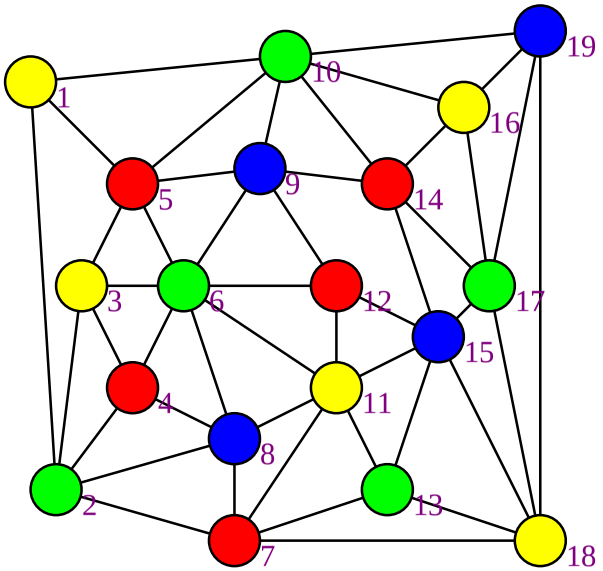
Representing problems

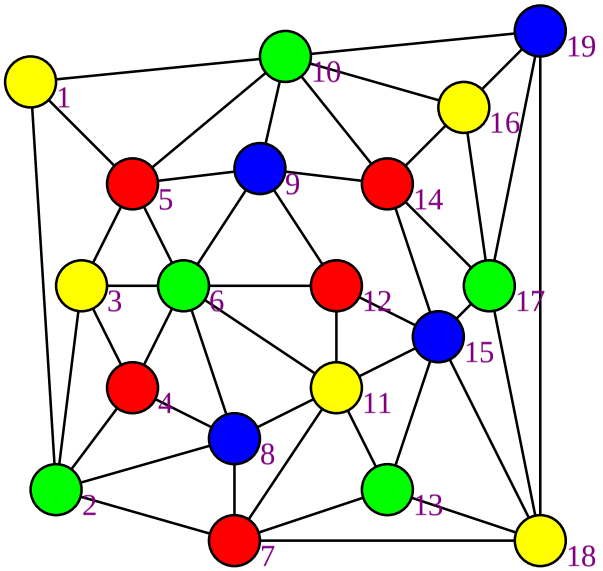
- Many problems have very simple, natural representations as a string
- Graph colouring is an example of this
- We can use standard string based mutation and crossover operators on these natural representation



(Y,G,Y,R,R,G,R,B,B,G,Y,R,G,R,B,Y,G,Y,B)

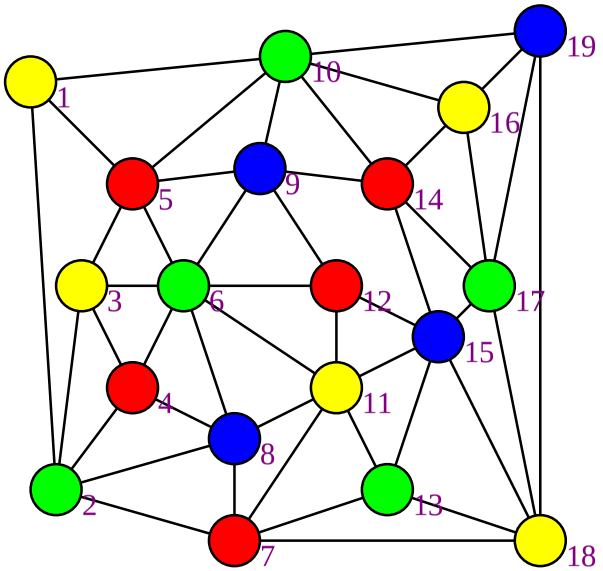
Representing problems

- Many problems have very simple, natural representations as a string
- Graph colouring is an example of this
- We can use standard string based mutation and crossover operators on these natural representation



Representing problems

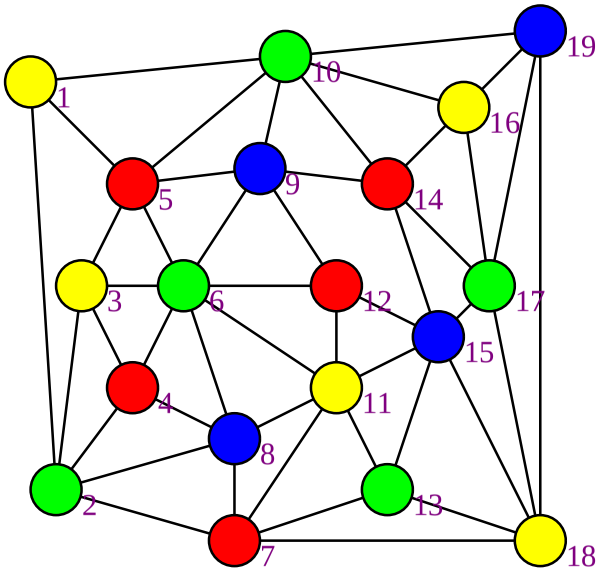
- Many problems have very simple, natural representations as a string
- Graph colouring is an example of this
- We can use standard string based mutation and crossover operators on these natural representation



(Y,G,Y,R,R,G,R,B,B,G,Y,R,G,R,B,Y,G,Y,B)

Representing problems

- Many problems have very simple, natural representations as a string
- Graph colouring is an example of this
- We can use standard string based mutation and crossover operators on these natural representation
- For graph colouring these operators don't work well (c.f. Galinier and Hao's crossover)



(Y,G,Y,R,R,G,R,B,B,G,Y,R,G,R,B,Y,G,Y,B)

TSP and GAs

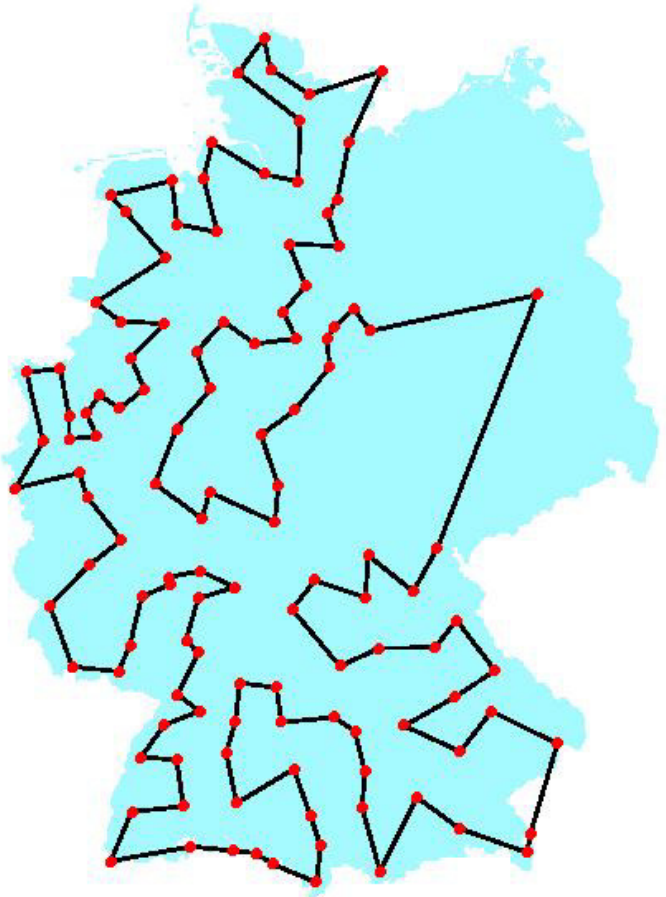
- In TSP there is not a natural string representation

- The most natural representation is a permutation of $(1, 2, 3, \cdots, n)$

- However normal mutation and crossover doesn't work

$$(1, 2, 3, 4) \overset{point\ mutation}{\longrightarrow} (1, 2, 1, 4)$$

$$\begin{matrix} (1, 2, |3, 4) \\ (1, 3, |4, 2) \end{matrix} \overset{single\ point\ crossover}{\longrightarrow} \begin{matrix} (1, 2, |4, 2) \\ (1, 3, |3, 4) \end{matrix}$$



TSP and GAs

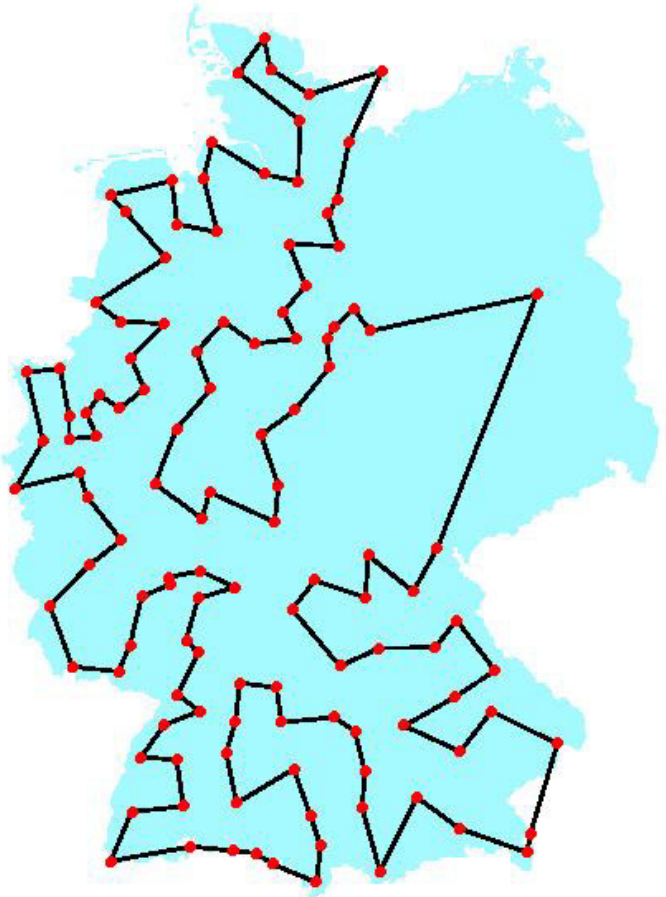
- In TSP there is not a natural string representation

- The most natural representation is a permutation of $(1, 2, 3, \cdots, n)$

- However normal mutation and crossover doesn't work

$$(1, 2, 3, 4) \overset{point\ mutation}{\longrightarrow} (1, 2, 1, 4)$$

$$\begin{matrix} (1, 2, |3, 4) \\ (1, 3, |4, 2) \end{matrix} \overset{single\ point\ crossover}{\longrightarrow} \begin{matrix} (1, 2, |4, 2) \\ (1, 3, |3, 4) \end{matrix}$$

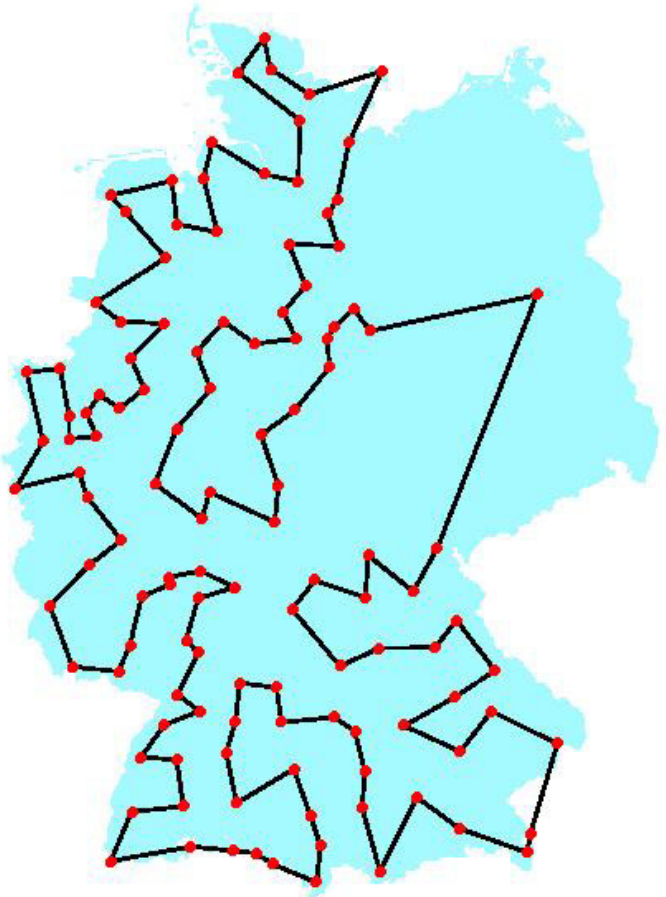


TSP and GAs

- In TSP there is not a natural string representation
- The most natural representation is a permutation of $(1, 2, 3, \cdots, n)$
- However normal mutation and crossover doesn't work

$$(1,2,3,4) \overset{point\ mutation}{\longrightarrow} (1,2,1,4)$$

$$\begin{matrix}(1,2,|3,4) \\ (1,3,|4,2)\end{matrix} \overset{single\ point\ crossover}{\longrightarrow} \begin{matrix}(1,2,|4,2) \\ (1,3,|3,4)\end{matrix}$$

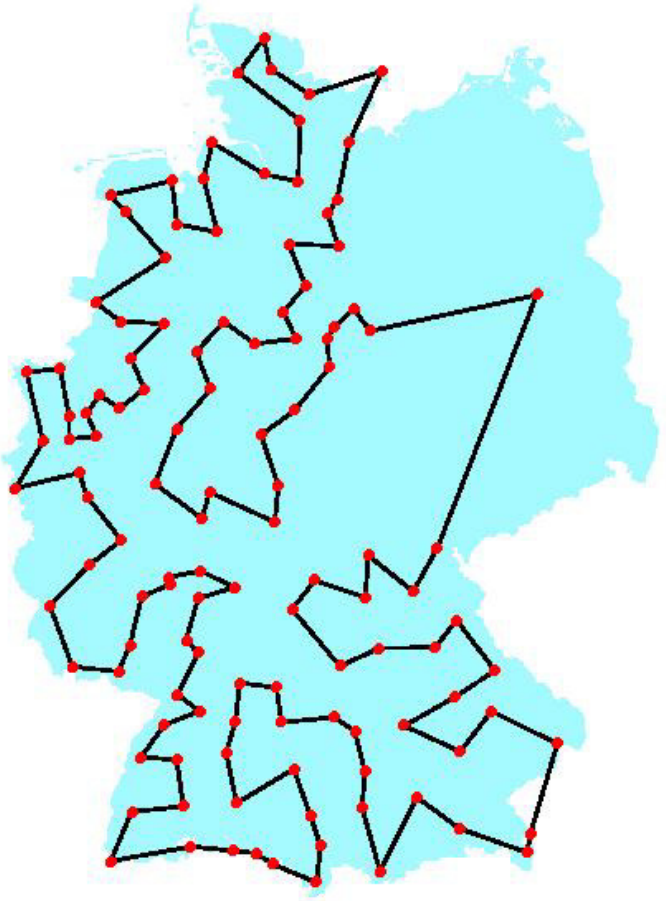


TSP and GAs

- In TSP there is not a natural string representation
- The most natural representation is a permutation of $(1, 2, 3, \cdots, n)$
- However normal mutation and crossover doesn't work

$$(1, 2, 3, 4) \overset{point\ mutation}{\longrightarrow} (1, 2, 1, 4)$$

$$(1, 2, |3, 4) \quad (1, 3, |4, 2) \quad \overset{single\ point\ crossover}{\longrightarrow} \quad (1, 2, |4, 2) \quad (1, 3, |3, 4)$$



TSP and GAs

- In TSP there is not a natural string representation

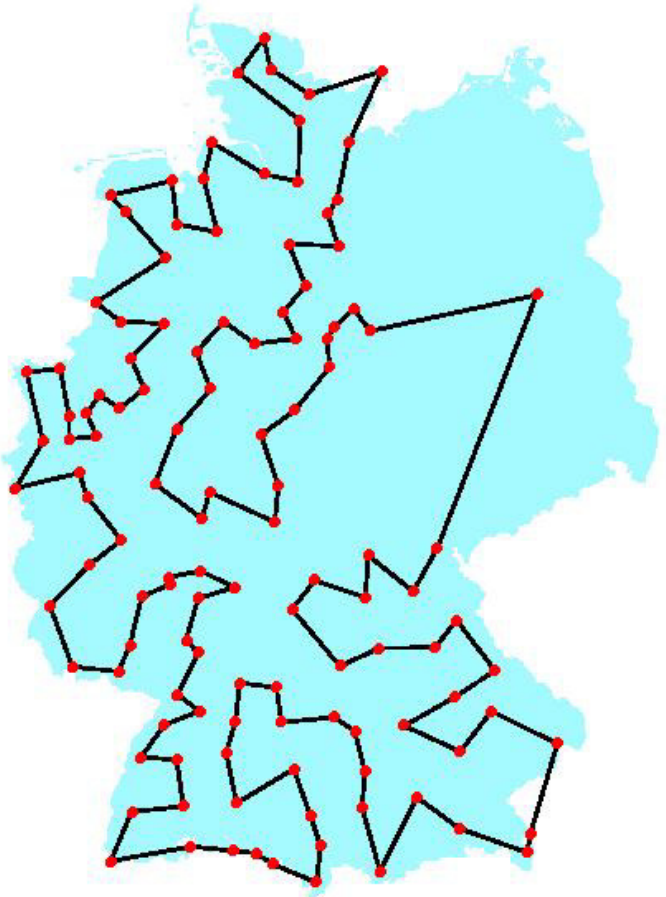
- The most natural representation is a permutation of $(1, 2, 3, \cdots, n)$

- However normal mutation and crossover doesn't work

$$(1,2,3,4) \overset{point\ mutation}{\longrightarrow} (1,2,1,4)$$

$$\begin{matrix}(1,2,|3,4) \\ (1,3,|4,2)\end{matrix} \overset{single\ point\ crossover}{\longrightarrow} \begin{matrix}(1,2,|4,2) \\ (1,3,|3,4)\end{matrix}$$

not legal tours!



Changing the Representation

- A number of different representations for the TSP have been proposed so that point mutation and one or two-point crossover work
- Unfortunately, such representations don't lead to good GAs
- More commonly a crossover is introduced which takes half the tour from one parent and takes as much of the tour as possible from the second parent but **repairs** the tour where necessary
- These tours are again not terribly successful

Changing the Representation

- A number of different representations for the TSP have been proposed so that point mutation and one or two-point crossover work
- Unfortunately, such representations don't lead to good GAs
- More commonly a crossover is introduced which takes half the tour from one parent and takes as much of the tour as possible from the second parent but **repairs** the tour where necessary
- These tours are again not terribly successful

Changing the Representation

- A number of different representations for the TSP have been proposed so that point mutation and one or two-point crossover work
- Unfortunately, such representations don't lead to good GAs
- More commonly a crossover is introduced which takes half the tour from one parent and takes as much of the tour as possible from the second parent but **repairs** the tour where necessary
- These tours are again not terribly successful

Changing the Representation

- A number of different representations for the TSP have been proposed so that point mutation and one or two-point crossover work
- Unfortunately, such representations don't lead to good GAs
- More commonly a crossover is introduced which takes half the tour from one parent and takes as much of the tour as possible from the second parent but **repairs** the tour where necessary
- These tours are again not terribly successful

Back to Basics

- These attempts miss the main point of why GAs (sometimes) work
- Instead they try to make traditional string based operators work on a permutation based problem
- GAs are a heuristic search mechanism which work by exploring points in the search space that have similar cost to previous points
- For this to work you need to make the GA operators (mutation and crossover) produce alterations which minimises the change in cost or at least considers cost

Back to Basics

- These attempts miss the main point of why GAs (sometimes) work
- Instead they try to make traditional string based operators work on a permutation based problem
- GAs are a heuristic search mechanism which work by exploring points in the search space that have similar cost to previous points
- For this to work you need to make the GA operators (mutation and crossover) produce alterations which minimises the change in cost or at least considers cost

Back to Basics

- These attempts miss the main point of why GAs (sometimes) work
- Instead they try to make traditional string based operators work on a permutation based problem
- GAs are a heuristic search mechanism which work by exploring points in the search space that have similar cost to previous points
- For this to work you need to make the GA operators (mutation and crossover) produce alterations which minimises the change in cost or at least considers cost

Back to Basics

- These attempts miss the main point of why GAs (sometimes) work
- Instead they try to make traditional string based operators work on a permutation based problem
- GAs are a heuristic search mechanism which work by exploring points in the search space that have similar cost to previous points
- For this to work you need to make the GA operators (mutation and crossover) produce alterations which minimises the change in cost or at least considers cost

Edge-Based Operators

- Since it is the edges which determines the cost it seems sensible to build operators to preserve as many edges as possible
- An example of an operator which does this is *edge recombination*
- These are slight variants on this strategy, but they all try to construct a child using as many edges of the parents as possible

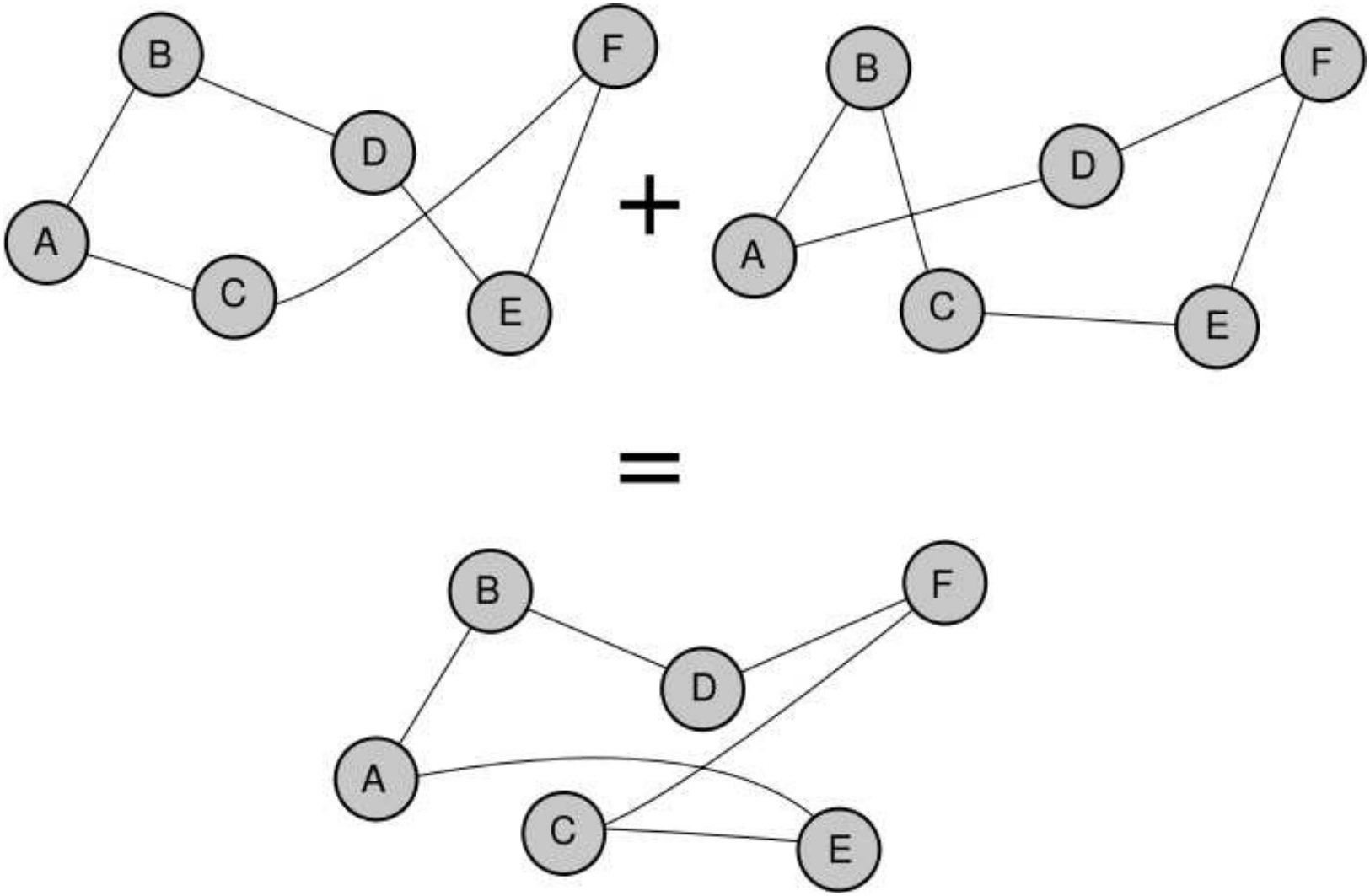
Edge-Based Operators

- Since it is the edges which determines the cost it seems sensible to build operators to preserve as many edges as possible
- An example of an operator which does this is *edge recombination*
- These are slight variants on this strategy, but they all try to construct a child using as many edges of the parents as possible

Edge-Based Operators

- Since it is the edges which determines the cost it seems sensible to build operators to preserve as many edges as possible
- An example of an operator which does this is *edge recombination*
- These are slight variants on this strategy, but they all try to construct a child using as many edges of the parents as possible

Edge-Recombination



Mutation

- Mutation is also non-trivial in TSP
- Point mutations make no sense
- Swapping two cities

$$(1,2,3,4,5,6) \overset{swap\ mutation}{\longrightarrow} (1,5,3,4,2,6)$$

  disrupts four edges 1–5, 5–3, 4–2 and 2–6

- This isn't a minimal change

Mutation

- Mutation is also non-trivial in TSP
- Point mutations make no sense
- Swapping two cities

$$(1,2,3,4,5,6) \overset{swap\ mutation}{\longrightarrow} (1,5,3,4,2,6)$$

  disrupts four edges 1–5, 5–3, 4–2 and 2–6

- This isn't a minimal change

Mutation

- Mutation is also non-trivial in TSP
- Point mutations make no sense
- Swapping two cities

$$(1,2,3,4,5,6) \overset{swap\ mutation}{\longrightarrow} (1,5,3,4,2,6)$$

disrupts four edges 1–5, 5–3, 4–2 and 2–6

- This isn't a minimal change

Mutation

- Mutation is also non-trivial in TSP
- Point mutations make no sense
- Swapping two cities

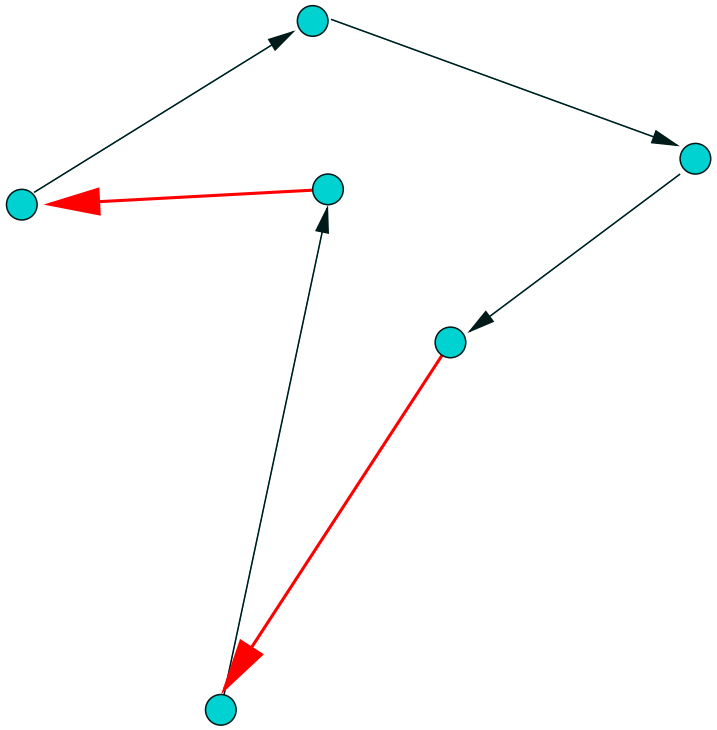
$$(1,2,3,4,5,6) \overset{swap\ mutation}{\longrightarrow} (1,5,3,4,2,6)$$

disrupts four edges 1–5, 5–3, 4–2 and 2–6

- This isn't a minimal change

2-Opt

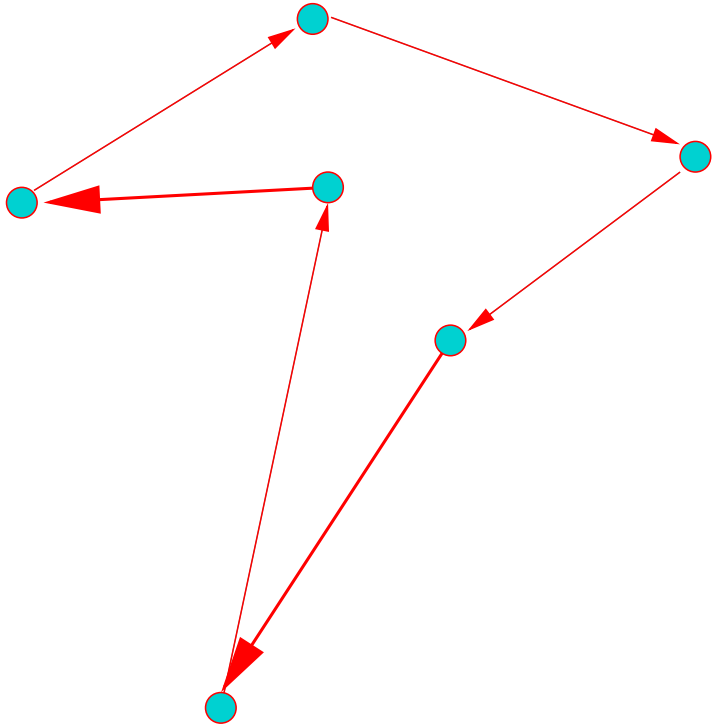
- The least disruptive mutation is known as **2-Opt**



- It is easy to prove for a 2-D Euclidean TSP that if any two lines on a tour cross then a 2-Opt move will uncross the tour and reduce the length

2-Opt

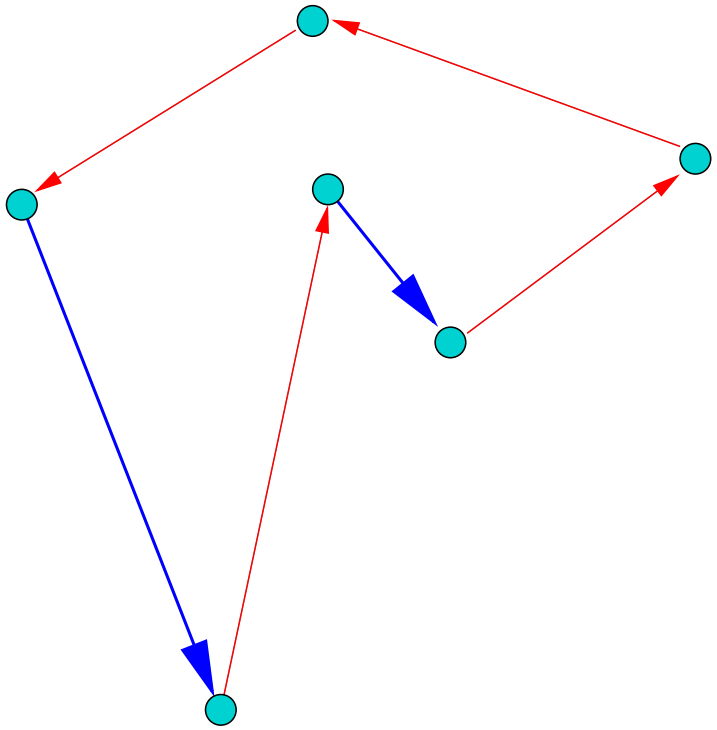
- The least disruptive mutation is known as **2-Opt**



- It is easy to prove for a 2-D Euclidean TSP that if any two lines on a tour cross then a 2-Opt move will uncross the tour and reduce the length

2-Opt

- The least disruptive mutation is known as **2-Opt**

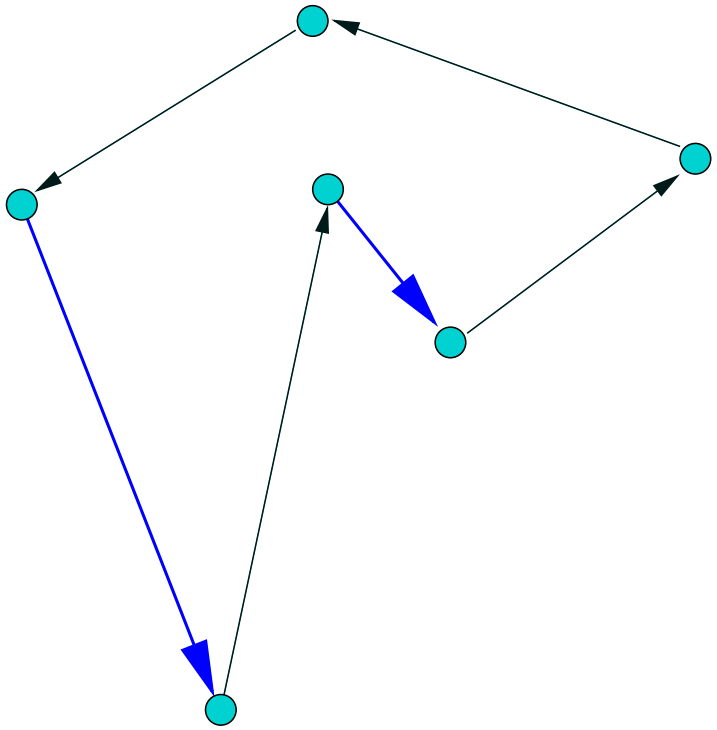


2-Opt neighbours

- It is easy to prove for a 2-D Euclidean TSP that if any two lines on a tour cross then a 2-Opt move will uncross the tour and reduce the length

2-Opt

- The least disruptive mutation is known as **2-Opt**

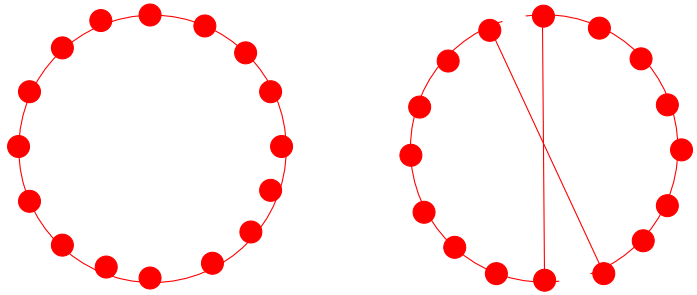


2-Opt neighbours

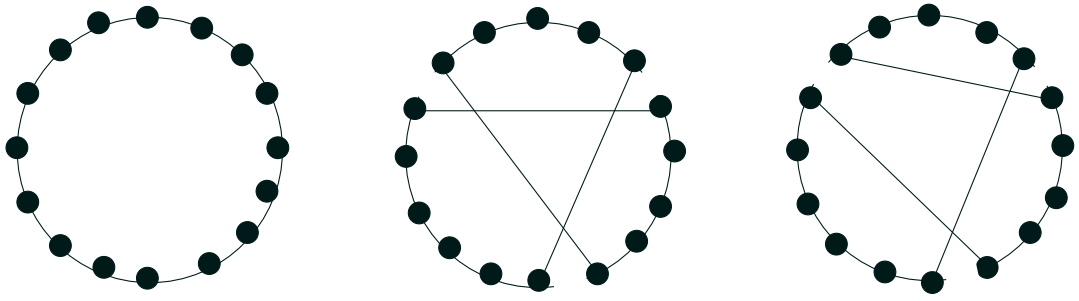
- It is easy to prove for a 2-D Euclidean TSP that if any two lines on a tour cross then a 2-Opt move will uncross the tour and reduce the length

3-Opt

- 2-Opts are not sufficiently powerful to escape from all local optima

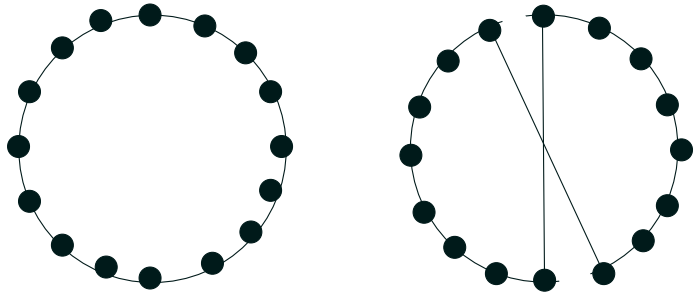


- A more powerful, but typically more disruptive move is 3-Opt

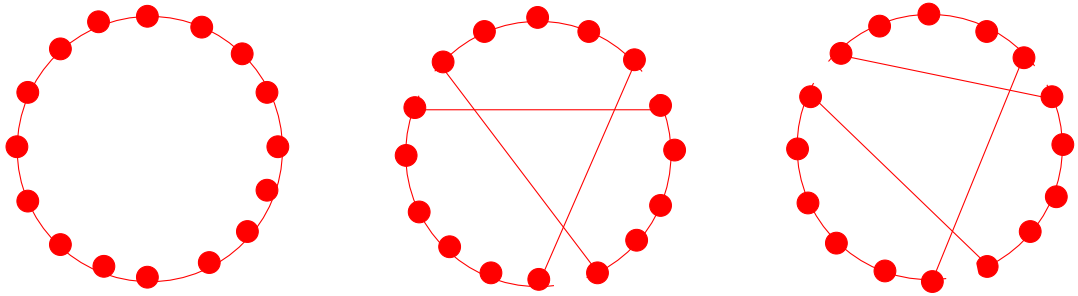


3-Opt

- 2-Opts are not sufficiently powerful to escape from all local optima



- A more powerful, but typically more disruptive move is 3-Opt



Neighbourhood Search

- Using 2-Opt and 3-Opt provides a fast neighbourhood search
- Neighbourhood search is fast because it is relatively cheap to recompute the cost after performing a 2-Opt or 3-Opt move
- The classic search method for TSP is the $Lin\text{-}Kernighan\ method$ which uses 2-Opts, 3-Opts and a specialised 4-Opt which combines two 2-Opts
- The best heuristic to date for finding good solutions for TSP is $Iterated\ Lin\text{-}Kernighan$
- For any GA to be competitive with the state-of-the-art it has to include a fast neighbourhood search