

# Estimation of Distribution Algorithms for Permutation-based Problems

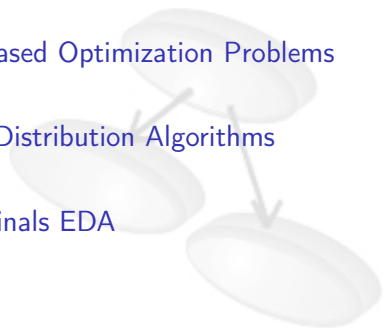
Josu Ceberio, Alexander Mendiburu, Jose A. Lozano

Intelligent Systems Group  
Department of Computer Science and Artificial Intelligence  
The University of the Basque Country

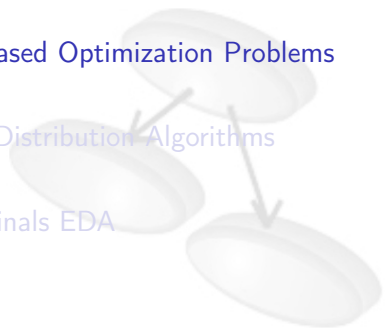


Donostia, 13<sup>th</sup> May 2011

# Outline

- 
- 1 Permutation-based Optimization Problems
  - 2 Estimation of Distribution Algorithms
  - 3  $K$ -Order Marginals EDA
  - 4 Mallows EDA
  - 5 Future Work

# Outline

- 
- 1 Permutation-based Optimization Problems
  - 2 Estimation of Distribution Algorithms
  - 3 *K*-Order Marginals EDA
  - 4 Mallows EDA
  - 5 Future Work

# Combinatorial Optimization Problems and Permutations

## Definition

- A specific subset of **NP-Hard** optimization problems
- Problems whose solution can be represented as permutations

# Combinatorial Optimization Problems and Permutations

## Definition

- A specific subset of **NP-Hard** optimization problems
- Problems whose solution can be represented as permutations

# Permutations

## What are permutations?

- A permutation is understood as a vector  $\sigma = (\sigma_1, \dots, \sigma_n)$  of the indexes  $\{1, \dots, n\}$  such that  $\sigma_i \neq \sigma_j$  for all  $i \neq j$

# Permutations

## What are permutations?

- A permutation is understood as a vector  $\sigma = (\sigma_1, \dots, \sigma_n)$  of the indexes  $\{1, \dots, n\}$  such that  $\sigma_i \neq \sigma_j$  for all  $i \neq j$
- Example:

$$\sigma = (1 \ 4 \ 5 \ 8 \ 2 \ 6 \ 3 \ 7)$$

# Travelling Salesman Problem

1

2

3

6

4

5

- Given a set of  $n$  cities and the distances between them  $D = [d_{ij}]$  find the shortest path that passes for each city once and comes back to the first city
- The objective function can be written as:

$$f(\sigma_1 \sigma_2 \dots \sigma_n) = \sum_{i=1}^{n-1} d_{\sigma_i, \sigma_{i+1}} + d_{\sigma_n, \sigma_1}$$



# Travelling Salesman Problem

1

2

3

6

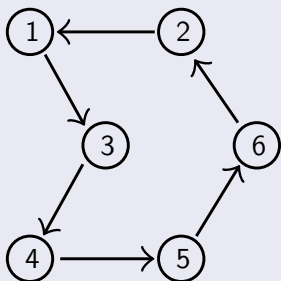
4

5

- Given a set of  $n$  cities and the distances between them  $D = [d_{ij}]$  find the shortest path that passes for each city once and comes back to the first city
- The objective function can be written as:

$$f(\sigma_1 \sigma_2 \dots \sigma_n) = \sum_{i=1}^{n-1} d_{\sigma_i, \sigma_{i+1}} + d_{\sigma_n, \sigma_1}$$

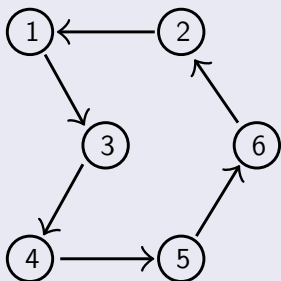
# Travelling Salesman Problem



- A solution can be represented as a permutation:

$$\sigma = (1 \ 3 \ 4 \ 5 \ 6 \ 2)$$

# Travelling Salesman Problem



## Interpretation

The objective function is independent of the absolute position of a city. It only depends on the **relative position of consecutive indexes**

$$f(1\ 3\ 4\ 5\ 6\ 2) = f(2\ 1\ 3\ 4\ 5\ 6)$$

# Flow Shop Scheduling Problem

## Definition

- It consists of scheduling  $n$  jobs on  $m$  machines.
- A job consists of  $m$  operations and the  $j^{th}$  operation of each job must be processed on machine  $j$  for a specific time.
- The goal of the optimization is to minimize the processing time of all the jobs.

## Flow Shop Scheduling Problem

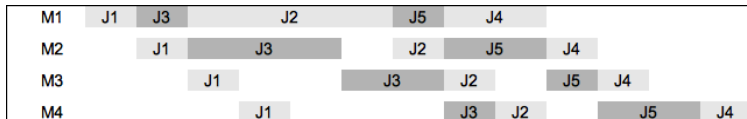


Figure: Example of a solution for an instance of 5 jobs on 4 machines

### Representation

A solution can be represented as a permutation:  $\sigma = (1 \ 3 \ 2 \ 5 \ 4)$

### Interpretation

In this case the objective function depends on the absolute position of each index

## Flow Shop Scheduling Problem

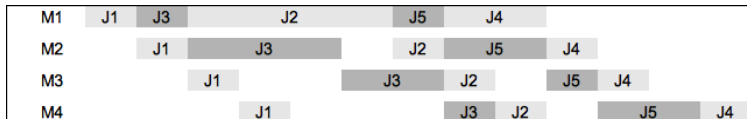


Figure: Example of a solution for an instance of 5 jobs on 4 machines

### Representation

A solution can be represented as a permutation:  $\sigma = (1 \ 3 \ 2 \ 5 \ 4)$

### Interpretation

In this case the objective function depends on the absolute position of each index

## Flow Shop Scheduling Problem

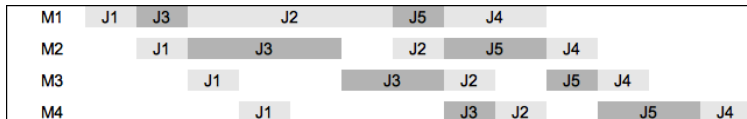


Figure: Example of a solution for an instance of 5 jobs on 4 machines

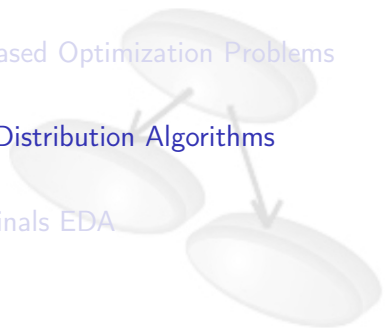
### Representation

A solution can be represented as a permutation:  $\sigma = (1 \ 3 \ 2 \ 5 \ 4)$

### Interpretation

In this case the objective function depends on the absolute position of each index

# Outline

- 
- 1 Permutation-based Optimization Problems
  - 2 Estimation of Distribution Algorithms**
  - 3 *K*-Order Marginals EDA
  - 4 Mallows EDA
  - 5 Future Work



# Evolutionary Computation

## Estimation of Distribution Algorithms (EDAs)

- An Evolutionary Algorithm
- Similar to Genetic Algorithms
- **Learn** a probability distribution from the selected individuals
- **Sample** the probability distribution to obtain the new population

# Evolutionary Computation

## Estimation of Distribution Algorithms (EDAs)

- An Evolutionary Algorithm
- Similar to Genetic Algorithms
- **Learn** a probability distribution from the selected individuals
- **Sample** the probability distribution to obtain the new population

# Evolutionary Computation

## Estimation of Distribution Algorithms (EDAs)

- An Evolutionary Algorithm
- Similar to Genetic Algorithms
- **Learn** a probability distribution from the selected individuals
- **Sample** the probability distribution to obtain the new population

## Pseudocode of EDAs

Obtain an initial population of individuals  $D_0$

**Repeat** until a stopping criterion is met

*Select from  $D_i$  a subset of individuals  $D_i^S$*

*Learn a probability distribution  $p_i(\mathbf{x})$  from  $D_i^S$*

*Sample  $p_i(\mathbf{x})$  to obtain  $D_{i+1/2}$*

*Create the new population  $D_{i+1}$  from  $D_i$  and  $D_{i+1/2}$*

# EDAs for Permutation-based problems

## Adapting classical approaches

- Adaptations of integer encoding EDA approaches
- Adaptations of real encoding EDA approaches

# EDAs for Permutation-based problems

## Adapting classical approaches

- Adaptations of integer encoding EDA approaches
- Adaptations of real encoding EDA approaches

## EDAs designed for integer problems

### Basics

- These algorithms learn, a probability distribution over a set (of variables)  $\Omega = \Omega_1 \times \Omega_2 \times \dots \times \Omega_n$  where  $\Omega_i = \{1, 2, \dots, r_i\}$ ,  $r_i \in \mathbb{N}$   $i = 1, \dots, n$
- Dependencies:
  - Univariate: UMDA
  - Bivariate: MIMIC
  - Multivariate: EBNA, Dependency-Trees

## EDAs designed for integer problems

### Drawbacks

- The sampling of these models may not provide permutations
- At the time of sampling the learnt relation between the variables is lost
- These methods do not learn a probability distribution in a permutation space



## EDAs designed for continuous problems

### Basics

- Optimization is carried out in the continuous space
- Given a real vector  $(x_1, x_2, \dots, x_n)$  of length  $n$ , a permutation can be obtained by ranking the positions using the values  $x_i$ ,  $(i = 1, \dots, n)$
- Given

$(2.35, 3.42, 9.35, 0.32, 11.54, 10.42, 5.23, 4.2, 7.8)$

the permutation obtained is  $\sigma = (2\ 3\ 7\ 1\ 9\ 8\ 5\ 4\ 6)$


- Examples: UMDA<sub>c</sub>, MIMIC<sub>c</sub>, EGNA...

## EDAs designed for continuous problems

### Drawbacks

- There are a lot of redundancy in the codification: many real vectors correspond with the same permutation
- There is not explicit distribution over permutations: Which is the probability of a particular permutation?

# Outline

- 
- 1 Permutation-based Optimization Problems
  - 2 Estimation of Distribution Algorithms
  - 3 K-Order Marginals EDA**
  - 4 Mallows EDA
  - 5 Future Work

# K-Order Marginals EDA

## Learning

- At each step a matrix of  $K$ -order marginals is learnt
- Each entry of the matrix is given by the probability:

$$P(\sigma_{i_1} = j_1, \dots, \sigma_{i_k} = j_k)$$

- It is calculated from the number of times that the configuration  $(\sigma_{i_1} = j_1, \dots, \sigma_{i_k} = j_k)$  appears in the selected individuals.

# K-Order Marginals EDA

Table: 2-order marginals matrix

	Index Combinations											
	(1,2)	(1,3)	(1,4)	(2,1)	(2,3)	(2,4)	(3,1)	(3,2)	(3,4)	(4,1)	(4,2)	(4,3)
(1,2)	0.20	0.05	0.10	0.05	0.10	0.05	0.05	0.10	0.05	0.05	0.10	0.10
(1,3)	0.05	0.20	0.10	0.10	0.05	0.05	0.05	0.05	0.10	0.10	0.10	0.05
(1,4)	0.10	0.10	0.15	0.05	0.05	0.10	0.10	0.05	0.05	0.10	0.05	0.10
(2,3)	0.05	0.05	0.05	0.10	0.15	0.15	0.10	0.10	0.05	0.05	0.05	0.10
(2,4)	0.05	0.05	0.05	0.10	0.15	0.15	0.10	0.05	0.10	0.05	0.10	0.05
(3,4)	0.05	0.10	0.10	0.10	0.05	0.05	0.05	0.10	0.15	0.10	0.05	0.10

# Marginals Sampling

## Sampling

- The individual is initialized as empty  $S = (-, -, -, -)$
- The sampling process is done by sampling a position of the individual at each step in the  $M_i$  matrix  $i = 1, \dots, k$

# Marginals Sampling

## Sampling

- The individual is initialized as empty  $S = (-, -, -, -)$
- The sampling process is done by sampling a position of the individual at each step in the  $M_i$  matrix  $i = 1, \dots, k$

## Step 1

- The uniformly random selected position is 2.
- $M_1$  marginals matrix:

		Index Combinations			
		1	2	3	4
Positions	1	0.41	0.16	0.16	0.25
	2	0.09	0.50	0.25	0.16
	3	0.25	0.16	0.33	0.25
	4	0.25	0.16	0.25	0.33

- Sampled index is 3.

# Marginals Sampling

## Step 2

- Partially sampled individual is  $S = (-, 3, -, -)$
- The uniformly random selected combination of positions is  $(2, 3)$ .
- $M_2$  marginals matrix:

		Index Combinations											
		(1,2)	(1,3)	(1,4)	(2,1)	(2,3)	(2,4)	<b>(3,1)</b>	<b>(3,2)</b>	<b>(3,4)</b>	(4,1)	(4,2)	(4,3)
Positions	(1,2)	0.20	0.05	0.10	0.05	0.10	0.05	0.05	0.10	0.05	0.05	0.10	0.10
	(1,3)	0.05	0.20	0.10	0.10	0.05	0.05	0.05	0.05	0.10	0.10	0.10	0.05
	(1,4)	0.10	0.10	0.15	0.05	0.05	0.10	0.10	0.05	0.05	0.10	0.05	0.10
	<b>(2,3)</b>	0.05	0.05	0.05	0.10	0.15	0.15	0.40	0.40	0.20	0.05	0.05	0.10
	(2,4)	0.05	0.05	0.05	0.10	0.15	0.15	0.10	0.05	0.10	0.05	0.10	0.05
	(3,4)	0.05	0.10	0.10	0.10	0.05	0.05	0.05	0.10	0.15	0.10	0.05	0.10

- Sampled indexes combination is  $(3, 2)$ .



# Marginals Sampling

## Step 3

- Partially sampled individual is  $S = (-, 3, 2, -)$
- The uniformly random selected combination of positions is  $(3, 4)$ .
- $M_2$  marginals matrix:

		Index Combinations											
		(1,2)	(1,3)	(1,4)	<b>(2,1)</b>	(2,3)	<b>(2,4)</b>	(3,1)	(3,2)	(3,4)	(4,1)	(4,2)	(4,3)
Positions	(1,2)	0.20	0.05	0.10	0.05	0.10	0.05	0.05	0.10	0.05	0.05	0.10	0.10
	(1,3)	0.05	0.20	0.10	0.10	0.05	0.05	0.05	0.05	0.10	0.10	0.10	0.05
	(1,4)	0.10	0.10	0.15	0.05	0.05	0.10	0.10	0.05	0.05	0.10	0.05	0.10
	(2,3)	0.05	0.05	0.05	0.10	0.15	0.15	0.10	0.10	0.05	0.05	0.05	0.10
	(2,4)	0.05	0.05	0.05	0.10	0.15	0.15	0.10	0.05	0.10	0.05	0.10	0.05
	<b>(3,4)</b>	0.05	0.10	0.10	0.66	0.05	0.33	0.05	0.10	0.15	0.10	0.05	0.10

- Sampled indexes combination is  $(2, 1)$ .

## Marginals Sampling

### Step4

- Partially sampled individual is  $S = (-, 3, 2, 1)$
- Remaining index is placed in position 1.
- The new individual is  $S = (4, 3, 2, 1)$ .

## Experiments

### Execution Parameter Set

Parameter	Value
<i>K</i> -order	{1, 2, 3}
Population size range	{10 <i>n</i> , 20 <i>n</i> , 40 <i>n</i> , 80 <i>n</i> , 160 <i>n</i> , 320 <i>n</i> , 640 <i>n</i> }.
Selection size	Population size / 2.
Offspring size	Population size - 1.
Selection type	Ranking selection method.
Elitism selection method	The best individual of the previous generation is guaranteed to survive.
Stopping criteria	A maximum number of generations: 100 <i>n</i> .

### Instances

- TSP (Grostel 17)
- FSSP (Taillard 20 jobs 10 machines)

# Experiments

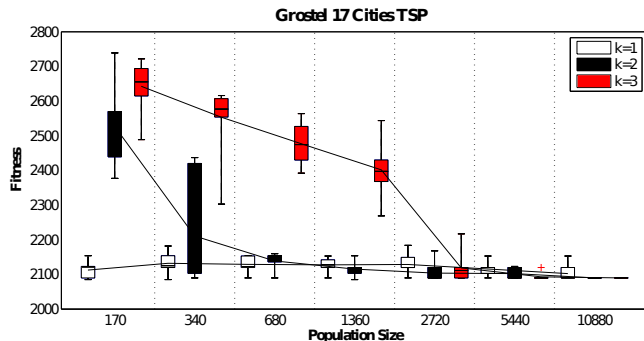


Figure:  $K$ -order marginals EDA solving Groset 17 TSP instance.

## Experiments

**Table:** Results of  $K$ -order marginals EDA for Grostel 17 TSP instance.

Pop. Size	$k = 1$			$k = 2$			$k = 3$		
	<i>Best</i>	<i>Mean</i>	<i>Dev</i>	<i>Best</i>	<i>Mean</i>	<i>Dev</i>	<i>Best</i>	<i>Mean</i>	<i>Dev</i>
170	2085	2112.5	23.6	2377	2526.8	107.0	2489	2642.7	67.5
340	2085	2132.1	26.4	2090	2210.9	153.4	2303	2553.5	92.1
680	2090	2129.0	23.2	2090	2138.9	26.5	2392	2477.0	56.0
1360	2090	2127.7	20.1	2085	2115.5	19.3	2269	2401.7	71.1
2720	2090	2128.6	31.2	2090	2103.8	25.7	2090	2118.9	37.9
5440	2090	2115.6	18.2	2090	2102.3	15.9	2090	2093.0	9.4
10880	2090	2102.3	21.7	2090	2090.0	0.0	2090	2090.0	0.0

# Experiments

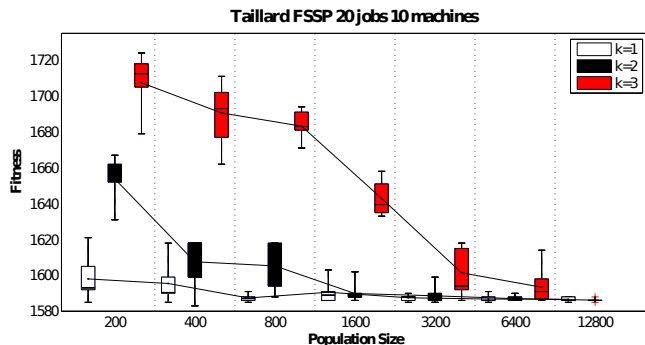


Figure: *K*-order marginals EDA solving Taillard 20-10 FSSP instance.

# Experiments

**Table:** Results of  $K$ -order marginals EDA for Taillard 20-10 FSSP.

Pop. Size	$k = 1$			$k = 2$			$k = 3$		
	<i>Best</i>	<i>Mean</i>	<i>Dev</i>	<i>Best</i>	<i>Mean</i>	<i>Dev</i>	<i>Best</i>	<i>Mean</i>	<i>Dev</i>
200	1585	<b>1598.0</b>	12.5	1631	1653.8	12.9	1679	1707.4	14.6
400	1585	<b>1595.5</b>	11.4	1583	1607.6	14.7	1662	1690.6	16.5
800	1585	<b>1587.4</b>	1.9	1588	1605.3	12.0	1671	1683.2	7.5
1600	1586	<b>1590.6</b>	5.6	1586	1589.9	4.9	1633	1642.9	9.5
3200	1585	<b>1587.5</b>	1.7	1585	1588.6	4.1	1586	1601.4	12.5
6400	1585	<b>1586.8</b>	1.7	1586	1587.0	1.4	1586	1593.4	8.6
12800	1585	<b>1586.7</b>	1.1	1585	<b>1586.2</b>	1.0	-	-	-

# Conclusions

## K-order Marginals EDA

- Higher order models are slightly better, however higher populations are needed
- Low order marginals can be computed due to the computational costs
- Estimating probability distributions of higher populations makes the model less random.
- We do not have an explicit probability distribution



# Conclusions

## K-order Marginals EDA

- Higher order models are slightly better, however higher populations are needed
- Low order marginals can be computed due to the computational costs
- Estimating probability distributions of higher populations makes the model less random.
- We do not have an explicit probability distribution

# Conclusions

## K-order Marginals EDA

- Higher order models are slightly better, however higher populations are needed
- Low order marginals can be computed due to the computational costs
- Estimating probability distributions of higher populations makes the model less random.
- We do not have an explicit probability distribution

# Conclusions

## K-order Marginals EDA

- Higher order models are slightly better, however higher populations are needed
- Low order marginals can be computed due to the computational costs
- Estimating probability distributions of higher populations makes the model less random.
- We do not have an explicit probability distribution


# Conclusions

## K-order Marginals EDA

- Higher order models are slightly better, however higher populations are needed
- Low order marginals can be computed due to the computational costs
- Estimating probability distributions of higher populations makes the model less random.
- We do not have an explicit probability distribution

Learn a probability distributions over the space of permutations

# Outline

- 
- ① Permutation-based Optimization Problems
  - ② Estimation of Distribution Algorithms
  - ③ *K*-Order Marginals EDA
  - ④ Mallows EDA
  - ⑤ Future Work

# Mallows model

## Definition

- The Mallows model estimates the *distance* of the selected individuals (permutations) to the central permutation
- Probability Distribution

$$p(\sigma) = \frac{1}{Z(\theta)} e^{-\theta d_K(\sigma, \sigma_0)}$$

- Parameters:
  - Central permutation  $\sigma_0$
  - Spread parameter  $\theta$
  - $d_K(\sigma, \sigma_0)$  is Kendall's distance: minimum number of adjacent transpositions to go from  $\sigma_0$  to  $\sigma$
  - Normalization constant  $Z(\theta)$

# Mallows model

## Definition

- The Mallows model estimates the *distance* of the selected individuals (permutations) to the central permutation Probability Distribution

$$p(\sigma) = \frac{1}{Z(\theta)} e^{-\theta d_K(\sigma, \sigma_0)}$$

- Parameters:
  - Central permutation  $\sigma_0$
  - Spread parameter  $\theta$
  - $d_K(\sigma, \sigma_0)$  is Kendall's distance: minimum number of adjacent transpositions to go from  $\sigma_0$  to  $\sigma$
  - Normalization constant  $Z(\theta)$

# Mallows model

## Definition

- The Mallows model estimates the *distance* of the selected individuals (permutations) to the central permutation Probability Distribution

$$p(\sigma) = \frac{1}{Z(\theta)} e^{-\theta d_K(\sigma, \sigma_0)}$$

- Parameters:
  - Central permutation  $\sigma_0$
  - Spread parameter  $\theta$
  - $d_K(\sigma, \sigma_0)$  is Kendall's distance: minimum number of adjacent transpositions to go from  $\sigma_0$  to  $\sigma$
  - Normalization constant  $Z(\theta)$



# Mallows model

## Definition

- The Mallows model estimates the *distance* of the selected individuals (permutations) to the central permutation Probability Distribution

$$p(\sigma) = \frac{1}{Z(\theta)} e^{-\theta d_K(\sigma, \sigma_0)}$$

- Parameters:
  - Central permutation  $\sigma_0$
  - Spread parameter  $\theta$
  - $d_K(\sigma, \sigma_0)$  is Kendall's distance: minimum number of adjacent transpositions to go from  $\sigma_0$  to  $\sigma$
  - Normalization constant  $Z(\theta)$

## $K$ -order Marginals vs. Mallows model

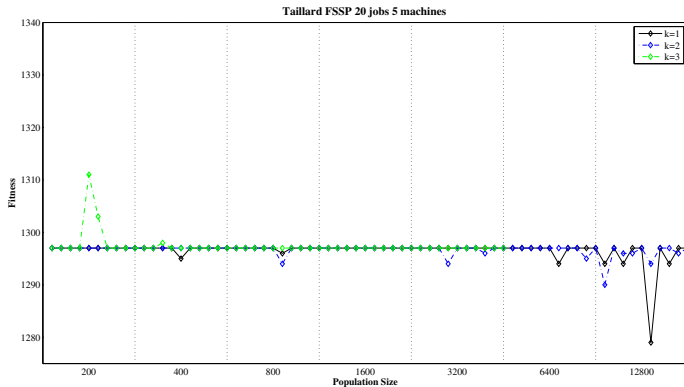


Figure: Experiments results. Taillard 20-10 FSSP instance.

## $K$ -order Marginals vs. Mallows model

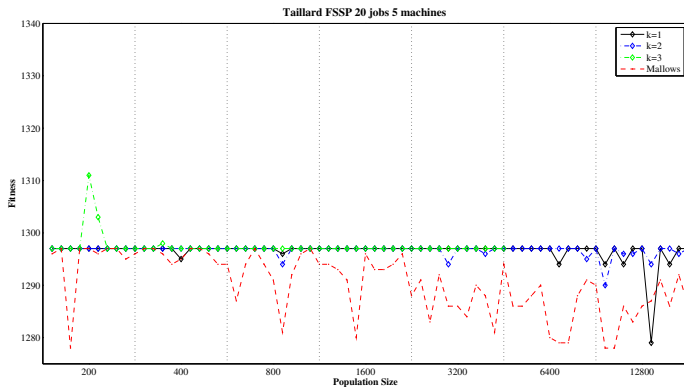


Figure: Experiments results. Taillard 20-10 FSSP instance.

# $K$ -order Marginals vs. Mallows model vs. Classical EDAs

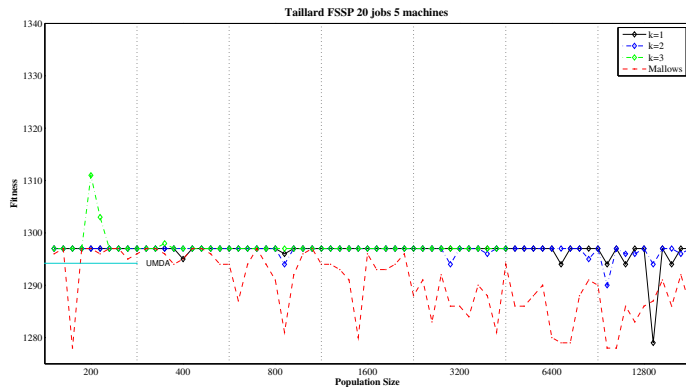


Figure: Experiments results. Taillard 20-10 FSSP instance.

# $K$ -order Marginals vs. Mallows model vs. Classical EDAs

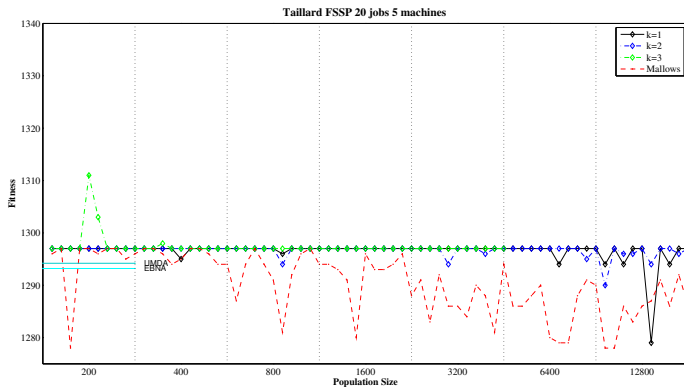


Figure: Experiments results. Taillard 20-10 FSSP instance.

# $K$ -order Marginals vs. Mallows model vs. Classical EDAs

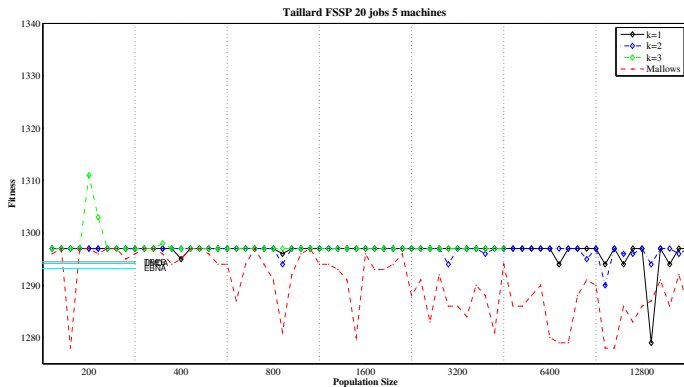


Figure: Experiments results. Taillard 20-10 FSSP instance.

# $K$ -order Marginals vs. Mallows model vs. Classical EDAs

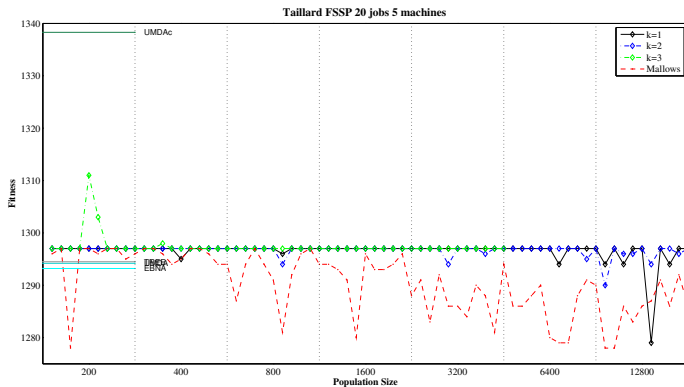


Figure: Experiments results. Taillard 20-10 FSSP instance.

# K-order Marginals vs. Mallows model vs. Classical EDAs

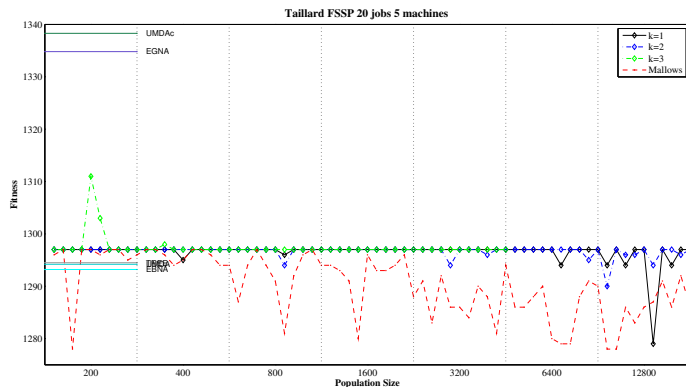


Figure: Experiments results. Taillard 20-10 FSSP instance.



# $K$ -order Marginals vs. Mallows model vs. Classical EDAs

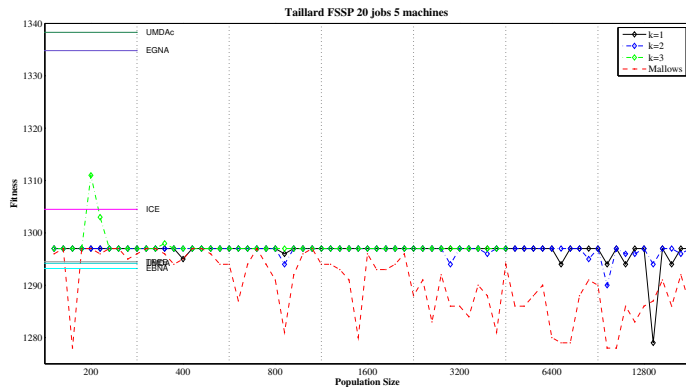


Figure: Experiments results. Taillard 20-10 FSSP instance.

# $K$ -order Marginals vs. Mallows model vs. Classical EDAs

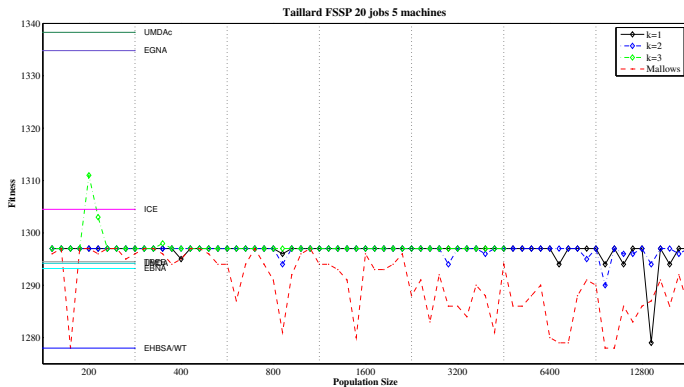


Figure: Experiments results. Taillard 20-10 FSSP instance.

# K-order Marginals vs. Mallows model vs. Classical EDAs

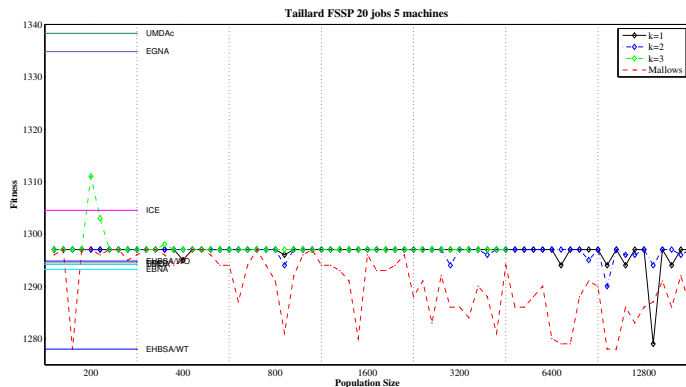


Figure: Experiments results. Taillard 20-10 FSSP instance.

# $K$ -order Marginals vs. Mallows model vs. Classical EDAs

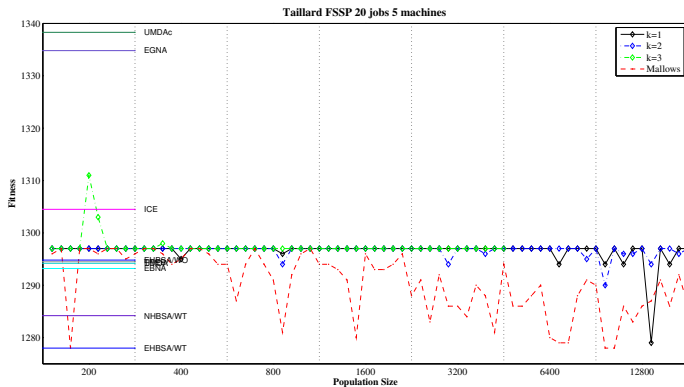


Figure: Experiments results. Taillard 20-10 FSSP instance.

# $K$ -order Marginals vs. Mallows model vs. Classical EDAs

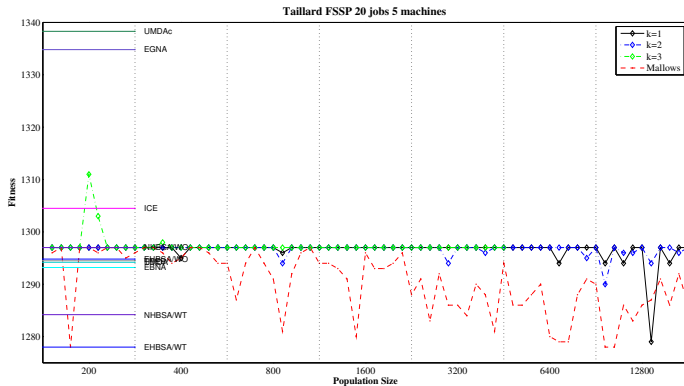


Figure: Experiments results. Taillard 20-10 FSSP instance.

# K-order Marginals vs. Mallows model vs. Classical EDAs

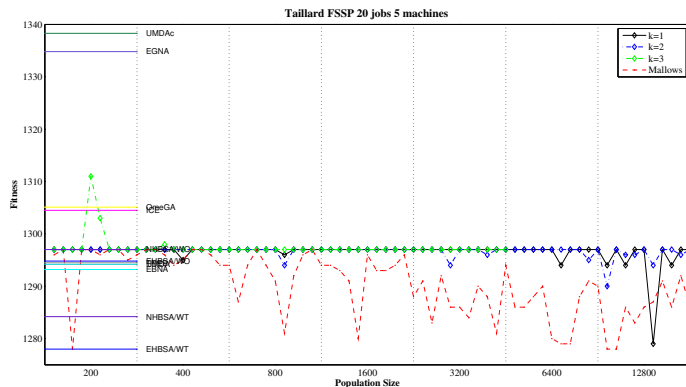



Figure: Experiments results. Taillard 20-10 FSSP instance.

# Outline

- 
- ① Permutation-based Optimization Problems
  - ② Estimation of Distribution Algorithms
  - ③ *K*-Order Marginals EDA
  - ④ Mallows EDA
  - ⑤ Future Work

# Generalized Mallows model

## Definition

- Probability Distribution

$$p(\sigma) = \frac{1}{Z(\theta_j)} e^{-\theta_j d_K(\sigma, \sigma_0)}$$

- A different spread parameter  $\theta_j$  to each position  $j$



## Cayley's Distance

### Definition

- Kendall's distance,  $d_K(\sigma, \sigma_0)$ : minimum number of adjacent transpositions to go from  $\sigma$  to  $\sigma_0$
- Cayley's distance,  $d_C(\sigma, \sigma_0)$ : minimum number transpositions, **not necessarily adjacent**, to go from  $\sigma$  to  $\sigma_0$

# Cayley's Distance

## Definition

- Kendall's distance,  $d_K(\sigma, \sigma_0)$ : minimum number of adjacent transpositions to go from  $\sigma$  to  $\sigma_0$
- Cayley's distance,  $d_C(\sigma, \sigma_0)$ : minimum number transpositions, **not necessarily adjacent**, to go from  $\sigma$  to  $\sigma_0$

# Estimation of Distribution Algorithms for Permutation-based Problems

Josu Ceberio, Alexander Mendiburu, Jose A. Lozano

Intelligent Systems Group  
Department of Computer Science and Artificial Intelligence  
The University of the Basque Country



Donostia, 13<sup>th</sup> May 2011