Hill-Climbing Algorithm:
let's go for a walk before finding the optimum

Leticia Hernando, Alexander Mendiburu and Jose A. Lozano

Intelligent Systems Group
University of the Basque Country UPV/EHU, Spain

July, 11th 2018
2018 IEEE Congress on Evolutionary Computation

UPV - EHU

Objectives

- Analysis of the solutions found in the attraction basins: distance to the local optimum vs. number of steps of the algorithm.
- The paths defined by a hill-climbing algorithm do not monotonically reduce the distance to the local optimum.
- Visual examples of the paths built by the algorithm.

Objectives

- Analysis of the solutions found in the attraction basins: distance to the local optimum vs. number of steps of the algorithm.
- The paths defined by a hill-climbing algorithm do not monotonically reduce the distance to the local optimum.
- Visual examples of the paths built by the algorithm.

UPV - EHU

Objectives

- Analysis of the solutions found in the attraction basins: distance to the local optimum vs. number of steps of the algorithm.
- The paths defined by a hill-climbing algorithm do not monotonically reduce the distance to the local optimum.
- Visual examples of the paths built by the algorithm.

Outline

1. Introduction
2. Results
3. Visualization
4. Conclusions

UPV - EHU

Outline

1. Introduction
2. Results
3. Visualization
4. Conclusions

UPV - EHU

Combinatorial Optimization Problem

Definition

A Combinatorial Optimization Problem consists of finding the points σ^* that minimize or maximize a function f:

$$\sigma^* = \arg\min_{\sigma \in \Omega} f(\sigma)$$

where Ω is a finite or countable infinite set

Permutation-based COP

A COP where Ω is the space of permutations of size n

- Permutation Flowshop Scheduling Problem
- Quadratic Assignment Problem
- Linear Ordering Problem

Combinatorial Optimization Problem

Definition

A Combinatorial Optimization Problem consists of finding the points σ^* that minimize or maximize a function f:

$$\sigma^* = \arg\min_{\sigma \in \Omega} f(\sigma)$$

where Ω is a finite or countable infinite set

Permutation-based COP

A COP where Ω is the space of permutations of size n

- Permutation Flowshop Scheduling Problem
- Quadratic Assignment Problem
- Linear Ordering Problem

Combinatorial Optimization Problem

Definition

A Combinatorial Optimization Problem consists of finding the points σ^* that minimize or maximize a function f:

$$\sigma^* = \arg\min_{\sigma \in \Omega} f(\sigma)$$

where Ω is a finite or countable infinite set

Permutation-based COP

A COP where Ω is the space of permutations of size n

- Permutation Flowshop Scheduling Problem
- Quadratic Assignment Problem
- Linear Ordering Problem

Combinatorial Optimization Problem

Definition

A Combinatorial Optimization Problem consists of finding the points σ^* that minimize or maximize a function f:

$$\sigma^* = \arg\min_{\sigma \in \Omega} f(\sigma)$$

where Ω is a finite or countable infinite set

Permutation-based COP

A COP where Ω is the space of permutations of size n

- Permutation Flowshop Scheduling Problem
- Quadratic Assignment Problem
- Linear Ordering Problem

Permutation Flowshop Scheduling Problem

n jobs
m machines
Each job consists of m operations

Permutation Flowshop Scheduling Problem

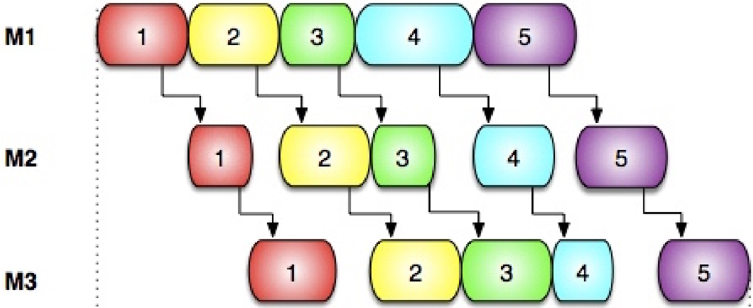
n jobs
m machines
Each job consists of m operations

Permutation Flowshop Scheduling Problem

n jobs
m machines
Each job consists of m operations

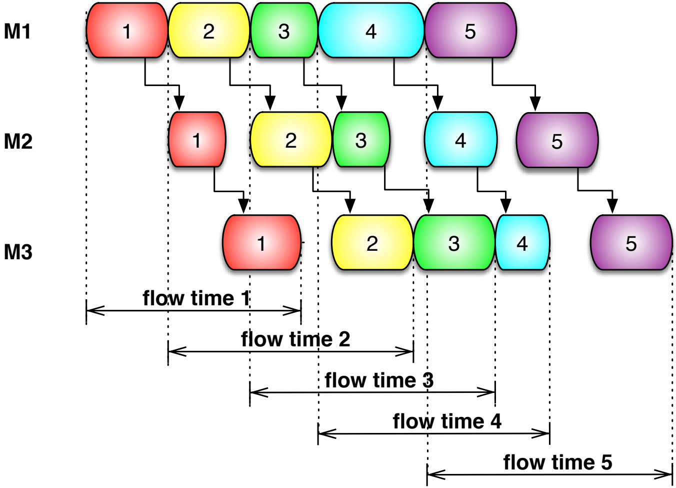
Permutation Flowshop Scheduling Problem

n jobs
m machines
Each job consists of *m* operations



UPV - EHU

Permutation Flowshop Scheduling Problem



Total Flow Time = FT1 + FT2 + FT3 + FT4 + FT5

minimize TFT

Neighborhood

A neighborhood N in a search space Ω is a mapping that assigns a set of neighboring solutions $N(\sigma) \in \mathcal{P}(\Omega)$ to each solution $\sigma \in \Omega$:

$$\begin{array}{rccc} N: & \Omega & \longrightarrow & \mathcal{P}(\Omega) \\ & \sigma & \longmapsto & N(\sigma) \end{array}$$

Neighborhoods. Examples

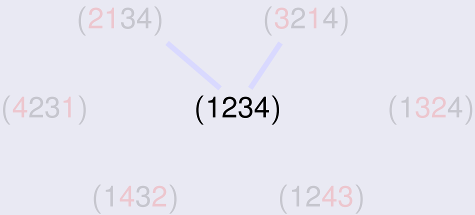
2-exchange or Swap

Swap two items, not necessarily adjacent

Neighborhoods. Examples

2-exchange or Swap

Swap two items, not necessarily adjacent



Neighborhoods. Examples

2-exchange or Swap

Swap two items, not necessarily adjacent

(2134) (3214)

(4231) (1234) (1324)

(1432) (1243)

Neighborhoods. Examples

2-exchange or Swap

Swap two items, not necessarily adjacent

(2134) (3214)

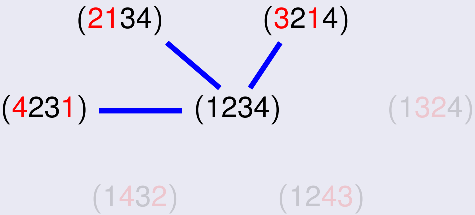
(4231) (1234) (1324)

(1432) (1243)

Neighborhoods. Examples

2-exchange or Swap

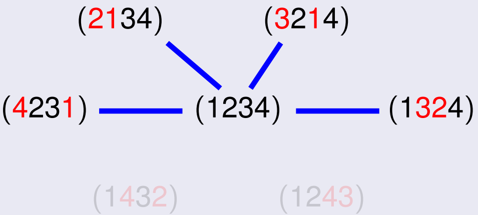
Swap two items, not necessarily adjacent



Neighborhoods. Examples

2-exchange or Swap

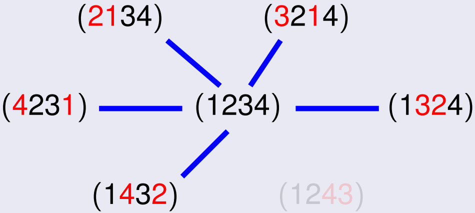
Swap two items, not necessarily adjacent



Neighborhoods. Examples

2-exchange or Swap

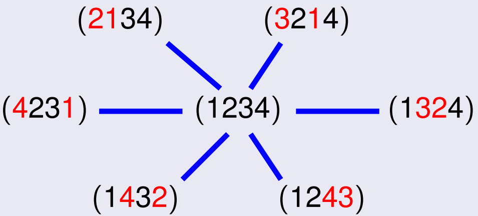
Swap two items, not necessarily adjacent



Neighborhoods. Examples

2-exchange or Swap

Swap two items, not necessarily adjacent



Neighborhoods. Examples

Insert

Move an item to a different position

(2134) (2314) (2341)

(1342)

(1324) (1234)

(3124)

(1243) (4123) (1423)

Neighborhoods. Examples

Insert

Move an item to a different position

(2134) (2314) (2341)
(1342)
(1324) (1234)
(3124)
(1243) (4123) (1423)

Neighborhoods. Examples

Insert

Move an item to a different position

(2134) (2314) (2341)

(1342)

(1324) (1234)

(3124)

(1243) (4123) (1423)

Neighborhoods. Examples

Insert

Move an item to a different position

(2134) (2314) (2341)

(1342)

(1324) (1234)

(3124)

(1243) (4123) (1423)

Neighborhoods. Examples

Insert

Move an item to a different position

(2134) (2314) (2341)

(1342)

(1324) (1234)

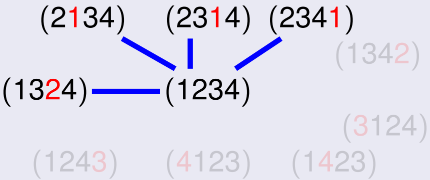
(3124)

(1243) (4123) (1423)

Neighborhoods. Examples

Insert

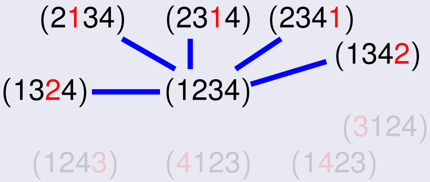
Move an item to a different position



Neighborhoods. Examples

Insert

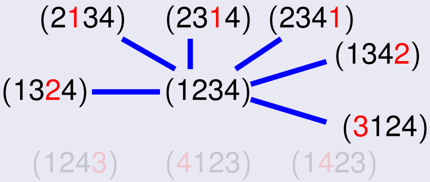
Move an item to a different position



Neighborhoods. Examples

Insert

Move an item to a different position

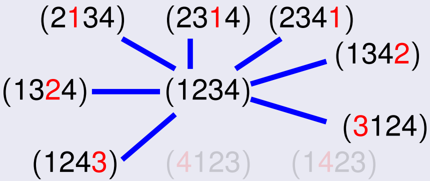


UPV - EHU

Neighborhoods. Examples

Insert

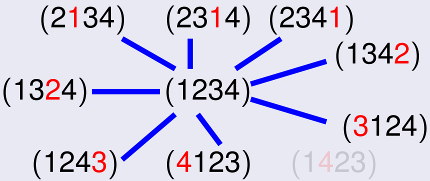
Move an item to a different position



Neighborhoods. Examples

Insert

Move an item to a different position

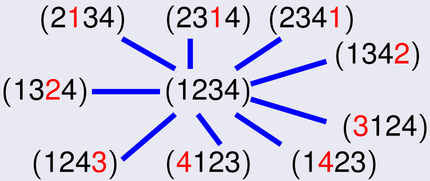


UPV - EHU

Neighborhoods. Examples

Insert

Move an item to a different position



Distance

- σ_1 and σ_2 are at distance i if, starting from σ_1, and moving from neighboring to neighboring solutions, the length of the shortest path to reach σ_2 is i.
- Two neighboring permutations are at distance one.
- Under the 2-exchange and the insert neighborhoods the maximum distance between two permutations is $n - 1$.

Distance

- σ_1 and σ_2 are at distance i if, starting from σ_1, and moving from neighboring to neighboring solutions, the length of the shortest path to reach σ_2 is i.
- Two neighboring permutations are at distance one.
- Under the 2-exchange and the insert neighborhoods the maximum distance between two permutations is $n - 1$.

Distance

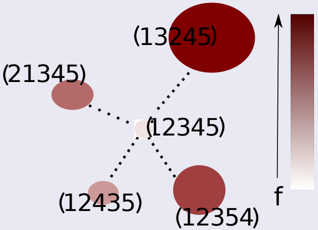
- σ_1 and σ_2 are at distance i if, starting from σ_1, and moving from neighboring to neighboring solutions, the length of the shortest path to reach σ_2 is i.
- Two neighboring permutations are at distance one.
- Under the 2-exchange and the insert neighborhoods the maximum distance between two permutations is $n - 1$.

UPV - EHU

Local Optima

- A solution $\sigma^* \in \Omega$ is a local optimum if

$$f(\sigma^*) \leq f(\sigma),\ \forall \sigma \in N(\sigma^*)\ \text{(local minimum)}$$



Attraction basins of local optima

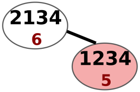
The attraction basin of a local optimum σ^*:

$$\mathcal{B}_{\sigma^*} = \{\sigma \in \Omega |\ \mathcal{H}(\sigma) = \sigma^*\},$$

where $\mathcal{H}$ is the operator that associates to each solution σ, the local optimum obtained after applying the algorithm

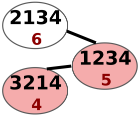
Deterministic best-improvement local search algorithm

Deterministic best-improvement local search algorithm

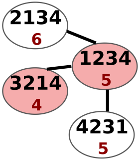


UPV - EHU

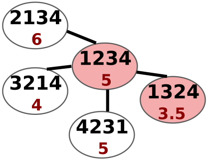
Deterministic best-improvement local search algorithm



Deterministic best-improvement local search algorithm

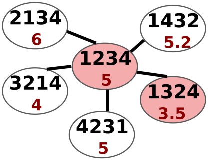


Deterministic best-improvement local search algorithm

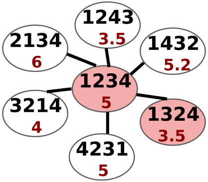


UPV - EHU

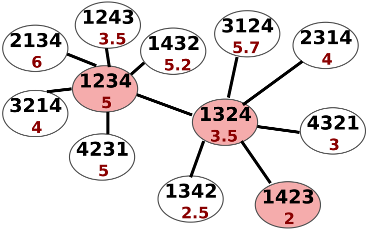
Deterministic best-improvement local search algorithm



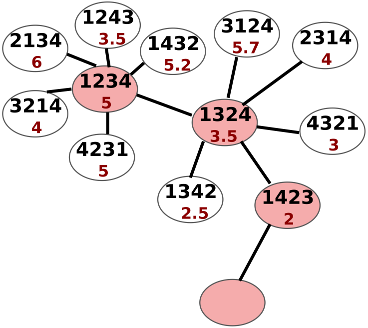
Deterministic best-improvement local search algorithm



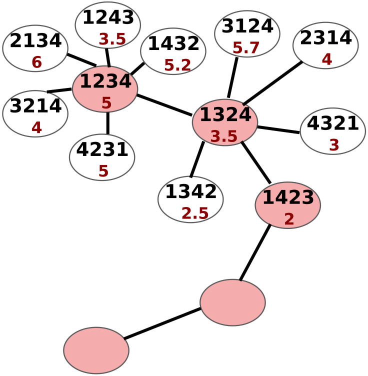
Deterministic best-improvement local search algorithm



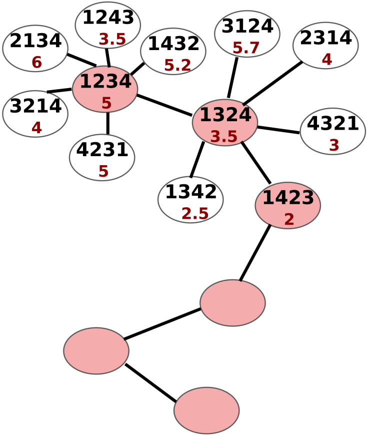
Deterministic best-improvement local search algorithm



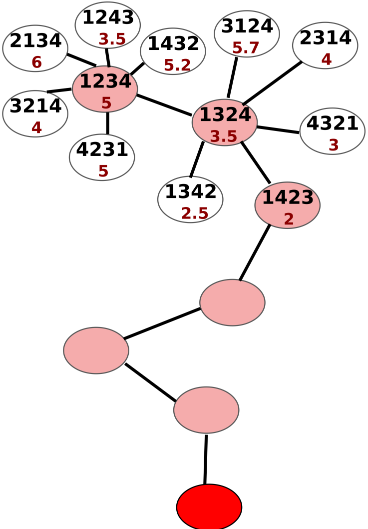
Deterministic best-improvement local search algorithm



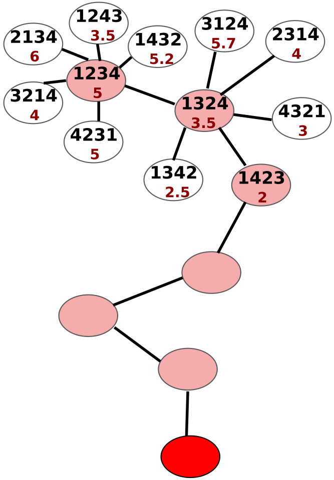
Deterministic best-improvement local search algorithm



Deterministic best-improvement local search algorithm



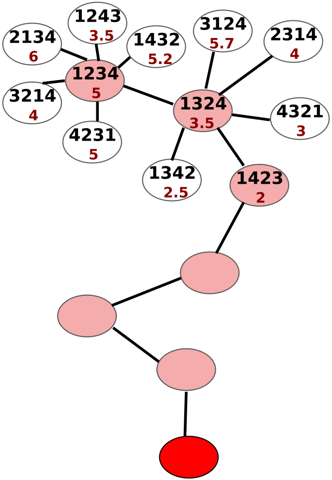
Deterministic best-improvement local search algorithm



Choose an initial solution $\sigma \in \Omega$
repeat
  $\sigma^* = \sigma$
  for each $\sigma'_i \in \mathcal{N}(\sigma^*)$ **do**
    if $f(\sigma'_i) < f(\sigma)$ **then**
      $\sigma = \sigma'_i$
    end if
  end for
until $\sigma = \sigma^*$

UPV - EHU

Deterministic best-improvement local search algorithm



Choose an initial solution $\sigma \in \Omega$
repeat
$\sigma^* = \sigma$
for each $\sigma_i' \in \mathcal{N}(\sigma^*)$ **do**
if $f(\sigma_i') < f(\sigma)$ **then**
$\sigma = \sigma_i'$
end if
end for
until $\sigma = \sigma^*$

The attraction basins are sets of paths!

UPV - EHU

Outline

1 Introduction

2 Results

3 Visualization

4 Conclusions

Experimental Design

- 9 PFSP instances of Taillard's benchmark.
- 10 jobs and 5 machines (n=10).
- The local optima and the attraction basins are calculated considering the 2-exchange and the insert neighborhoods.

UPV - EHU

Experimental Design

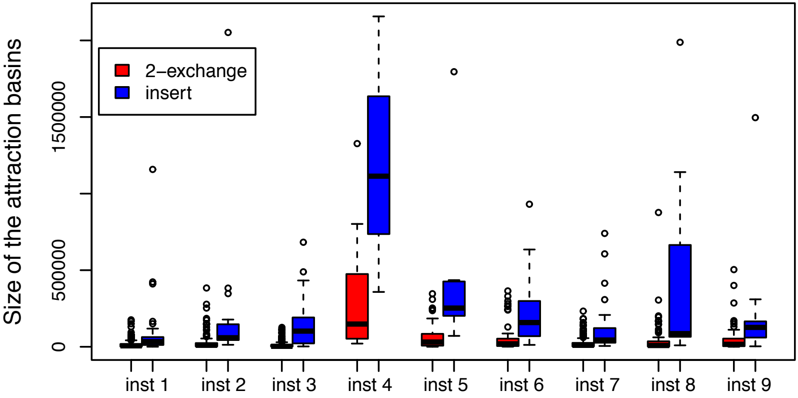
- 9 PFSP instances of Taillard's benchmark.
- 10 jobs and 5 machines (n=10).
- The local optima and the attraction basins are calculated considering the 2-exchange and the insert neighborhoods.

Experimental Design

- 9 PFSP instances of Taillard's benchmark.
- 10 jobs and 5 machines (n=10).
- The local optima and the attraction basins are calculated considering the 2-exchange and the insert neighborhoods.

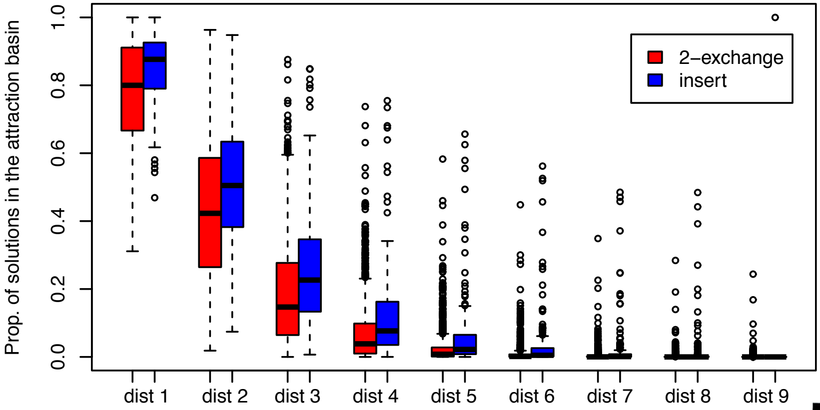
No. of local optima and sizes of attraction basins

# Loc. Opt	inst 1	inst 2	inst 3	inst 4	inst 5	inst 6	inst 7	inst 8	inst 9
2-exch.	225	117	295	11	58	58	158	83	80
insert	43	16	24	3	8	15	31	8	19



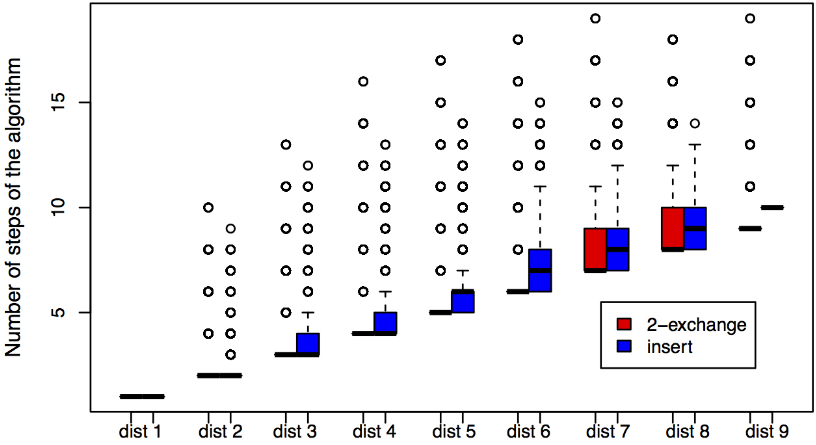
UPV - EHU

Distance to local optima vs. number of solutions in attraction basins

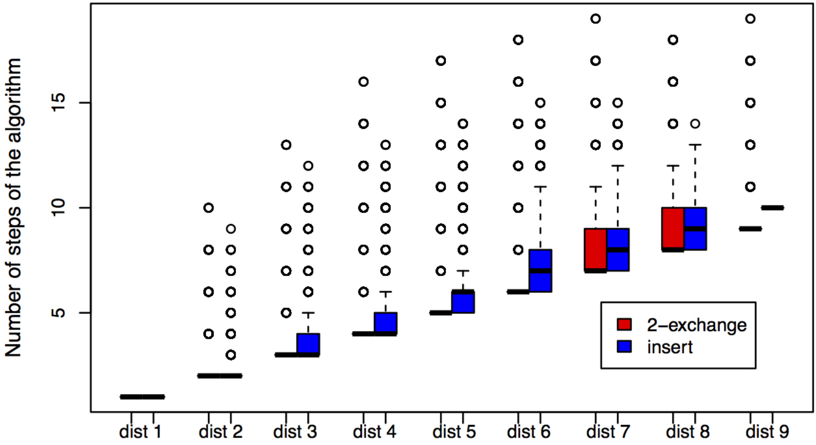


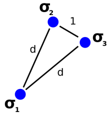
UPV - EHU

Distance to local optima vs. number of steps of the algorithm

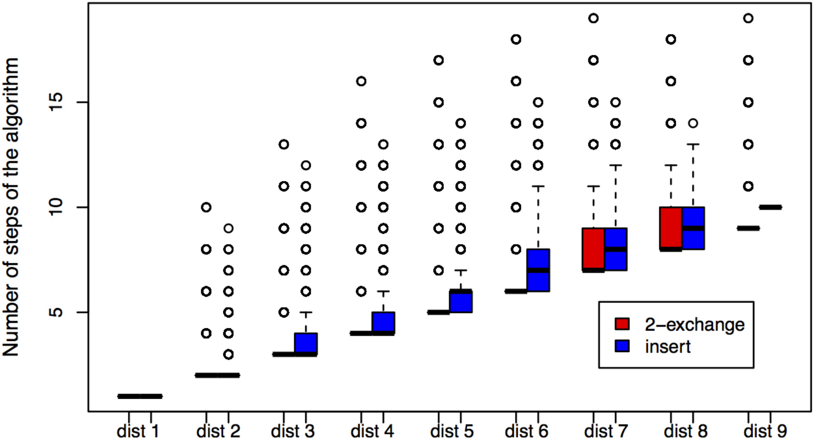


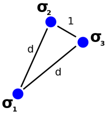
Distance to local optima vs. number of steps of the algorithm

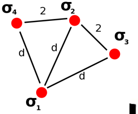




Distance to local optima vs. number of steps of the algorithm







UPV - EHU

Outline

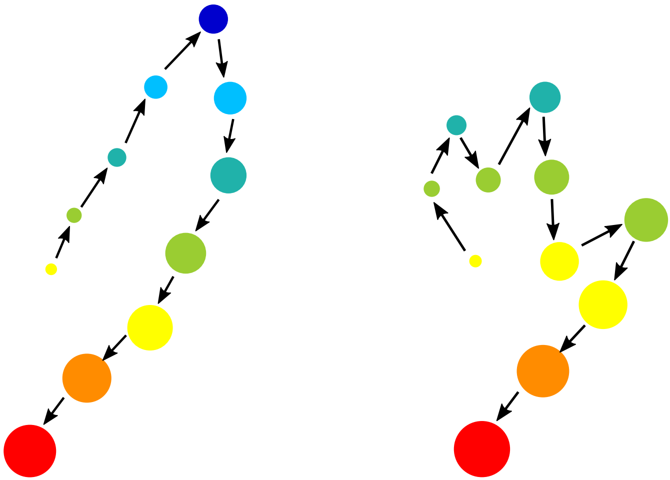
1 Introduction

2 Results

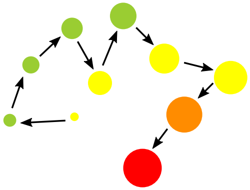
3 Visualization

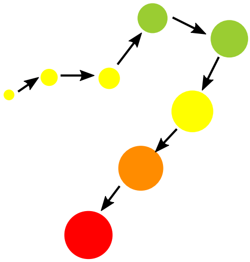
4 Conclusions

Visualization of paths: PFSP 2-exchange



Visualization of paths: PFSP insert





Outline

1 Introduction

2 Results

3 Visualization

4 Conclusions

UPV - EHU

Conclusions

The algorithm goes for a walk before finding the optimum!

Future Work

- Analysis of larger instances
- Use this information to design/modify algorithms

Conclusions

The algorithm goes for a walk before finding the optimum!

Future Work

- Analysis of larger instances
- Use this information to design/modify algorithms

Hill-Climbing Algorithm: let's go for a walk before finding the optimum

Leticia Hernando, Alexander Mendiburu and Jose A. Lozano

Intelligent Systems Group
University of the Basque Country UPV/EHU, Spain

July, 11th 2018

2018 IEEE Congress on Evolutionary Computation