIndex = random.nextInt(tempPopulation.size());
       CandidateSolution randomSelection = tempPopulation.get(randomlySelectedIndex);
       randomCandidates.add(randomSelection);



    }




}
```

```
private List<CandidateSolution> parentSelection() {

    List<CandidateSolution> tempPopulation = new ArrayList<>(population);
    List<CandidateSolution> randomCandidates = new ArrayList<>();

    /* create parent pool */
    for(int i = 0; i < parentPoolSize; i++)
    {
       /* select a random candidate solution from the temp population */
       int randomlySelectedIndex = random.nextInt(tempPopulation.size());
       CandidateSolution randomSelection = tempPopulation.get(randomlySelectedIndex);
       randomCandidates.add(randomSelection);

       /* delete the candidate from the temp population, so we can't pick it again */
       tempPopulation.remove(randomlySelectedIndex);
    }



}
```

```
private List<CandidateSolution> parentSelection() {

    List<CandidateSolution> tempPopulation = new ArrayList<>(population);
    List<CandidateSolution> randomCandidates = new ArrayList<>();

    /* create parent pool */
    for(int i = 0; i < parentPoolSize; i++)
    {
       /* select a random candidate solution from the temp population */
       int randomlySelectedIndex = random.nextInt(tempPopulation.size());
       CandidateSolution randomSelection = tempPopulation.get(randomlySelectedIndex);
       randomCandidates.add(randomSelection);

       /* delete the candidate from the temp population, so we can't pick it again */
       tempPopulation.remove(randomlySelectedIndex);
    }

    /* Sort the population so that the best candidates are up front */
    Collections.sort(randomCandidates);



}
```

```
private List<CandidateSolution> parentSelection() {

    List<CandidateSolution> tempPopulation = new ArrayList<>(population);
    List<CandidateSolution> randomCandidates = new ArrayList<>();

    /* create parent pool */
    for(int i = 0; i < parentPoolSize; i++)
    {
       /* select a random candidate solution from the temp population */
       int randomlySelectedIndex = random.nextInt(tempPopulation.size());
       CandidateSolution randomSelection = tempPopulation.get(randomlySelectedIndex);
       randomCandidates.add(randomSelection);

       /* delete the candidate from the temp population, so we can't pick it again */
       tempPopulation.remove(randomlySelectedIndex);
    }

    /* Sort the population so that the best candidates are up front */
    Collections.sort(randomCandidates);

    /* return a list with size parentSelectionSize with the best CandidateSolutions   */
    return randomCandidates.subList(0, parentSelectionSize);
}
```

bas.knopper@jdriven.com @BWKnopper github.com/bknopper

EA Algorithm (Pseudocode)

```
INITIALISE population with random candidate solutions;
EVALUATE each candidate;
WHILE ( TERMINATION CONDITION is not satisfied ) {
   1 SELECT parents;



}
```

EA Algorithm (Pseudocode)

```
INITIALISE population with random candidate solutions;
EVALUATE each candidate;
WHILE ( TERMINATION CONDITION is not satisfied ) {
   1 SELECT parents;
   2 RECOMBINE pairs of parents;



}
```

Recombination

- In our example:
  - half-half does not work
  - “Cut-and-crossfill”

bas.knopper@jdriven.com @BWKnopper github.com/bknopper

```
public List<CandidateSolution> recombine(CandidateSolution otherParent) {

}
```

```
public List<CandidateSolution> recombine(CandidateSolution otherParent) {
    /* get routes of both parents */
    List<City> parentRoute1 = getRoute();
    List<City> parentRoute2 = otherParent.getRoute();









}
```

```
public List<CandidateSolution> recombine(CandidateSolution otherParent) {
    /* get routes of both parents */
    List<City> parentRoute1 = getRoute();
    List<City> parentRoute2 = otherParent.getRoute();
    /* initialize the routes for the children */
    List<City> childRoute1 = new ArrayList<City>();
    List<City> childRoute2 = new ArrayList<City>();









}
```

```
public List<CandidateSolution> recombine(CandidateSolution otherParent) {
    /* get routes of both parents */
    List<City> parentRoute1 = getRoute();
    List<City> parentRoute2 = otherParent.getRoute();
    /* initialize the routes for the children */
    List<City> childRoute1 = new ArrayList<City>();
    List<City> childRoute2 = new ArrayList<City>();

    /* randomize cutIndex for "cross-and-fill point" */
    int cutIndex = new Random().nextInt(parentRoute1.size());









}
```

```
public List<CandidateSolution> recombine(CandidateSolution otherParent) {
    /* get routes of both parents */
    List<City> parentRoute1 = getRoute();
    List<City> parentRoute2 = otherParent.getRoute();
    /* initialize the routes for the children */
    List<City> childRoute1 = new ArrayList<City>();
    List<City> childRoute2 = new ArrayList<City>();

    /* randomize cutIndex for "cross-and-fill point" */
    int cutIndex = new Random().nextInt(parentRoute1.size());

    /* copy the first part of the parents cut into the children */
    childRoute1.addAll(parentRoute1.subList(0,  cutIndex));
    childRoute2.addAll(parentRoute2.subList(0,  cutIndex));









}
```

```
public List<CandidateSolution> recombine(CandidateSolution otherParent) {
    /* get routes of both parents */
    List<City> parentRoute1 = getRoute();
    List<City> parentRoute2 = otherParent.getRoute();
    /* initialize the routes for the children */
    List<City> childRoute1 = new ArrayList<City>();
    List<City> childRoute2 = new ArrayList<City>();

    /* randomize cutIndex for "cross-and-fill point" */
    int cutIndex = new Random().nextInt(parentRoute1.size());

    /* copy the first part of the parents cut into the children */
    childRoute1.addAll(parentRoute1.subList(0,  cutIndex));
    childRoute2.addAll(parentRoute2.subList(0,  cutIndex));

    /* perform crossfill for both children */
    crossFill(childRoute1, parentRoute2, cutIndex);
    crossFill(childRoute2, parentRoute1, cutIndex);



}
```

JDriven

```java
public List<CandidateSolution> recombine(CandidateSolution otherParent) {
    /* get routes of both parents */
    List<City> parentRoute1 = getRoute();
    List<City> parentRoute2 = otherParent.getRoute();
    /* initialize the routes for the children */
    List<City> childRoute1 = new ArrayList<City>();
    List<City> childRoute2 = new ArrayList<City>();

    /* randomize cutIndex for "cross-and-fill point" */
    int cutIndex = new Random().nextInt(parentRoute1.size());

    /* copy the first part of the parents cut into the children */
    childRoute1.addAll(parentRoute1.subList(0, cutIndex));
    childRoute2.addAll(parentRoute2.subList(0, cutIndex));

    /* perform crossfill for both children */
    crossFill(childRoute1, parentRoute2, cutIndex);
    crossFill(childRoute2, parentRoute1, cutIndex);

    /* create new children using the new children routes */
    CandidateSolution child1 = new CandidateSolution(childRoute1);
    CandidateSolution child2 = new CandidateSolution(childRoute2);


}
```

JDriven

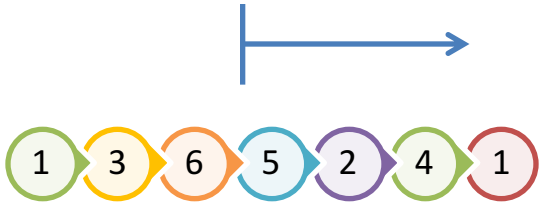
```
public List<CandidateSolution> recombine(CandidateSolution otherParent) {
    /* get routes of both parents */
    List<City> parentRoute1 = getRoute();
    List<City> parentRoute2 = otherParent.getRoute();
    /* initialize the routes for the children */
    List<City> childRoute1 = new ArrayList<City>();
    List<City> childRoute2 = new ArrayList<City>();

    /* randomize cutIndex for "cross-and-fill point" */
    int cutIndex = new Random().nextInt(parentRoute1.size());

    /* copy the first part of the parents cut into the children */
    childRoute1.addAll(parentRoute1.subList(0,  cutIndex));
    childRoute2.addAll(parentRoute2.subList(0,  cutIndex));

    /* perform crossfill for both children */
    crossFill(childRoute1, parentRoute2, cutIndex);
    crossFill(childRoute2, parentRoute1, cutIndex);

    /* create new children using the new children routes */
    CandidateSolution child1 = new CandidateSolution(childRoute1);
    CandidateSolution child2 = new CandidateSolution(childRoute2);

    /* put the children in a list and return it (omitted for layout reasons) */
}
```

JDriven

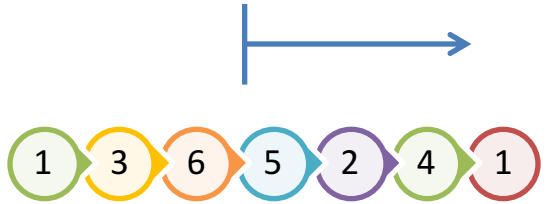
bas.knopper@jdriven.com @BWKnopper github.com/bknopper

```
/**
 * Check the rest of the route in the crossing parent and add the cities
 * that are not yet in the child (in the order of the route of the crossing
 * parent)
 */
private void crossFill(List<City> childRoute, List<City> parentRoute, int cutIndex) {









}
```

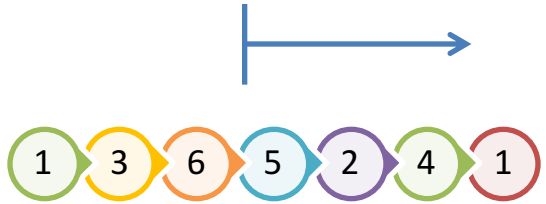
```
/**
 * Check the rest of the route in the crossing parent and add the cities
 * that are not yet in the child (in the order of the route of the crossing
 * parent)
 */
private void crossFill(List<City> childRoute, List<City> parentRoute, int cutIndex) {

    /* traverse the parent route from the cut index on and add every city not yet in the child to the child */
    for (int i = cutIndex; i < parentRoute.size(); i++) {



    }



}
```



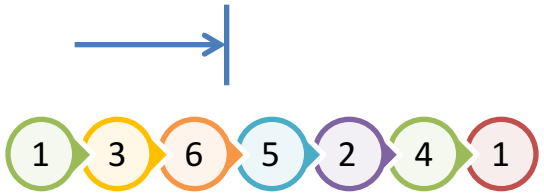
```java
/**
 * Check the rest of the route in the crossing parent and add the cities
 * that are not yet in the child (in the order of the route of the crossing
 * parent)
 */
private void crossFill(List<City> childRoute, List<City> parentRoute, int cutIndex) {

   /* traverse the parent route from the cut index on and add every city not yet in the child to the child */
   for (int i = cutIndex; i < parentRoute.size(); i++) {
      City nextCityOnRoute = parentRoute.get(i);

      if (!childRoute.contains(nextCityOnRoute)) {
         childRoute.add(nextCityOnRoute);
      }
   }

}
```



```java
/**
 * Check the rest of the route in the crossing parent and add the cities
 * that are not yet in the child (in the order of the route of the crossing
 * parent)
 */
private void crossFill(List<City> childRoute, List<City> parentRoute, int cutIndex) {

   /* traverse the parent route from the cut index on and add every city not yet in the child to the child */
   for (int i = cutIndex; i < parentRoute.size(); i++) {
      City nextCityOnRoute = parentRoute.get(i);

      if (!childRoute.contains(nextCityOnRoute)) {
         childRoute.add(nextCityOnRoute);
      }
   }
```




```java
   /* traverse the parent route from the start of the route and add every city not yet in the child to the child */
   for (int i = 0; i < cutIndex; i++) {
      City nextCityOnRoute = parentRoute.get(i);

      if (!childRoute.contains(nextCityOnRoute)) {
         childRoute.add(nextCityOnRoute);
      }
   }
}
```



EA Algorithm (Pseudocode)

```
INITIALISE population with random candidate solutions;
EVALUATE each candidate;
WHILE ( TERMINATION CONDITION is not satisfied ) {
   1 SELECT parents;
   2 RECOMBINE pairs of parents;



}
```

EA Algorithm (Pseudocode)

```
INITIALISE population with random candidate solutions;
EVALUATE each candidate;
WHILE ( TERMINATION CONDITION is not satisfied ) {
   1 SELECT parents;
   2 RECOMBINE pairs of parents;
   3 MUTATE the resulting offspring;


}
```

Mutation

- Change of nr won't work
  - Infeasible candidate solution
- Swap to the rescue!

```java
/**
 * Mutates the current individual by swapping two random cities in its
 * route.
 */
public void mutate() {




}
```

```java
/**
 * Mutates the current individual by swapping two random cities in its
 * route.
 */
public void mutate() {

   Random random = new Random();

   /* randomly select two indices in the route */
   int indexFirstCity = random.nextInt(route.size());
   int indexSecondCity = random.nextInt(route.size());

}
```

```java
/**
 * Mutates the current individual by swapping two random cities in its
 * route.
 */
public void mutate() {

   Random random = new Random();

   /* randomly select two indices in the route */
   int indexFirstCity = random.nextInt(route.size());
   int indexSecondCity = random.nextInt(route.size());

   /* Make sure they are different */
   while (indexFirstCity == indexSecondCity) {
      indexSecondCity = random.nextInt(route.size());
   }



}
```

```java
/**
 * Mutates the current individual by swapping two random cities in its
 * route.
 */
public void mutate() {

   Random random = new Random();

   /* randomly select two indices in the route */
   int indexFirstCity = random.nextInt(route.size());
   int indexSecondCity = random.nextInt(route.size());

   /* Make sure they are different */
   while (indexFirstCity == indexSecondCity) {
      indexSecondCity = random.nextInt(route.size());
   }

   /* retrieve the Cities on the given indices */
   City firstCity = route.get(indexFirstCity);
   City secondCity = route.get(indexSecondCity);



}
```

```java
/**
 * Mutates the current individual by swapping two random cities in its
 * route.
 */
public void mutate() {

   Random random = new Random();

   /* randomly select two indices in the route */
   int indexFirstCity = random.nextInt(route.size());
   int indexSecondCity = random.nextInt(route.size());

   /* Make sure they are different */
   while (indexFirstCity == indexSecondCity) {
      indexSecondCity = random.nextInt(route.size());
   }

   /* retrieve the Cities on the given indices */
   City firstCity = route.get(indexFirstCity);
   City secondCity = route.get(indexSecondCity);


   /* Changer! */
   route.set(indexFirstCity, secondCity);
   route.set(indexSecondCity, firstCity);

}
```

EA Algorithm (Pseudocode)

```
INITIALISE population with random candidate solutions;
EVALUATE each candidate;
WHILE ( TERMINATION CONDITION is not satisfied ) {
   1 SELECT parents;
   2 RECOMBINE pairs of parents;
   3 MUTATE the resulting offspring;


}
```

EA Algorithm (Pseudocode)

```
INITIALISE population with random candidate solutions;
EVALUATE each candidate;
WHILE ( TERMINATION CONDITION is not satisfied ) {
   1 SELECT parents;
   2 RECOMBINE pairs of parents;
   3 MUTATE the resulting offspring;
   4 EVALUATE new candidates;

}
```

EA Algorithm (Pseudocode)

```
INITIALISE population with random candidate solutions;
EVALUATE each candidate;
WHILE ( TERMINATION CONDITION is not satisfied ) {
   1 SELECT parents;
   2 RECOMBINE pairs of parents;
   3 MUTATE the resulting offspring;
   4 EVALUATE new candidates;
   5 SELECT individuals for the next generation;
}
```

JDriven

Survivor Selection

- Replacement Strategy
- Do nothing
- Replace Worst

```
/**
 * Selects the survivors by removing the worst candidate
 * solutions from the list, so we have the original
 * population size again
 */
private void selectSurvivors() {



}
```

```
/**
 * Selects the survivors by removing the worst candidate
 * solutions from the list, so we have the original
 * population size again
 */
private void selectSurvivors() {

  Collections.sort(population);

}
```

JDriven

```
/**
 * Selects the survivors by removing the worst candidate
 * solutions from the list, so we have the original
 * population size again
 */
private void selectSurvivors() {

  Collections.sort(population);
  population = population.subList(0, populationSize);
}
```

EA Algorithm (Pseudocode)

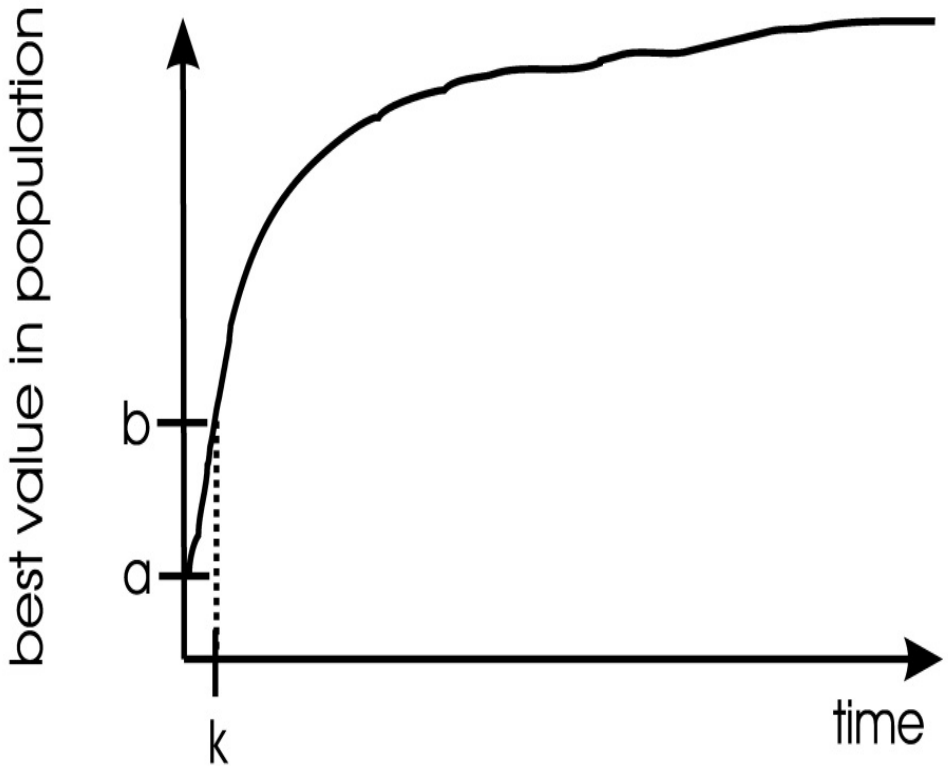
```
INITIALISE population with random candidate solutions;
EVALUATE each candidate;
WHILE ( TERMINATION CONDITION is not satisfied ) {
   1 SELECT parents;
   2 RECOMBINE pairs of parents;
   3 MUTATE the resulting offspring;
   4 EVALUATE new candidates;
   5 SELECT individuals for the next generation;
}
```

JDriven

Tuning…

- Mutation probability
- Population size
- Nr of offspring
- Termination condition (# runs or fitness)
- Parent selection
- Survival selection
- Initialisation
  - Random

EA Behavior



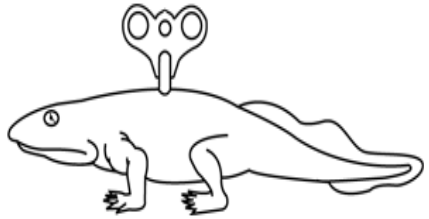
Figures from "Introduction to Evolutionary Computing" by A.E. Eiben & J.E. Smith (Springer)

JDriven

Demo!

- Backend
  - Java code shown
  - Used NASA World Wind to do calculations on the backend
- Frontend
  - AngularJs + Bootstrap + Google Maps ☺
- https://github.com/bknopper/TSPEvolutionaryAlgorithmsDemo.git

Java Frameworks & API's



WATCHMAKER FRAMEWORK

FOR EVOLUTIONARY COMPUTATION

http://watchmaker.uncommons.org

JGAP

JAVA GENETIC ALGORITHMS PACKAGE

http://jgap.sourceforge.net/

bas.knopper@jdriven.com @BWKnopper github.com/bknopper

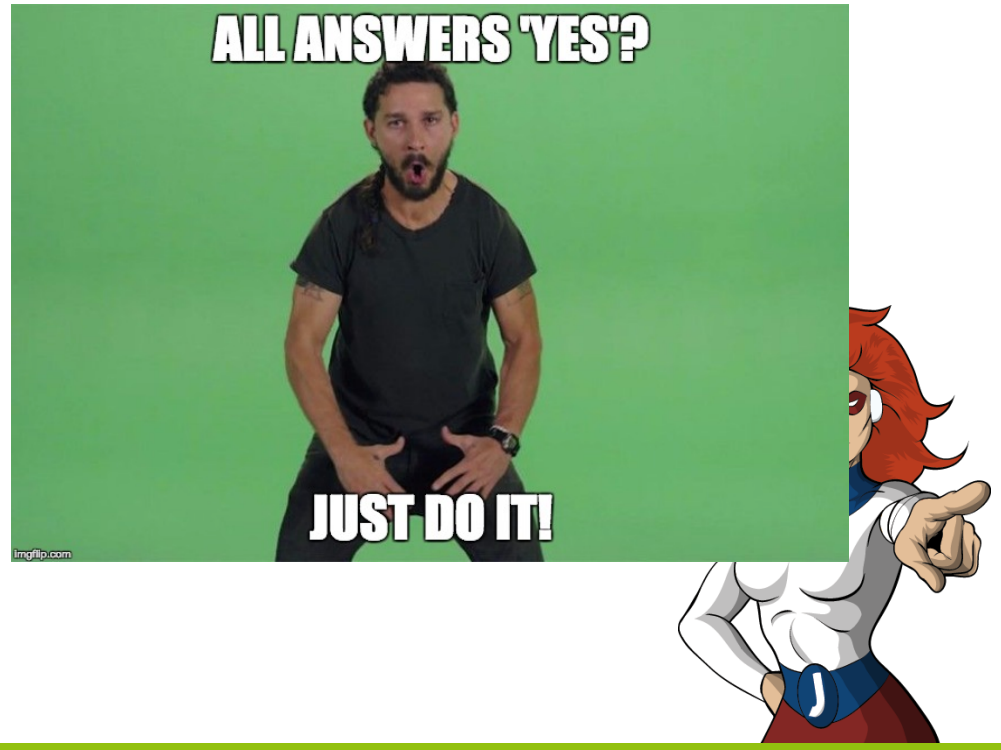
Java Frameworks & API's

- ECJ
  - http://cs.gmu.edu/~eclab/projects/ecj/
- MOEA Framework
  - http://www.moeaframework.org
- JEAF
  - https://github.com/GII/JEAF
- …

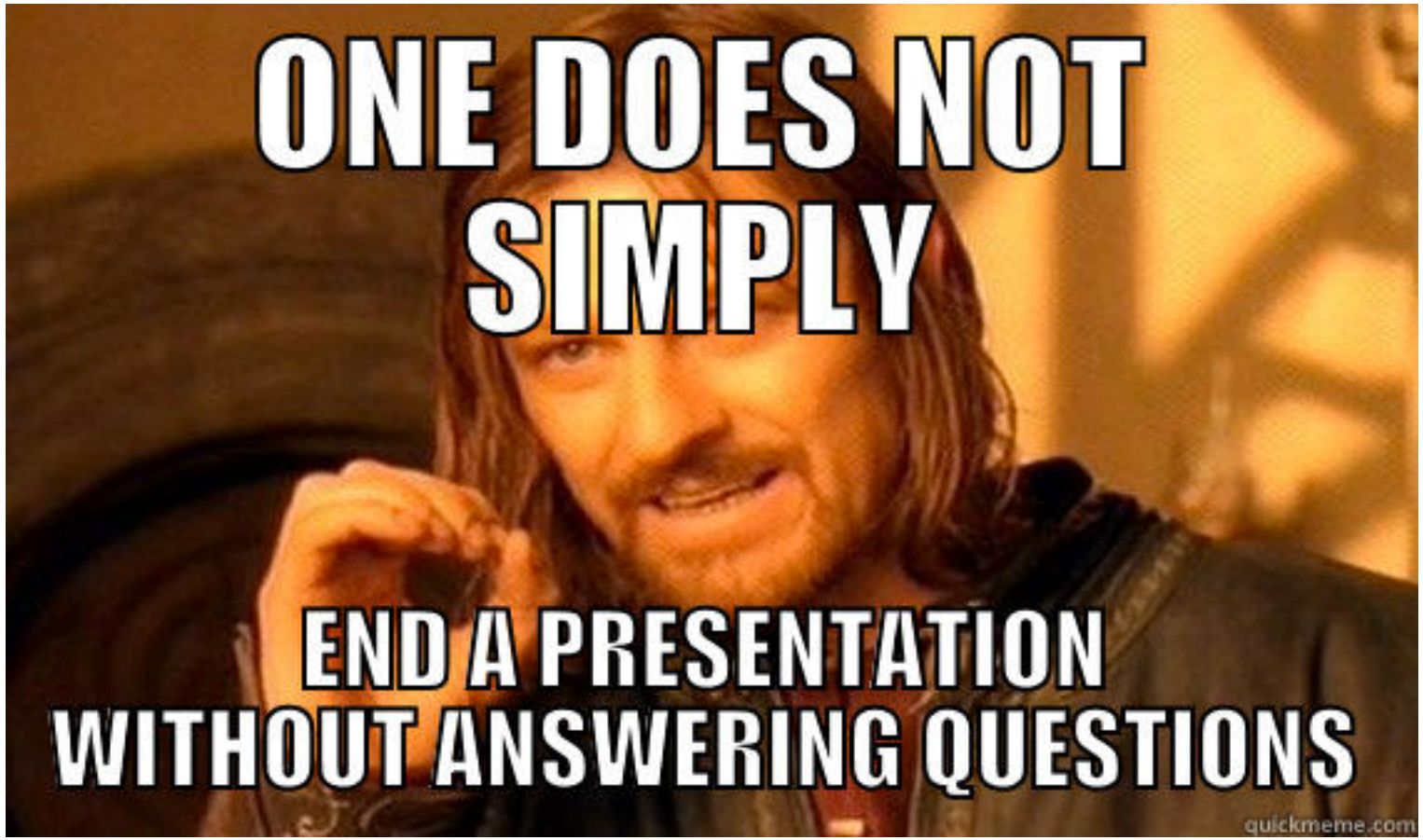
With great power comes great responsibility

- I'm sure I cannot find a solution using a brute-force approach?
  - (within a reasonable amount of time)

- Am I facing an optimization or search problem?
- Can I encode a candidate solution to the problem?
  - Representation possible?
- Can I determine the fitness of a candidate solution?



JDriven



bas.knopper@jdriven.com @BWKnopper github.com/bknopper

JDriven

Additional questions?

- Contact me on @BWKnopper
- Google it!
  - There's lots to find…
    - Papers
    - Demo's
    - Aforementioned Frameworks/API's


THANK
YOU