Computer software engineering

June 2020

LOPMENT OF AGGREGATION MET OF PARTIALLY ORDERED SETS

César garcía cabeza

Tutors: irene díaz rodríguez & elías fernández combarro

Table of contents

introduction

Real life application

pepe

pepa

pepo

real life application

best

pepa

pepe

pepo

worst

motivation

n	combinations
3	6
4	24
5	120
6	720
7	5040
8	40320
9	362880
10	3628800
11	39916800
12	479001600

Np-har

1) C. Bachmaier et al. “On the hardness of maximum rank aggregat

Basic concepts

Partially ordered set or *poset*

Set p with a binary relati

Reflexivity

Antisymm
etry

transitivit
y

2) CMU - department of mathematical sciences. *partially o*

Partially ordered set or *poset*

Set p with a binary relati

Reflexivity

Antisymm etry

transitivit y

$x \leq x$

2) CMU - department of mathematical sciences. *partially o*

Partially ordered set or *poset*

Set p with a binary relati

Reflexivity

Antisymm
etry

transitivit
y

$$x \leq y, y \leq x \Rightarrow x = y$$

2) CMU - department of mathematical sciences. *partially o*

Partially ordered set or *poset*

Set p with a binary relati

Reflexivity

Antisymm
etry

transitivit
y

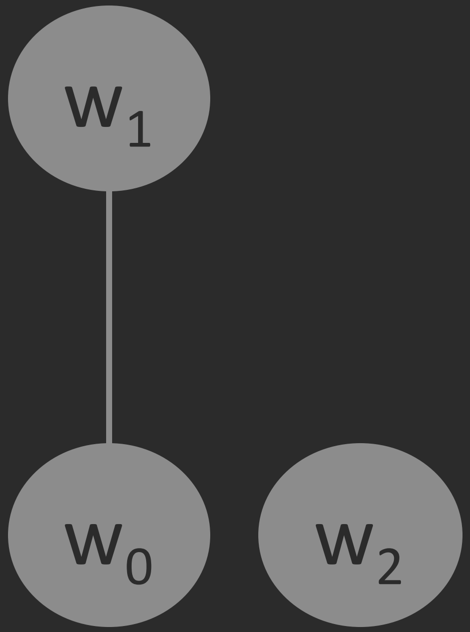
$$x \leq y, y \leq z \Rightarrow x \leq z$$

2) CMU - department of mathematical sciences. *partially o*

Comparable objects

$\boxed{x \leq y}$ or $\boxed{y \leq x}$

poset representation



$\equiv$

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Hasse diagra

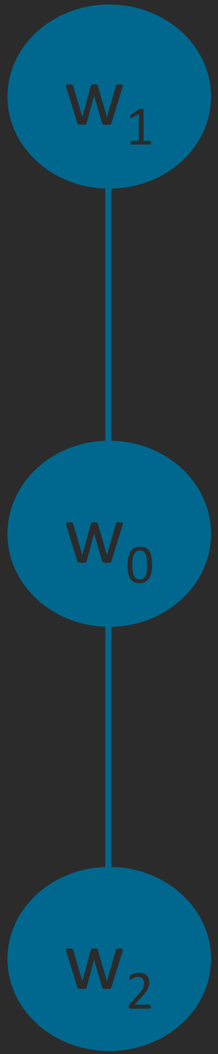
matrix

3) E.F. Combarro, i. Díaz and p. miranda. “on random generation of fu

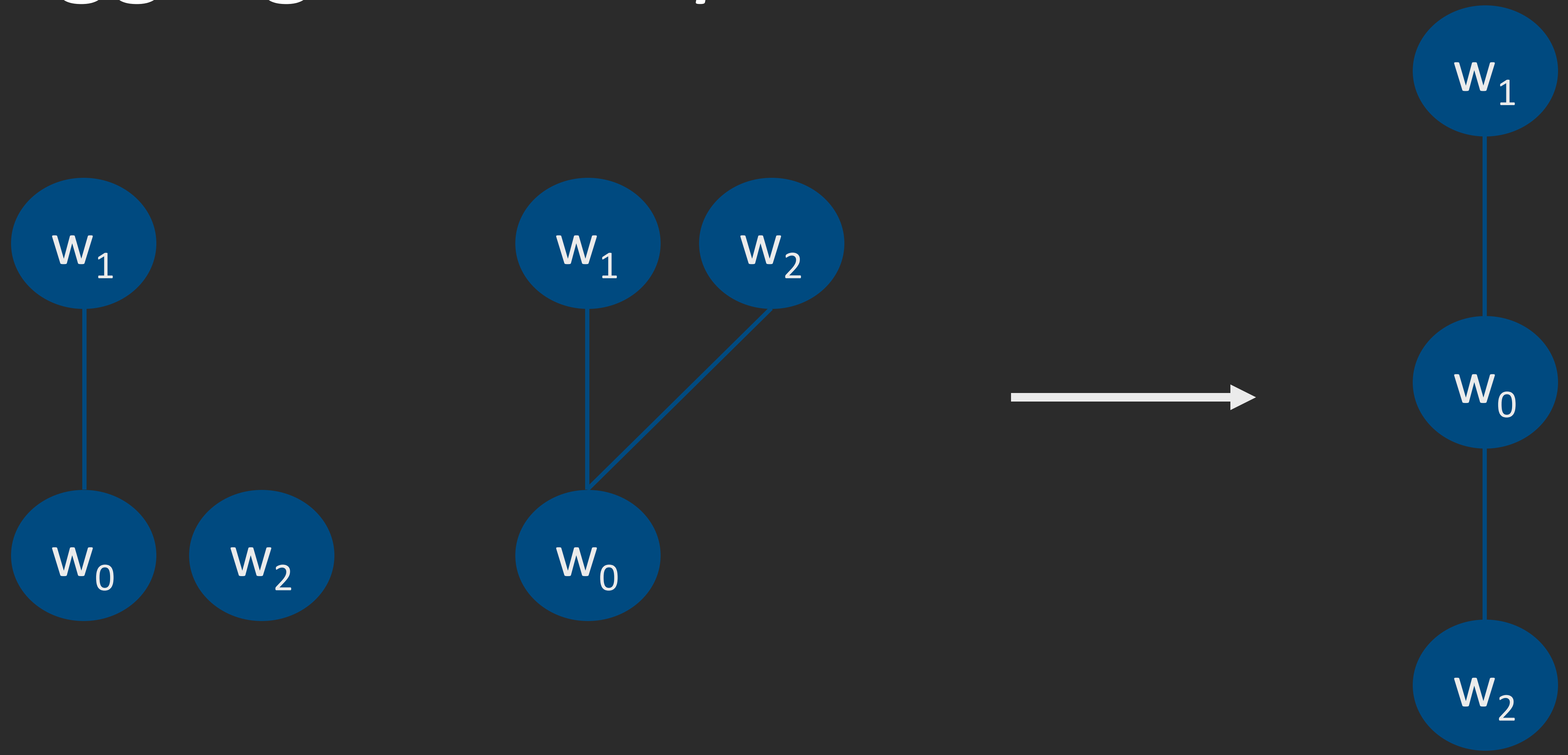
Linear extension

Total order relationship

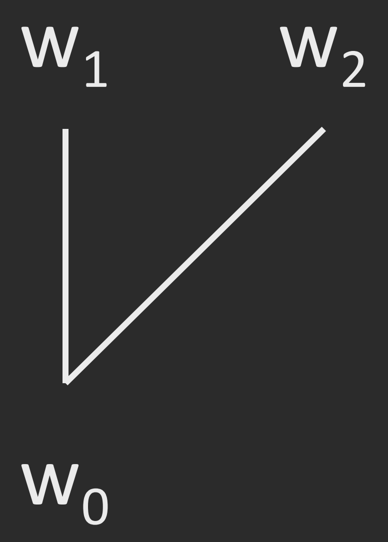
All elements are comparable to each other



Aggregation of *posets*



Aggregation matrix



Aggregation matrix

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Aggregation matrix

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 1 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

Cost of an aggregation

Not optimal

Partial restrictions violated by the linear extension

$$\begin{pmatrix} 2 & 2 & 1 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

w_0

w_1

w_2

Cost =

Cost of an aggregation

Not optimal

Partial restrictions violated by the linear extension

$$\begin{pmatrix} 2 & 2 & 1 \\ 0 & 2 & \mathbf{0} \\ 0 & 0 & 2 \end{pmatrix}$$

w_0

$\mathbf{w}_1$

$\mathbf{w}_2$

Cost = 0

Cost of an aggregation

Not optimal

Partial restrictions violated by the linear extension

$$\begin{pmatrix} 2 & 2 & \mathbf{1} \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

$\mathbf{w_0}$

w_1

$\mathbf{w_2}$

Cost = 0 + 1

Cost of an aggregation

Not optimal

Partial restrictions violated by the linear extension

$$\begin{pmatrix} 2 & \mathbf{2} & 1 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

$\mathbf{w_0}$

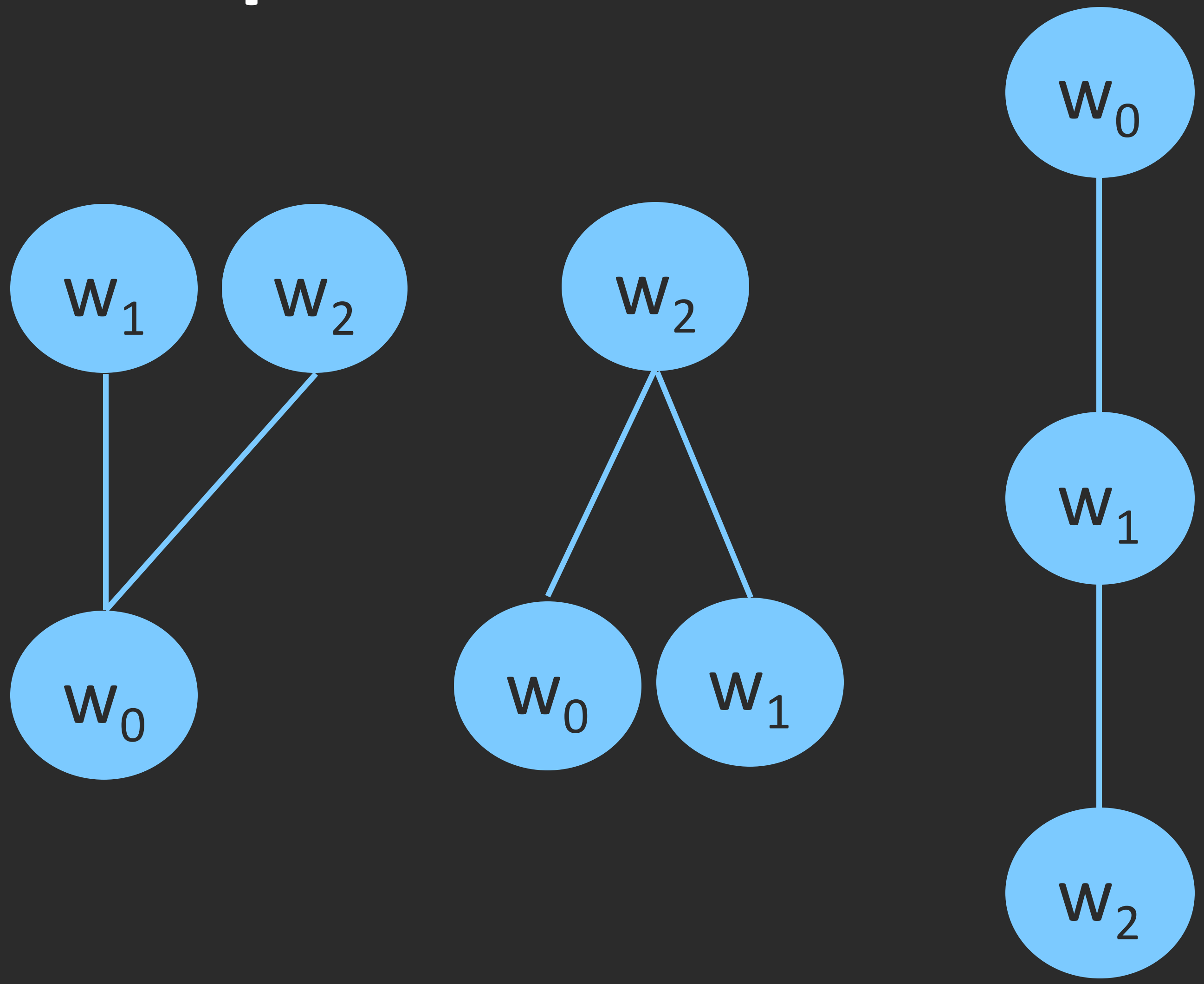
$\mathbf{w_1}$

w_2

Cost = 0 + 1 + 2 = 3

algorithms

example



$$A = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

Mincost st

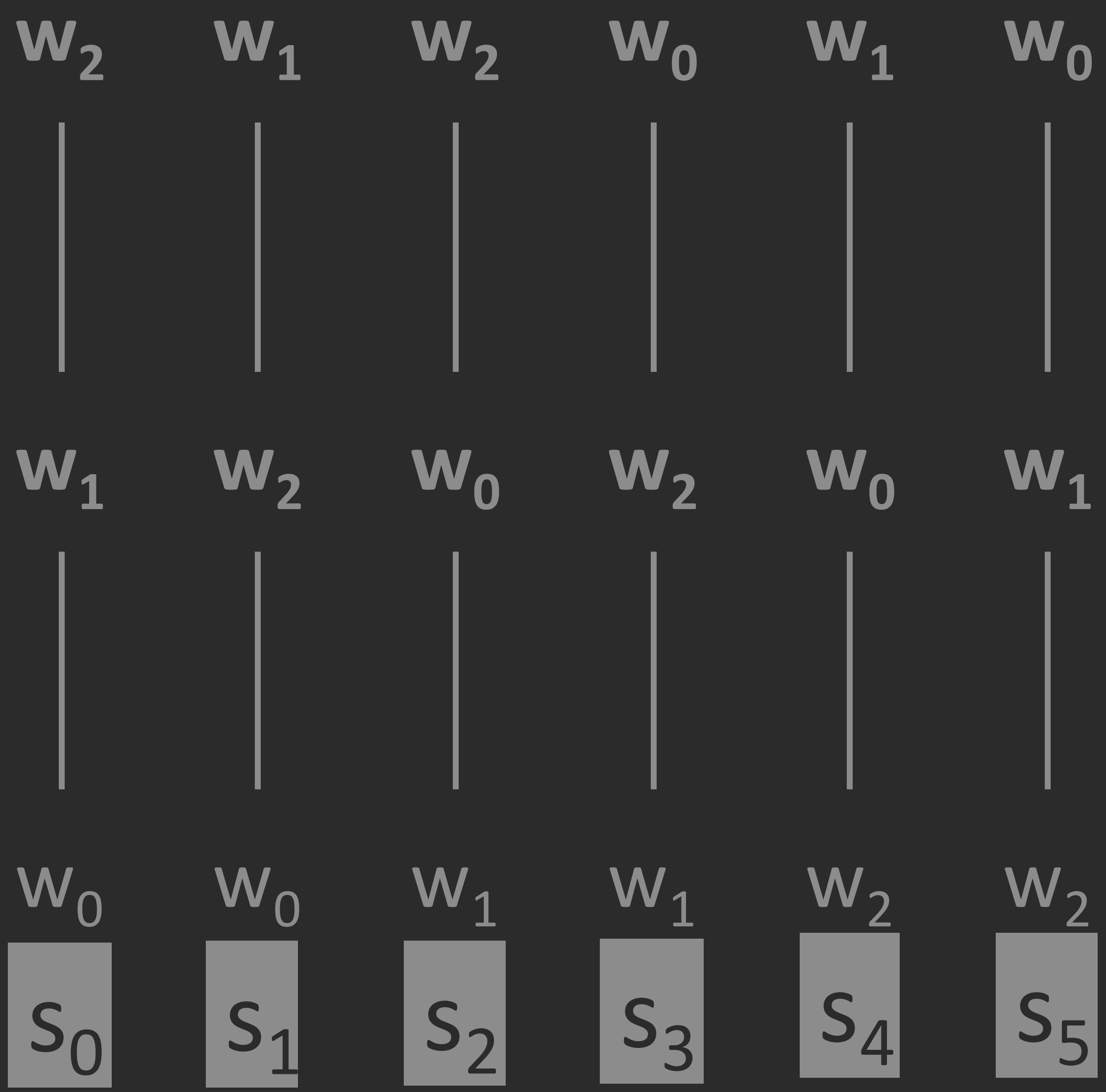
Computing all the posible linear extensions and

Sequentially calculates the cost

Optimal algorithm

High execution times

Mincost st - example



Possible solution	Cost
S_0	3
S_1	3
S_2	4
S_3	3
S_4	4
S_5	4

minimals[4]

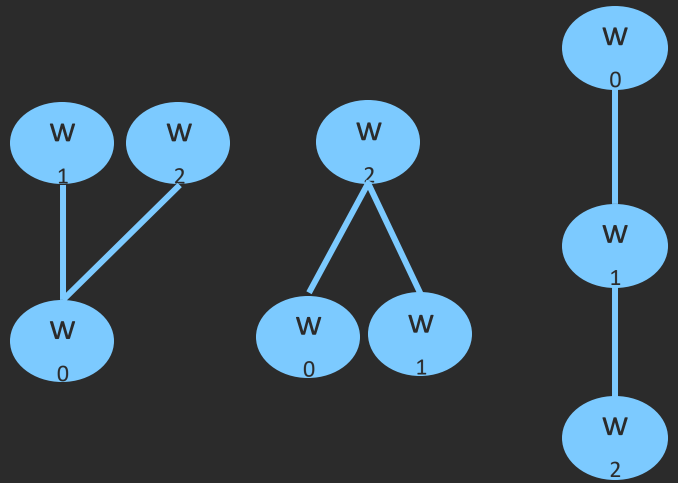
What is a *minimal* ele

An element $a \in P$ is a *minimal* element if there is no $b \in P$

4) e.f. combarro, j.h. de Saracho and i.d. rodríguez. “minimals plus: an improved...” (2019)

Minimals - initialization

Vector up → Vector down → Bound constant → Used vector

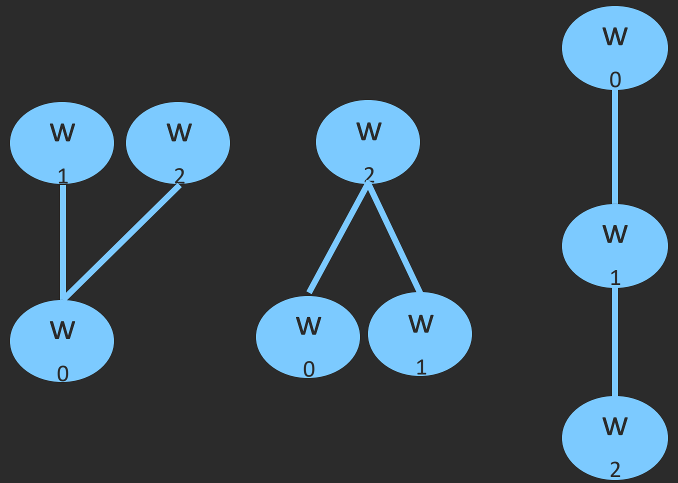


$$A = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

Minimals - initialization

Vector up → Vector down → Bound constant → Used vector

$$up[i] = \sum_{j}^{n} A[i,j] \quad up = [6,5,5]$$

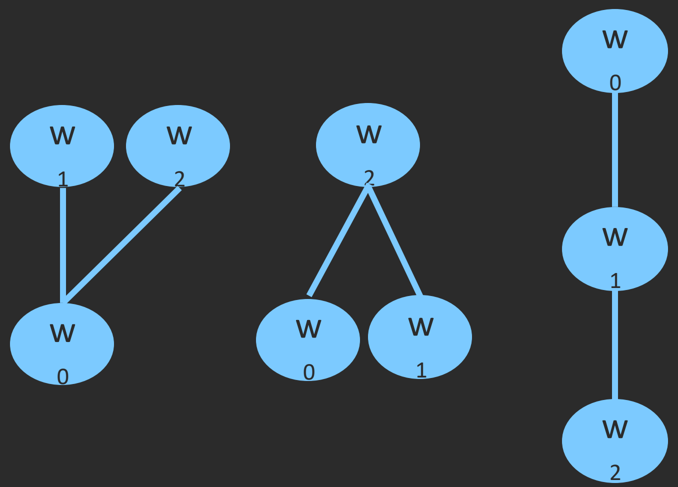


$$A = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

Minimals - initialization

Vector up → Vector down → Bound constant → Used vector

$$\text{down}[i] = \sum_j^n A[j,i] \quad \text{down} = [5, 5, 6]$$

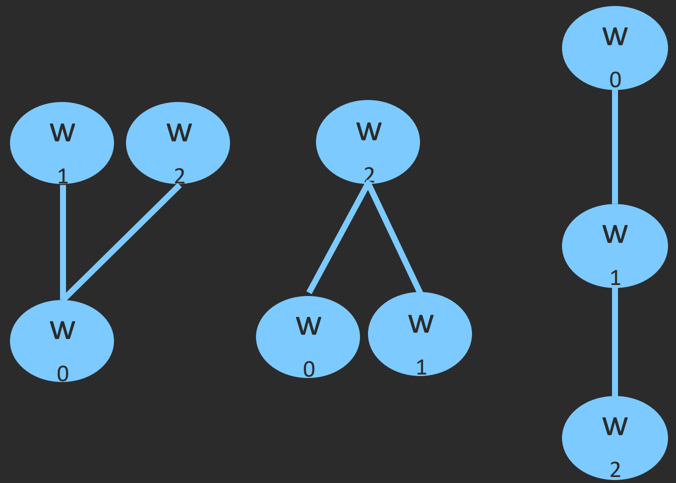


$$A = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

Minimals - initialization

Vector up → Vector down → **Bound constant** → Used vector

$$bound = \sum_{j}^{n} up[i] \qquad bound = 16$$

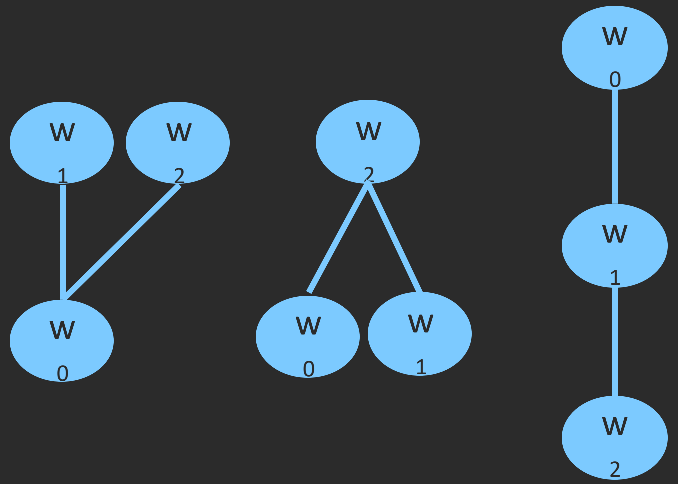


$$A = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

Minimals - initialization

Vector up → Vector down → Bound constant → **Used vector**

$$used = [False, False, False]$$



$$A = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

Minimals – search of the minimals

1º) Lowest numb
Elements below

2º) *minim*

3º) Choose *mi*

4º) upda

Min =

Minimals = [w_0,

probabilities = [$\frac{6}{11}$,

$$P(i) = \frac{up[i]}{\sum_{j\ minimal} up[i]}$$

w_1

$$A = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

	w_0	**w_1**	**w_2**
up	6	5	5
down	5	5	6
used	False	False	False

Bound =

Minimals – search of the minimals

1º) Lowest numb
Elements below

Min =

2º) *minim*

Minimals = [w_0,

3º) Choose *mir*

probabilities = [$\frac{6}{11}$,

4º) upda

w_1

$$A = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

	w_0	w_1	w_2
up	6-1 = 5	5-3 = 2	5-1 = 4
down	5-1 = 4	5-3 = 2	6-1=5
used	false	true	False

Bound =

Minimals – search of the minimals

1º) Lowest numb
Elements below

Min =

2º) *minim*

Minimals = [

3º) Choose *mi*

probabilities =

4º) upda

$$w_0$$
|
$$w_1$$

$$A = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

	w_0	**w_1**	**w_2**
up	5	2	4
down	4	2	5
used	false	true	False

Bound =

Minimals – search of the minimals

1º) Lowest numb
Elements below

Min =

2º) *minim*

Minimals = [

3º) Choose *mi*

probabilities =

4º) upda

$$w_0$$
|
$$w_1$$

$$A = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

	$\mathbf{w_0}$	$\mathbf{w_1}$	$\mathbf{w_2}$
up	5-3 = 2	2-1 = 1	4-1 = 3
down	4-3 = 1	2-1 = 1	5-2 = 3
used	true	true	False

Bound =

Minimals – search of the minimals

1º) Lowest numb
Elements below

Min =

2º) *minim*

Minimals = [

3º) Choose *mi*

probabilities =

4º) upda

w_2
|
w_0
|
w_1

$$A = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

	w_0	w_1	w_2
up	2	1	3
down	1	1	3
used	true	true	False

Bound =

Minimals – search of the minimals

1º) Lowest numb
Elements below

2º) *minim*

3º) Choose *mi*

4º) upda

Min =

Minimals = [

probabilities =

w_2
|
w_0
|
w_1

$$A = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

	w_0	w_1	w_2
up	2-2=0	1-1=0	3-3=0
down	1-1=0	1-1=0	3-3=0
used	true	true	true

Bound =

Minimals random

Randomly cho

~~Ve[illegible]or u~~

Minimals random - example

minima

Minimals rand

Minimals = [w$_{0}$

Minimals = [w$_{0}$

probabilities = [$\frac{6}{11}$,

probabilities = [$\frac{1}{2}$

$P(i) = P(i) = \frac{1}{k} \frac{[i]}{up[i]}$

mincost mt

Based on mincost st

parallel[5]

5) e. ouellet and o. Saad “fast implementations and a new indexing…” (2018)

Sorting algorithms

- bubble
- selectio n
- insertio n
- Quickso rt
- merges ort

sorting_method(what_to_order, cor

Sorting algorithms – comparison a

only take into account the times an *i* element is lower or greater than another *i*

$$P(i \leq j) = \begin{cases} \frac{lower}{lower+greater} & if\ lower + greater \neq \\ 0.5 & otherwise \end{cases}$$

lower → A[i, j]

greater → A[j, i]

Sorting algorithms – comparison b

the times an *i* element is lower or greater than

The times objects i and j are not comparable

$$P\,(i \leq j) = \left\{ \frac{lower + 0.5\ x\ notCompared}{total} \right.$$

lower → A[i, j] greater → A[j, i]

total → A[i, i]

Not compared → total – (lower + greater)

Sorting algorithms – example

A =

$$\begin{bmatrix} 3 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{bmatrix}$$

Comparison a

$$\begin{bmatrix} 1 & \frac{A[0,1]}{A[0,1] + A[1,0]} & \frac{A[0,2]}{A[0,2] + A[2,0]} \\ \frac{A[1,0]}{A[1,0] + A[0,1]} & 1 & \frac{A[1,2]}{A[1,2] + A[2,1]} \\ \frac{A[2,0]}{A[2,0] + A[0,2]} & \frac{A[2,1]}{A[2,1] + A[1,2]} & 1 \end{bmatrix}$$

Comparison b

$$\begin{bmatrix} 1 & \frac{A[0,1] + 0.5\ x\ 1}{3} & \frac{A[0,2] + 0.5\ x\ 0}{3} \\ \frac{A[1,0] + 0.5\ x\ 1}{3} & 1 & \frac{A[1,2] + 0.5\ x\ 1}{3} \\ \frac{A[2,0] + 0.5\ x\ 0}{3} & \frac{A[2,1] + 0.5\ x\ 1}{3} & 1 \end{bmatrix}$$

Sorting algorithms – example

$$A = \begin{bmatrix} 3 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{bmatrix}$$

Comparison a

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{2}{3} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 1 \end{bmatrix}$$

Comparison b

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{2}{3} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 1 \end{bmatrix}$$

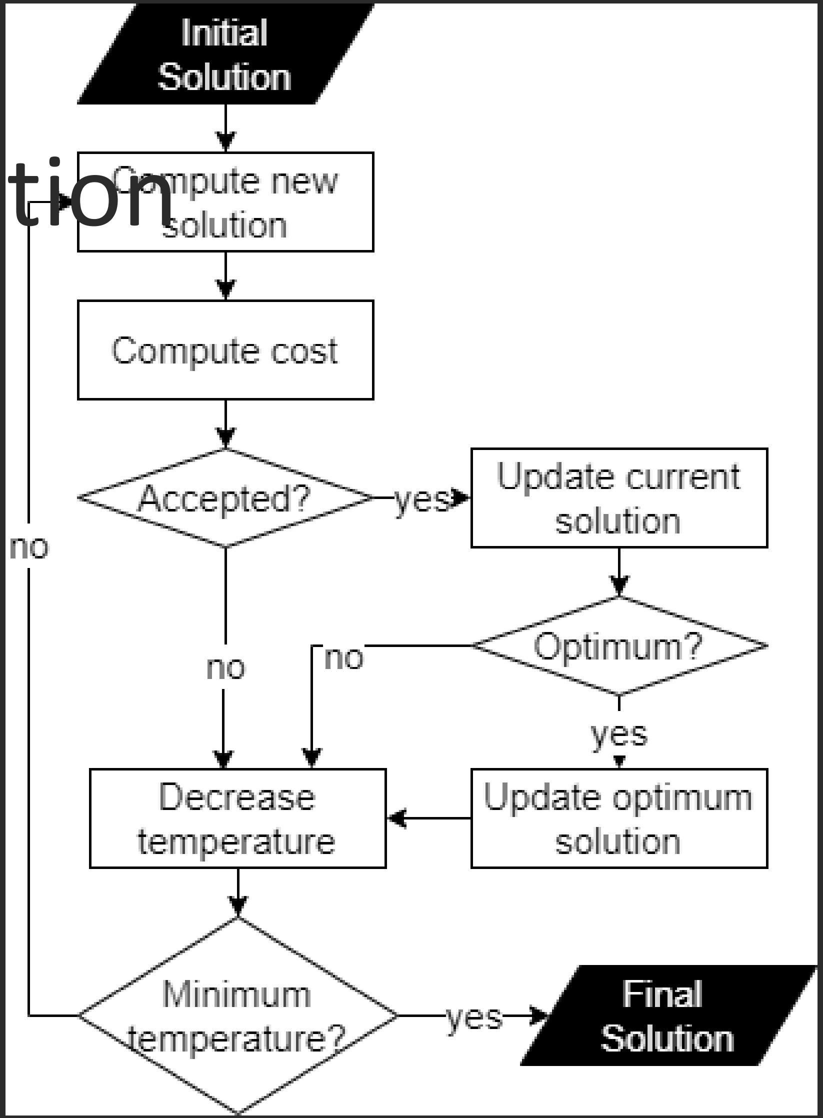
simulated annealing[6]

Optimization algor → Initial solution

6) d.t. pham and d. karaboga "Intelligent optimisation techniques: genetics…" (2000)

simulated annealing - flowchart

Compute new solution



Update optin

Compute c

Lower tempera

Accept and upc

simulated annealing – calculate new so

Swap two positi

[w_0, w_2, v

[w_1, w_2, v

I = random(n)

simulated annealing – calculate cost so

Algorithm of the

simulated annealing – accept and upda

Newcost <= current

Newcost > current

$$P(accepted) = 1$$

$$P^{7}(accepted) = e^{\frac{-\Delta cost}{T}}$$

7) j. dréo et al. “metaheuristics for hard optimization: simulated annealing, …” (2006)

simulated annealing – update best

Keep best solu

simulated annealing – lower temperatu

Initial temperat

Cooling syst

Limit tempera

$$T_{i+1} = \beta\, T_i$$

linear programming – 4 concepts[6]

Decision variables

- Vector **x**

Domain

- **X ≥ 0**

Constraints

- **Ax ≤ b**

Objective function

- $\mathbf{c}^T\mathbf{x}$

6) A. Vidhya. “Introductory guide on linear programming” (2020)

linear programming – standard form[7]

$$\max\{\mathbf{c}^\intercal \mid \mathbf{A}\mathbf{x} \leq \mathbf{b} \wedge \mathbf{x} \geq 0\}$$

$$\min\{\mathbf{c}^\intercal \mid \mathbf{A}\mathbf{x} \leq \mathbf{b} \wedge \mathbf{x} \geq 0\}$$

7) o.a. Camarena. “linear programming” (2018)

linear programming – example

variables

3 objects

V matrix n x n size

$$V = \begin{bmatrix} V_{o,o} & V_{o,1} & V_{o,2} \\ V_{1,0} & V_{1,1} & V_{1,2} \\ V_{2,0} & V_{2,1} & V_{2,2} \end{bmatrix}$$

linear programming – example

domain

binary

zero-one linear programming

$$D = \{0, 1\}$$

linear programming – example

constraints

Constraint	Type
V[0,0] = 1	Diagonal
V[1,1] = 1	Diagonal
V[2,2] = 1	Diagonal
V[0,1] + V[1,0] = 1	No cycles
V[0,2] + V[2,0] = 1	No cycles
V[2,1] + V[1,2] = 1	No cycles
V[0,1] + V[1,2] - V[0,2] ≤ 1	Transitivity
V[0,2] + V[2,1] - V[0,1] ≤ 1	Transitivity
V[1,0] + V[0,2] - V[1,2] ≤ 1	Transitivity
V[1,2] + V[2,0] - V[1,0] ≤ 1	Transitivity

linear programming – example

Objective function

Cost of the aggregation

minimise

Variable x partial cost

$$objectiveFunction(V, A) = \sum_{i,j}^{n} V[i,j] x\, A[j,i] = V[0,1] x\, A[1,0] + V[0,2] x\, A[2,0] + \ \ldots + V[2,1] x\, A[1,2]$$

$$\forall i \neq j$$

experiments

parameters

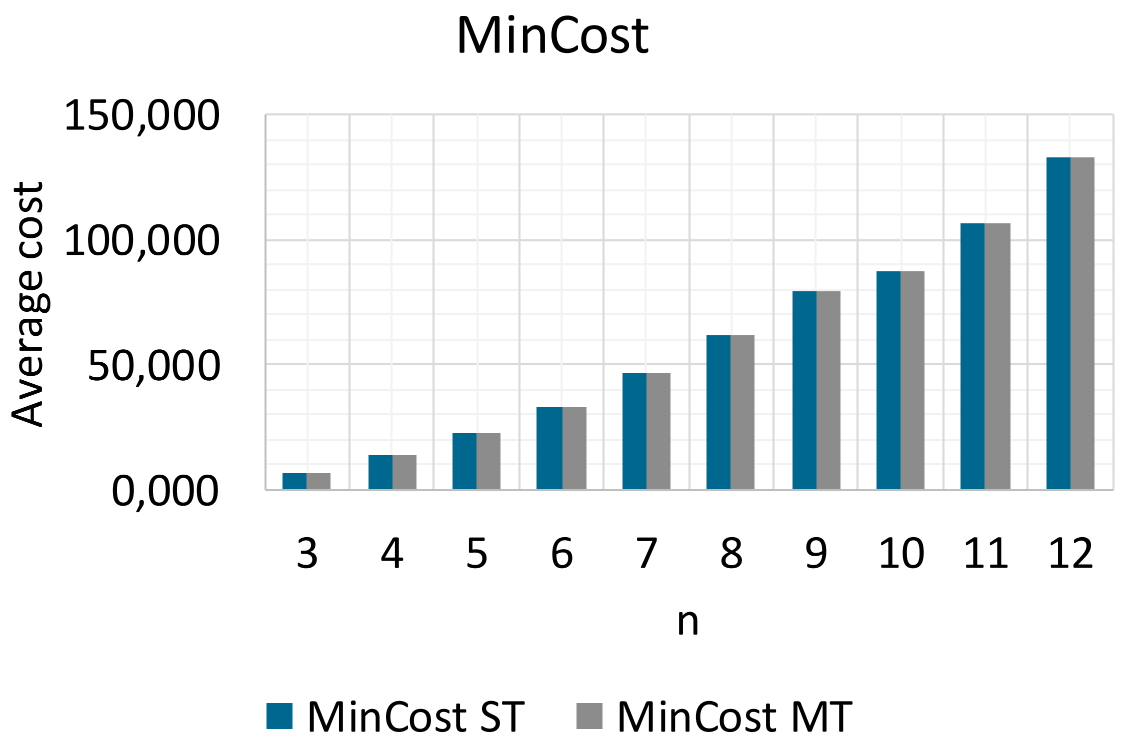
results

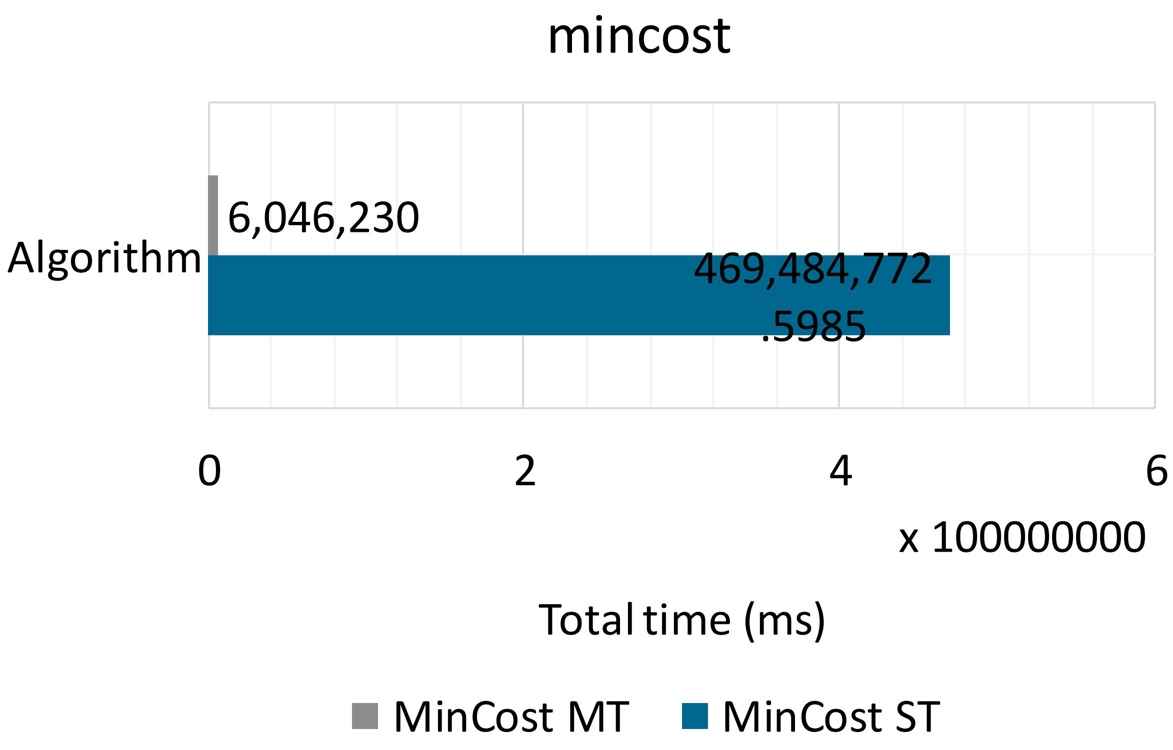
Size of the pos

Amount of pos

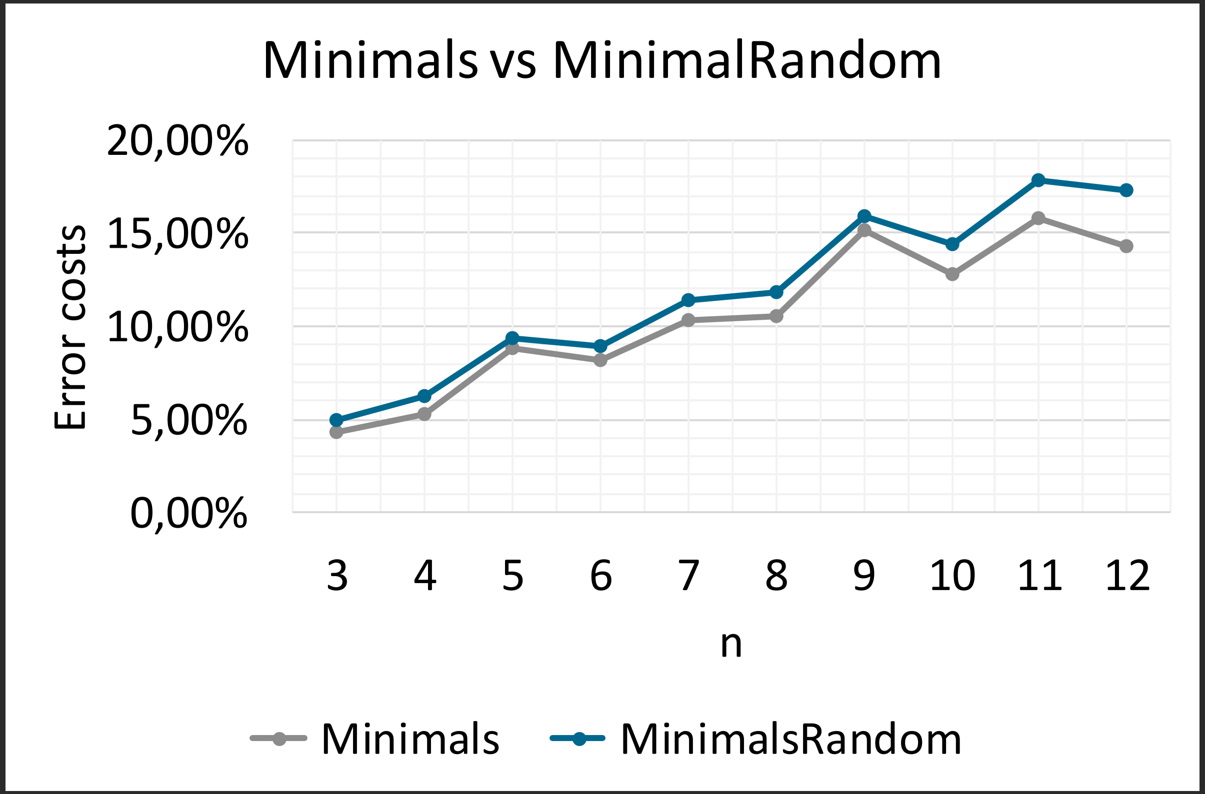
averag

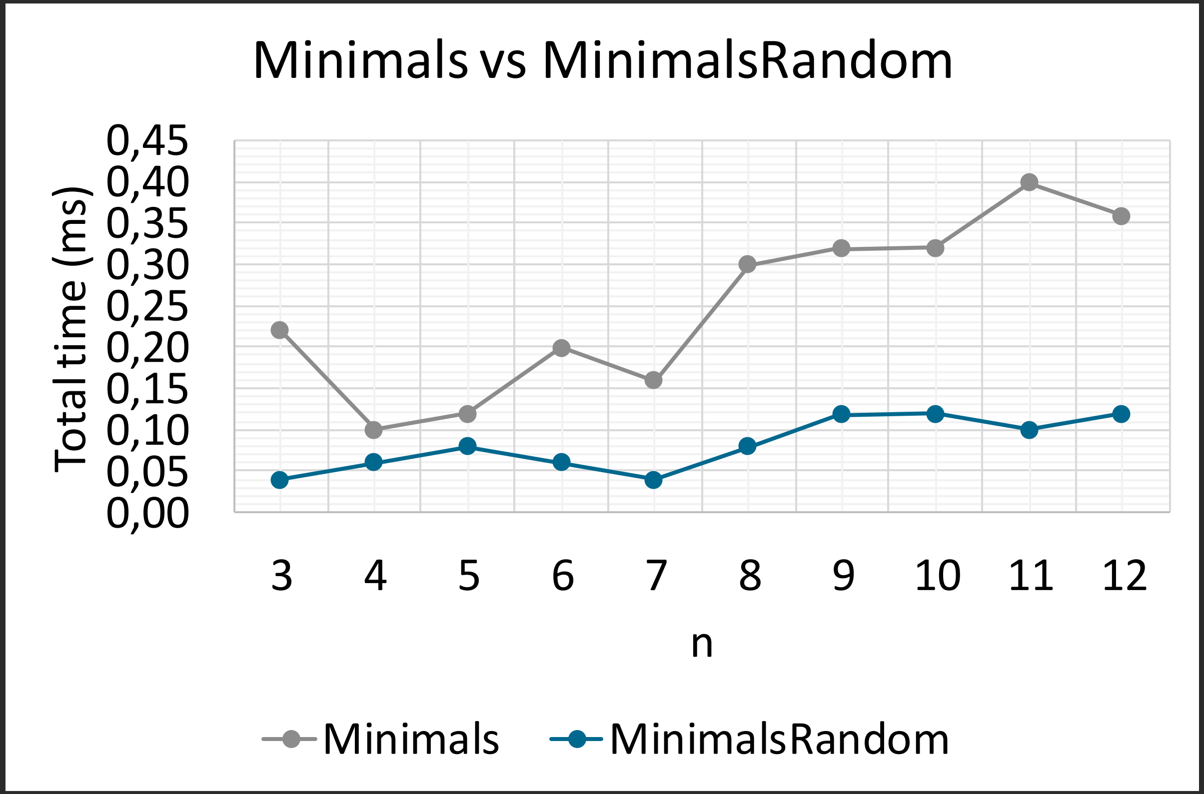
Mincost st vs mincost mt



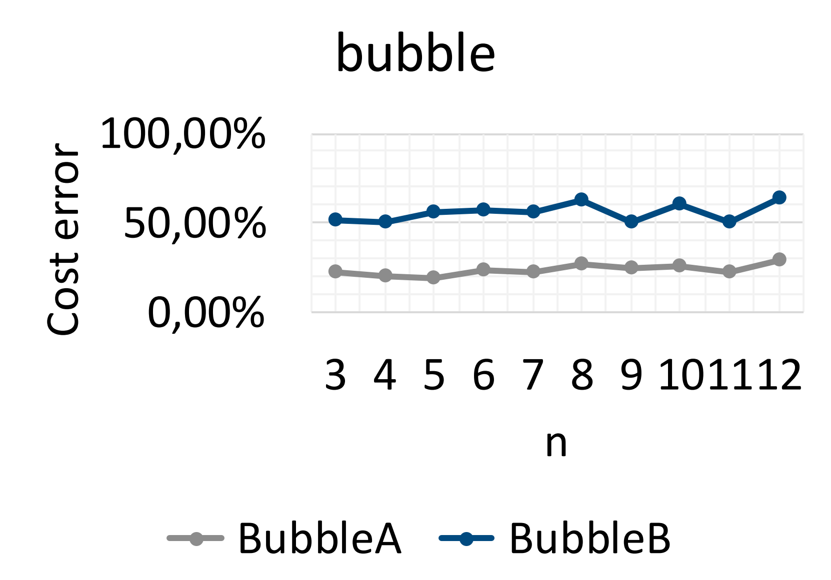


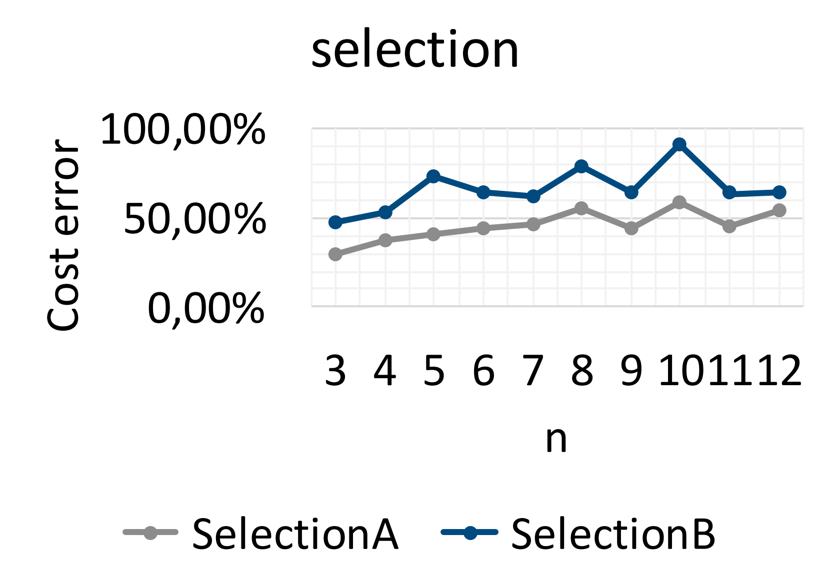
Minimals vs minimals random

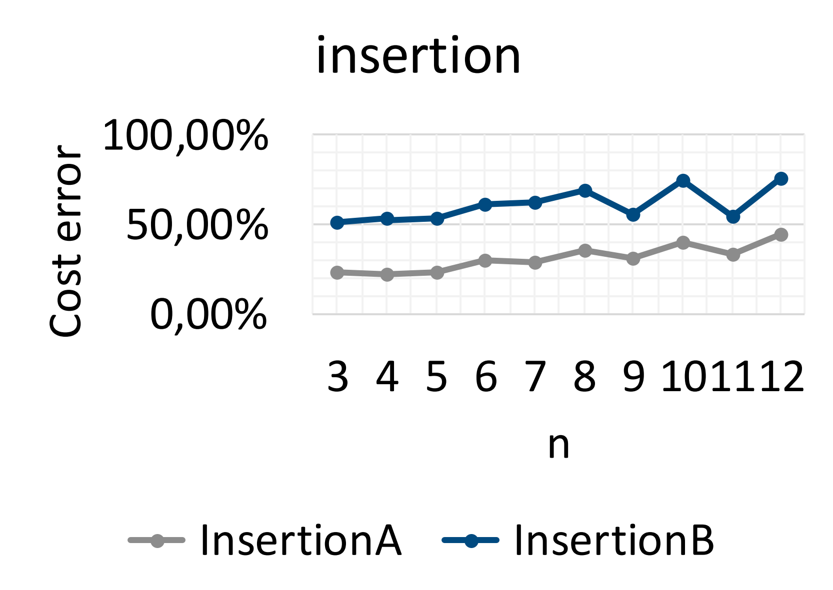


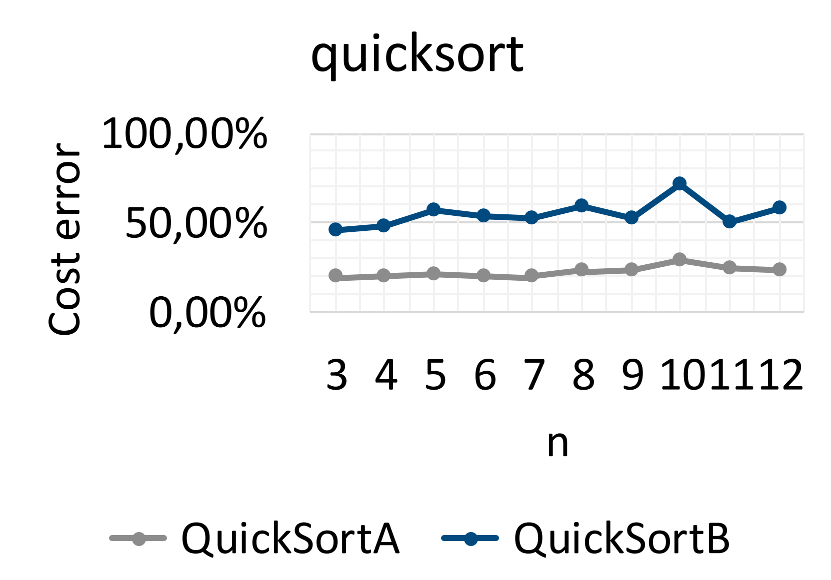


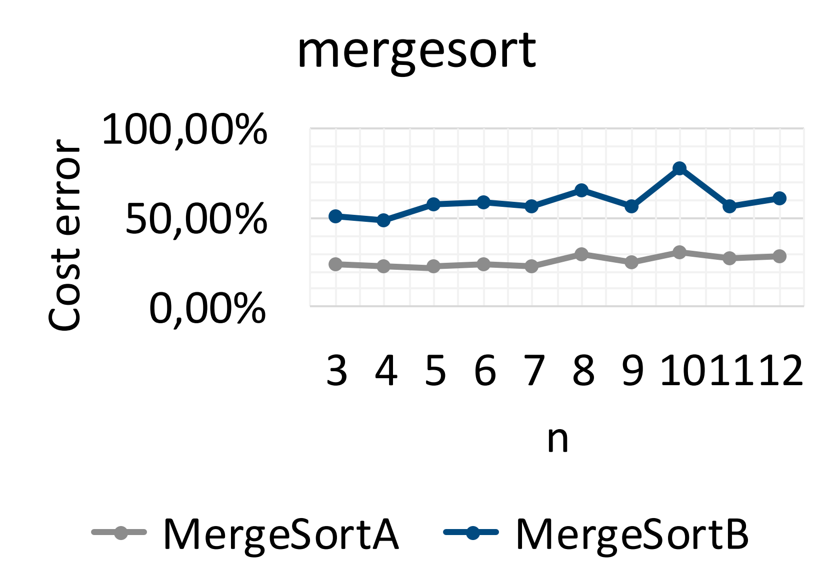
Sorting algorithms: comparison A vs cor



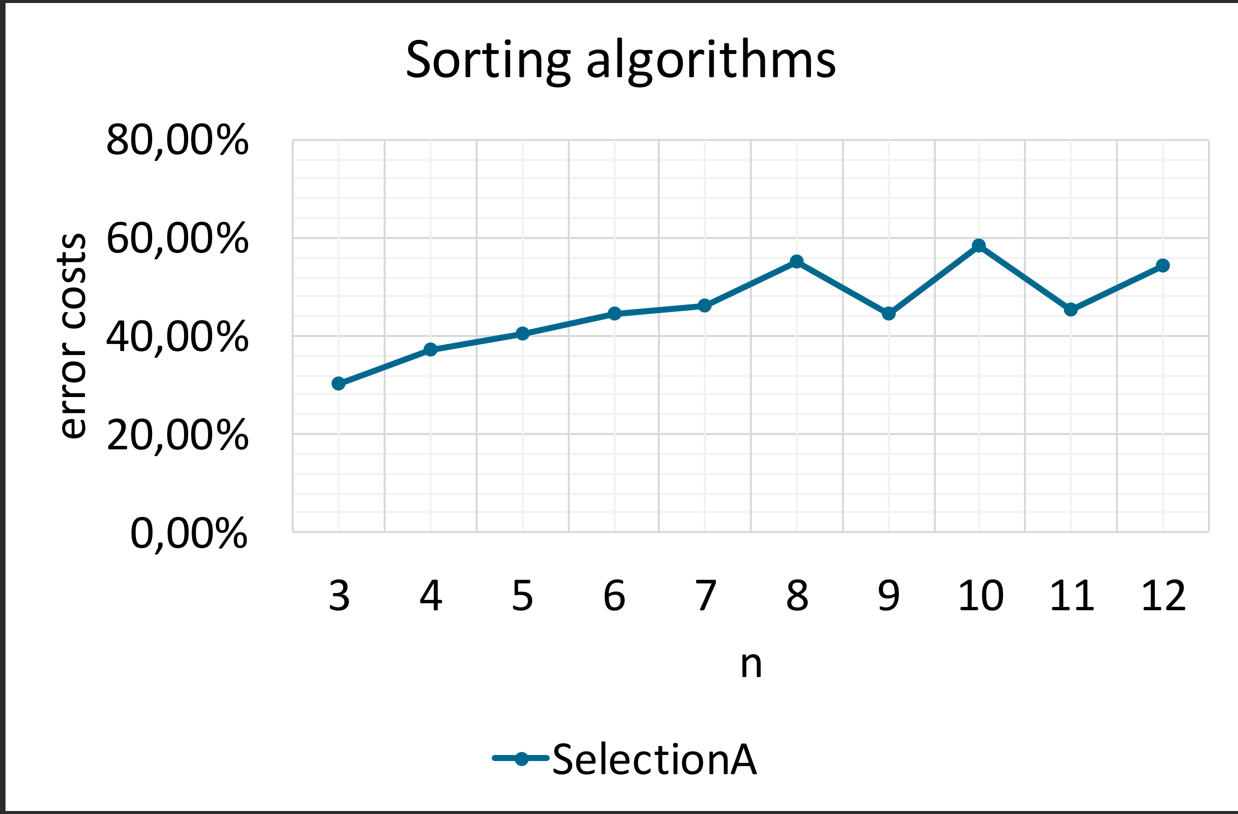


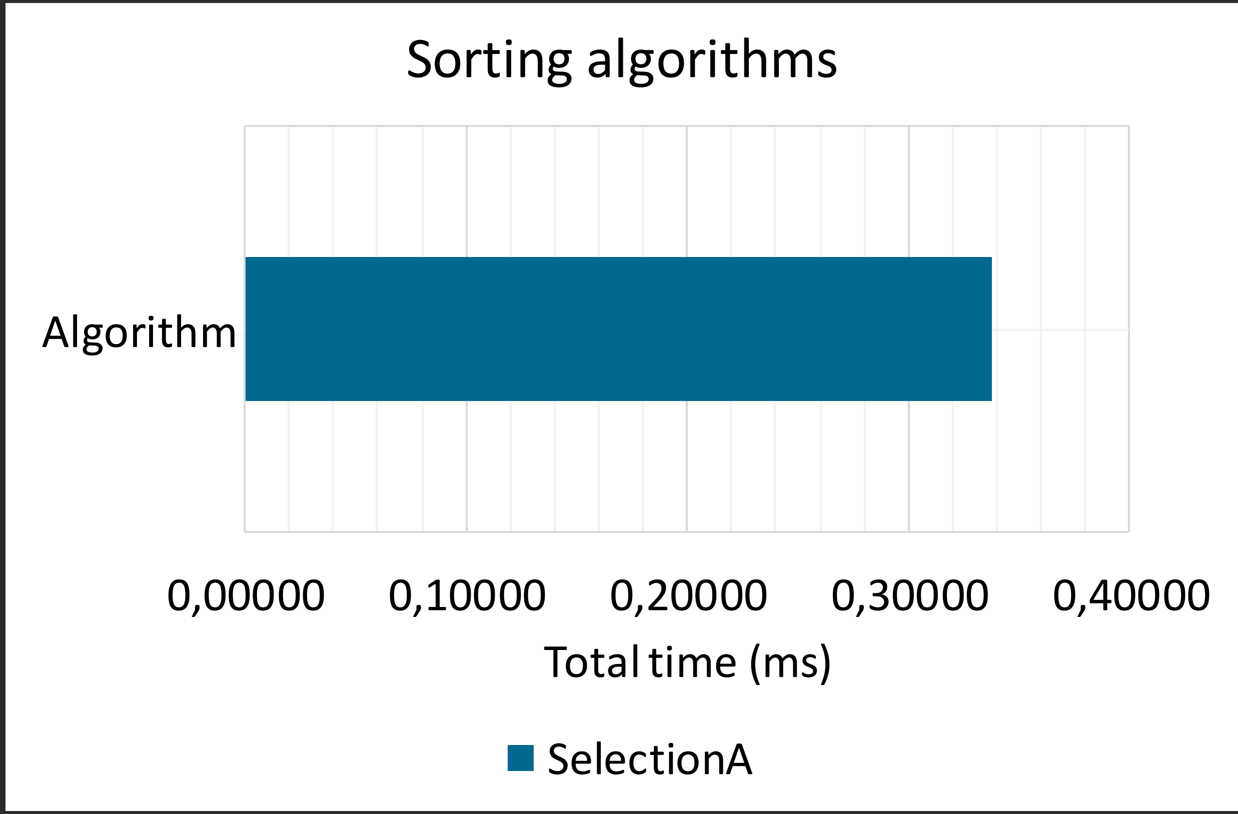






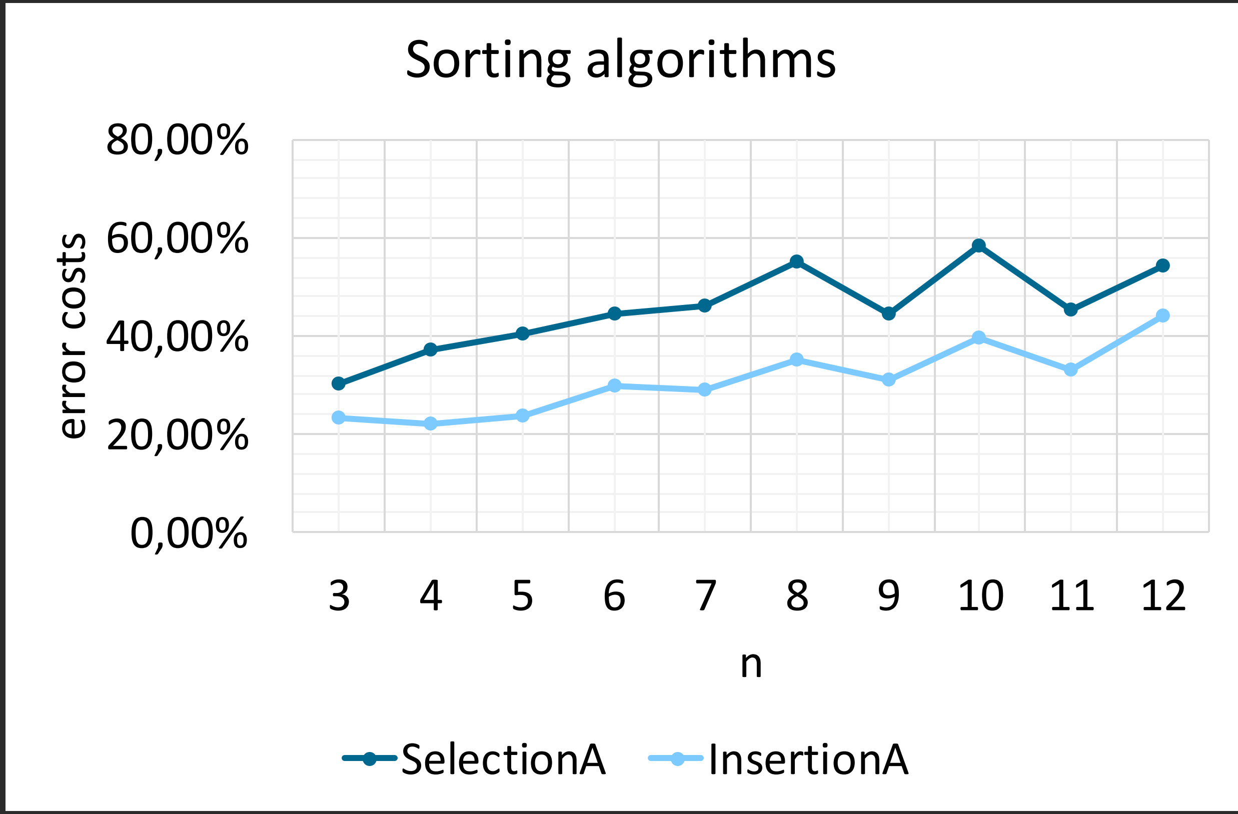
Sorting algorithms: general

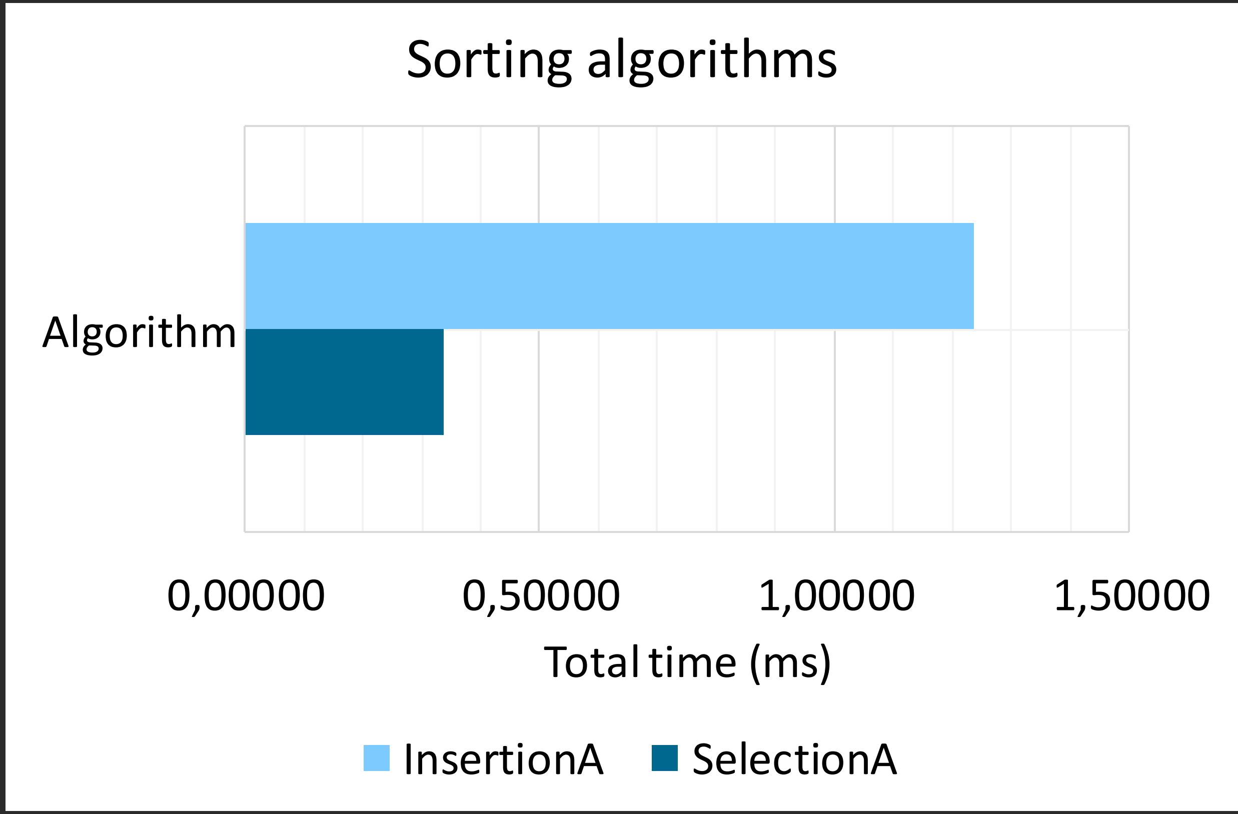


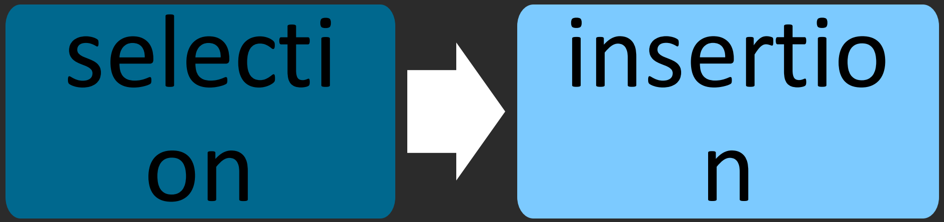


selection

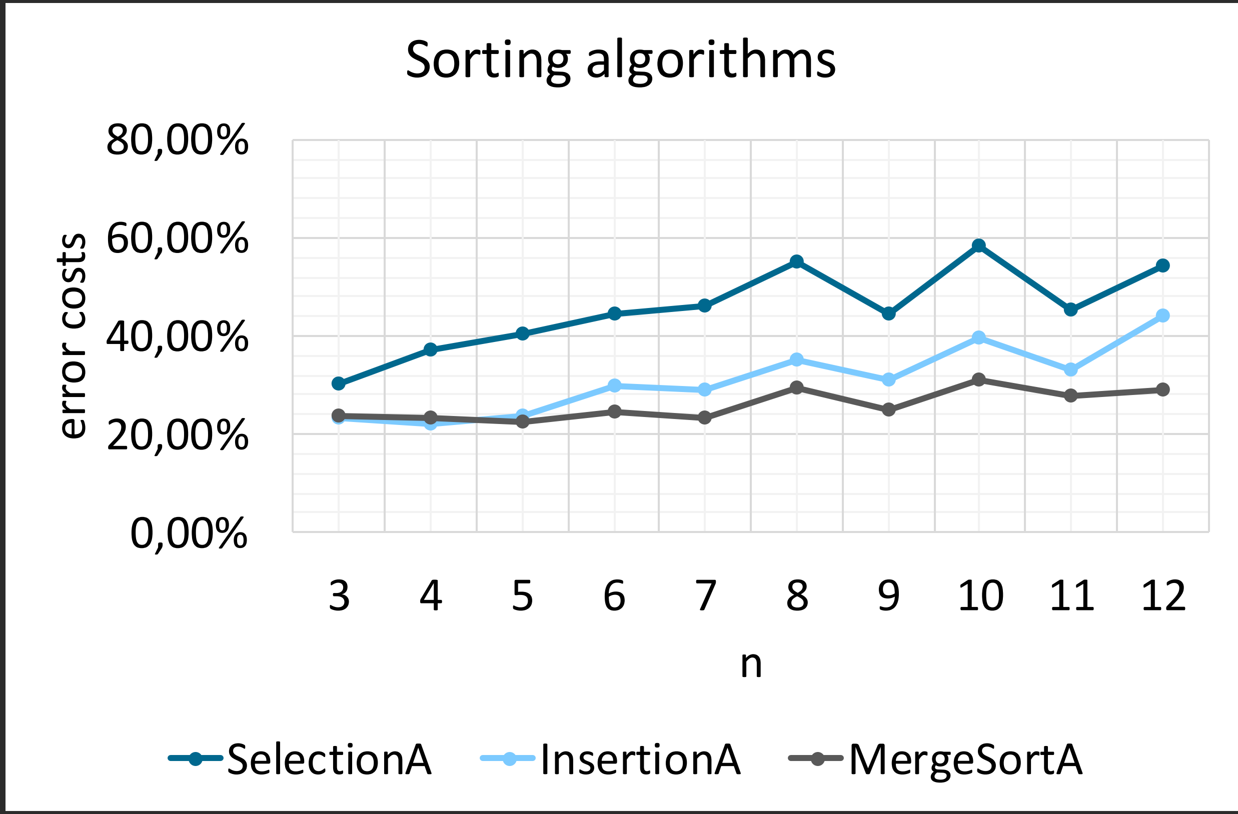
Sorting algorithms: general

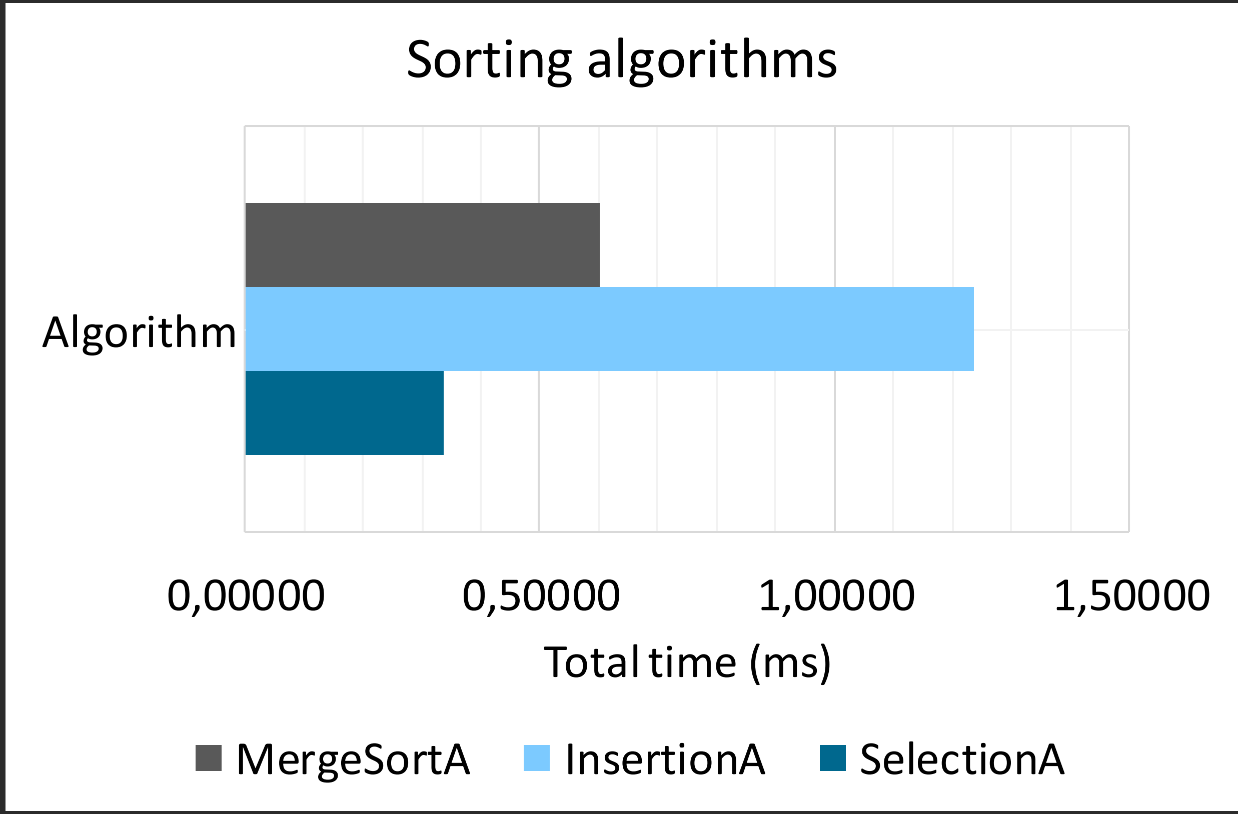






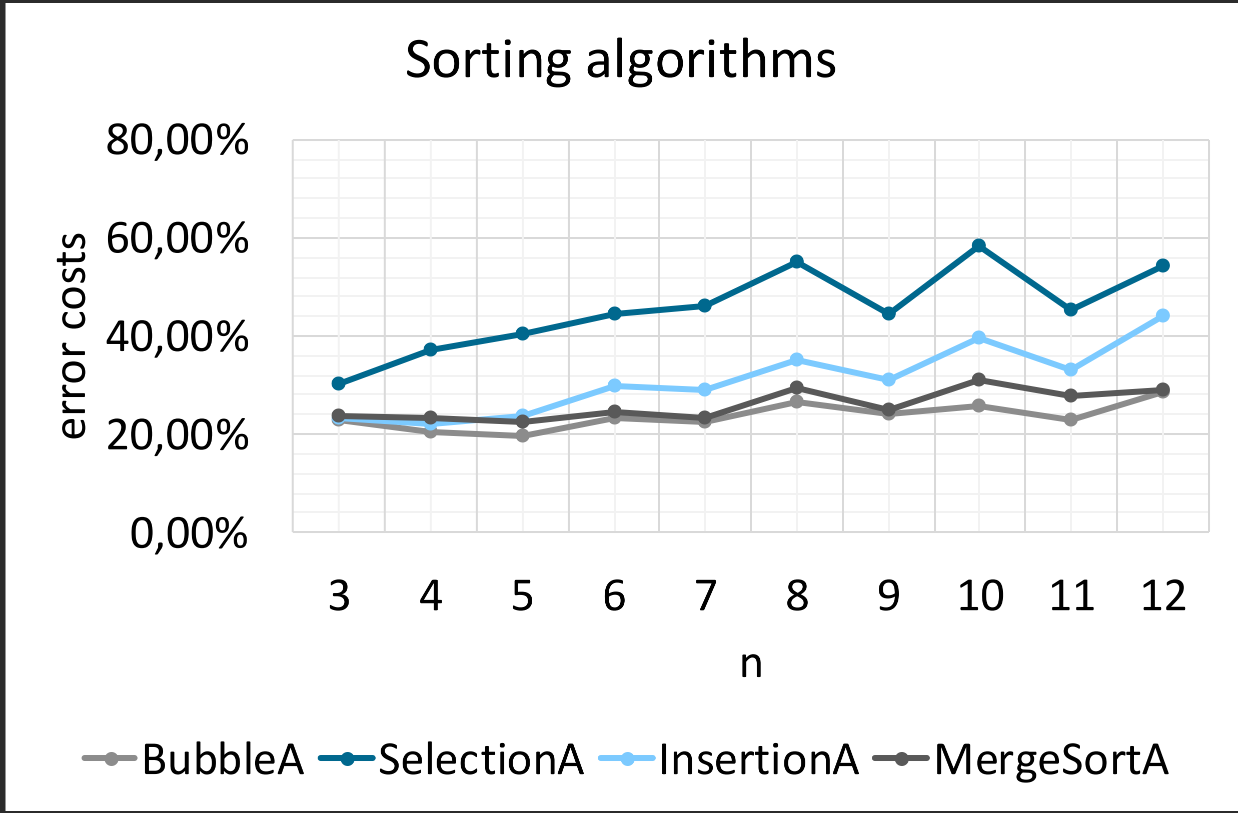
Sorting algorithms: general

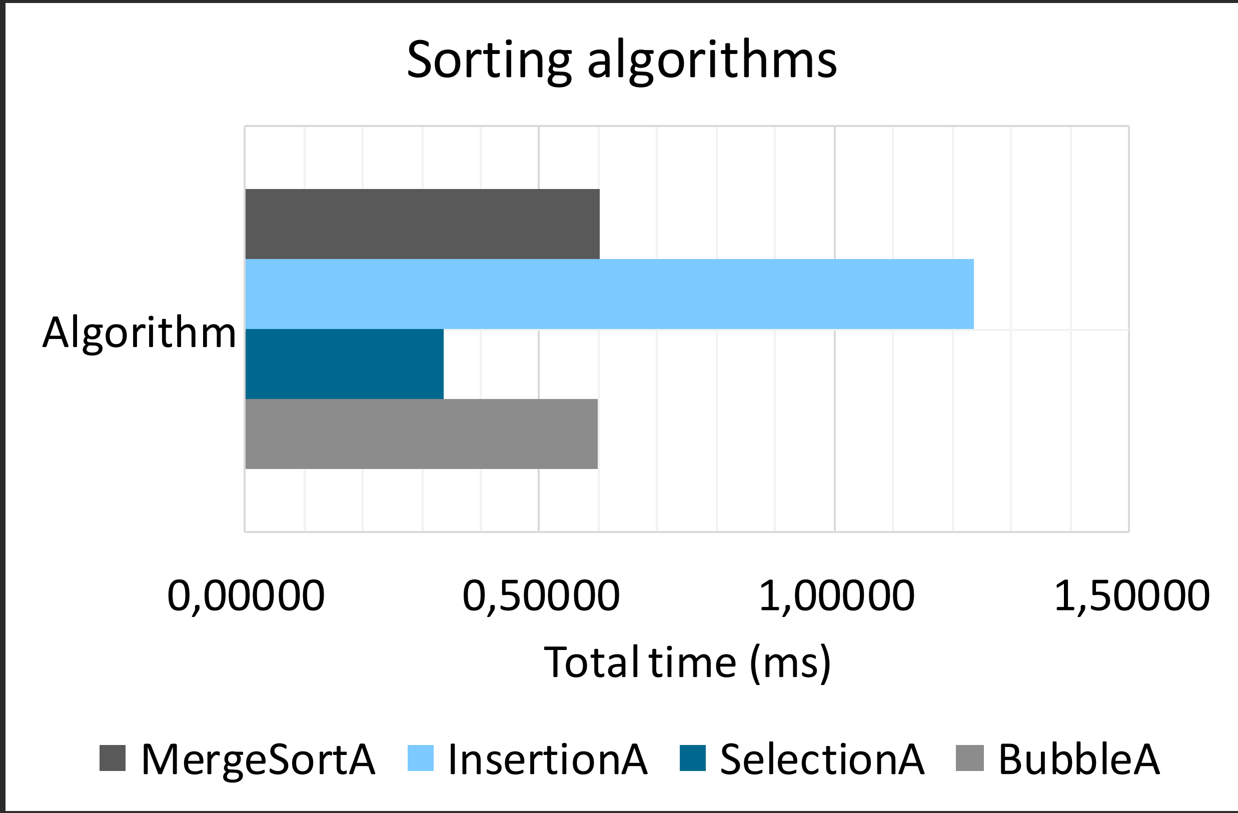


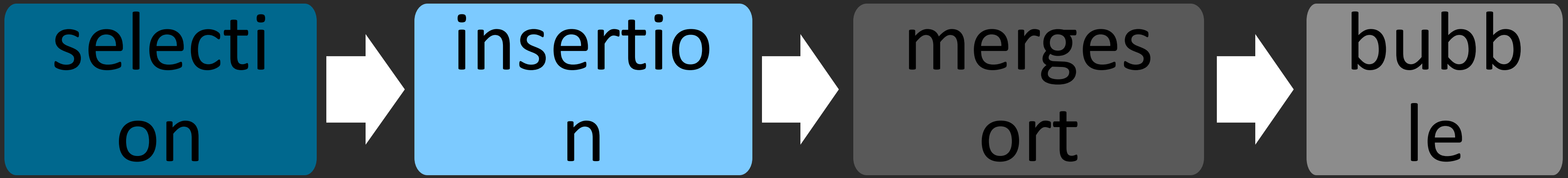


selecti on → insertio n → merges ort

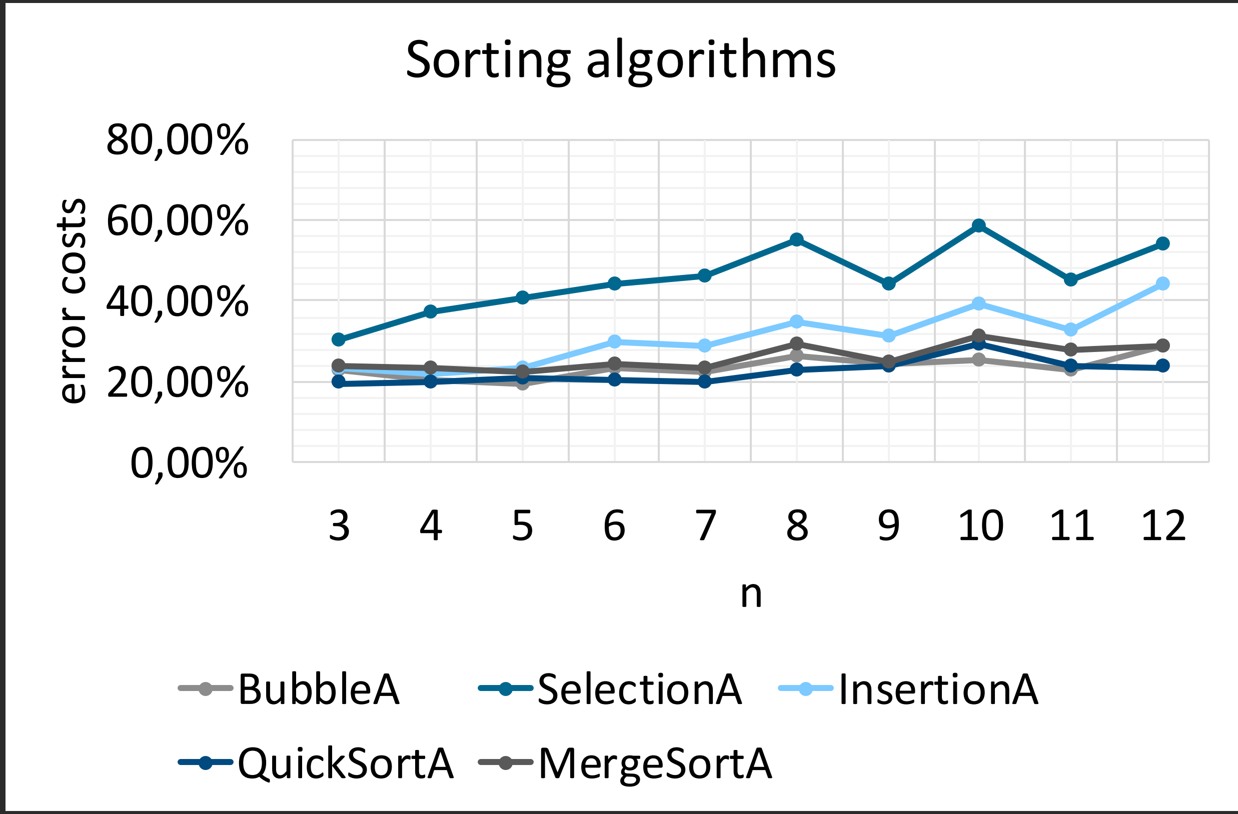
Sorting algorithms: general

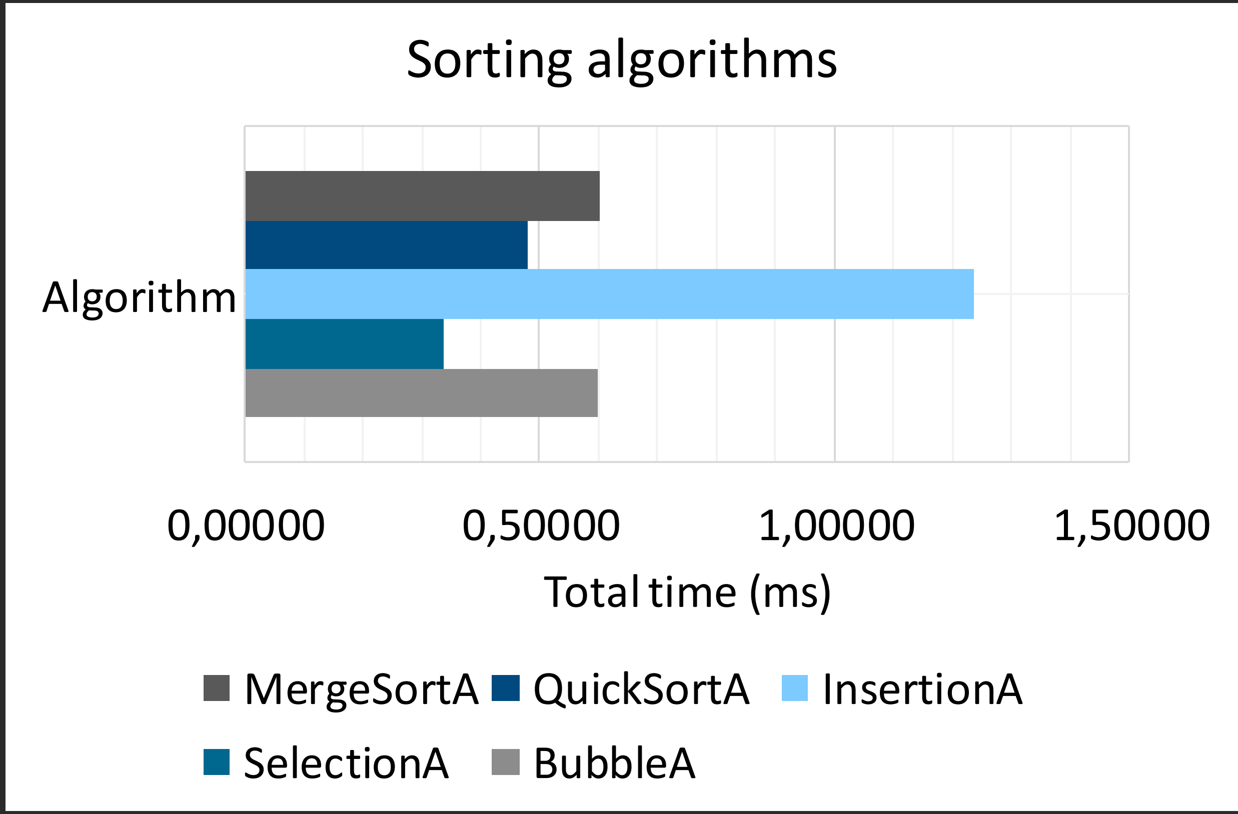






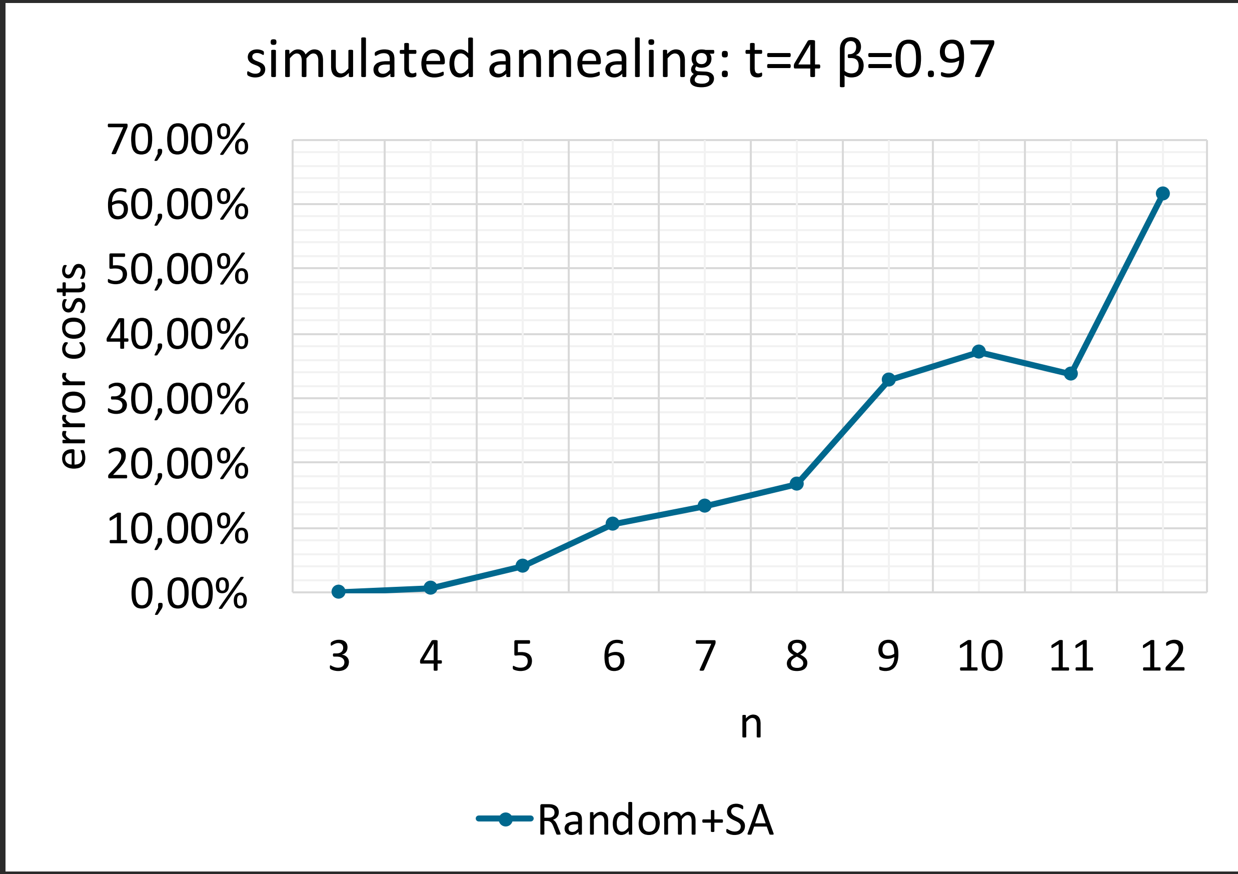
Sorting algorithms: general

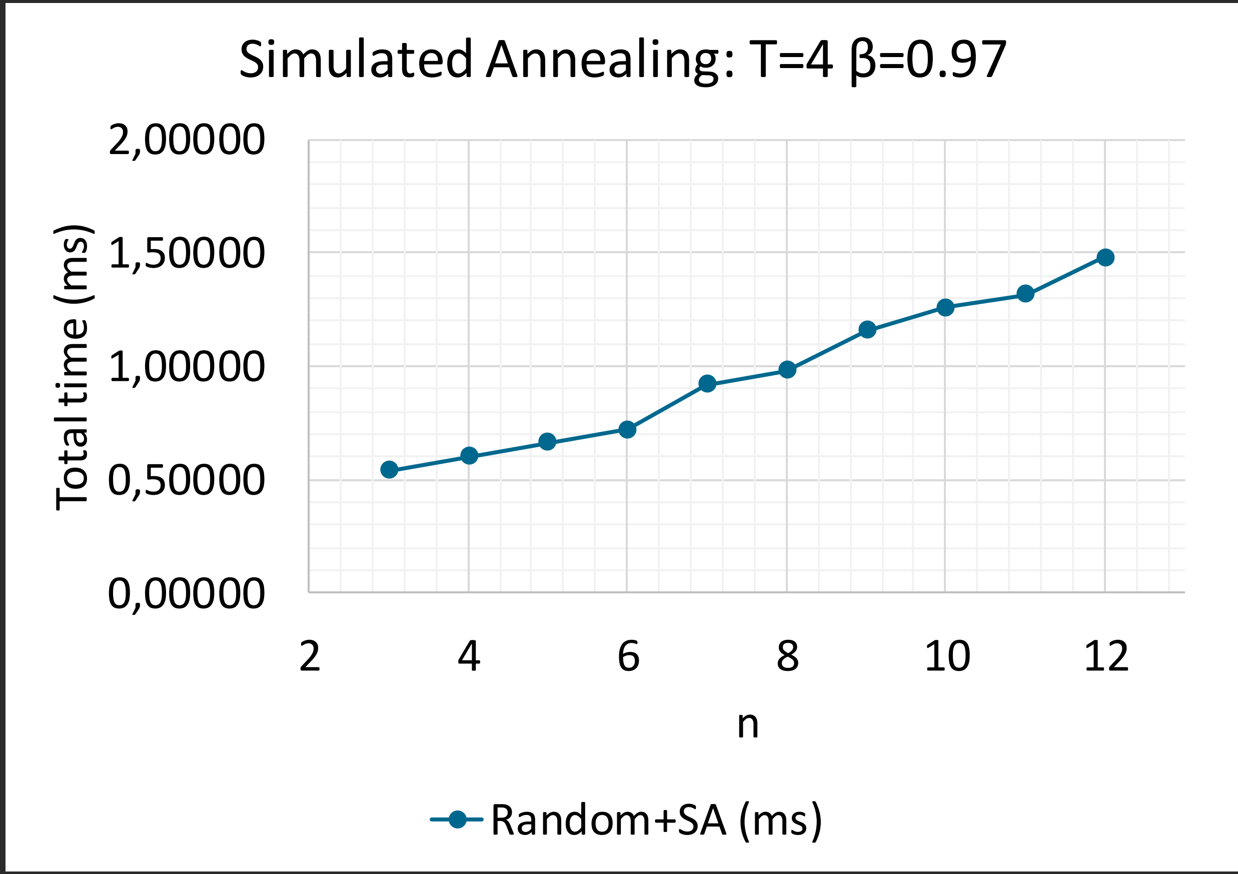




selection → insertion → mergesort → bubble → quicksort

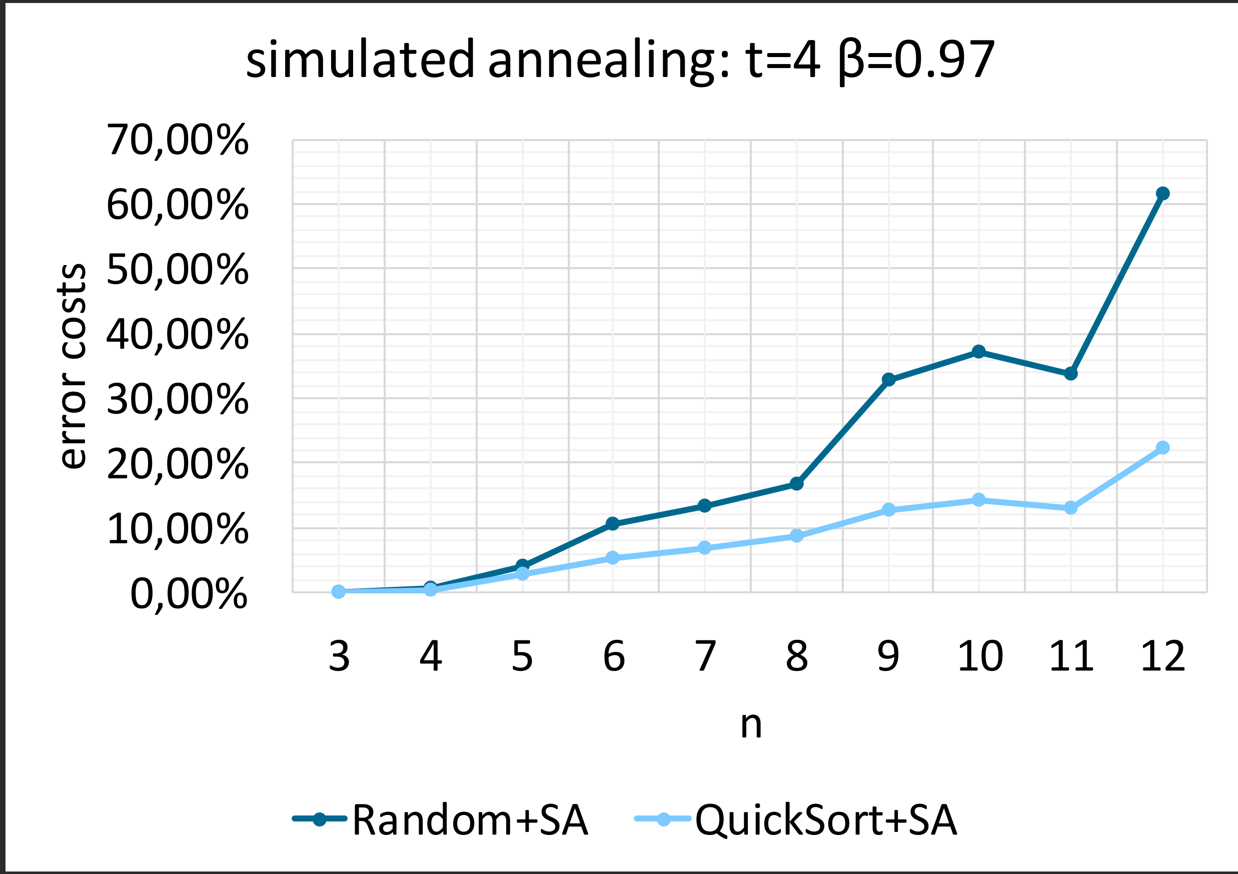
Simulated annealing: initial algorithm

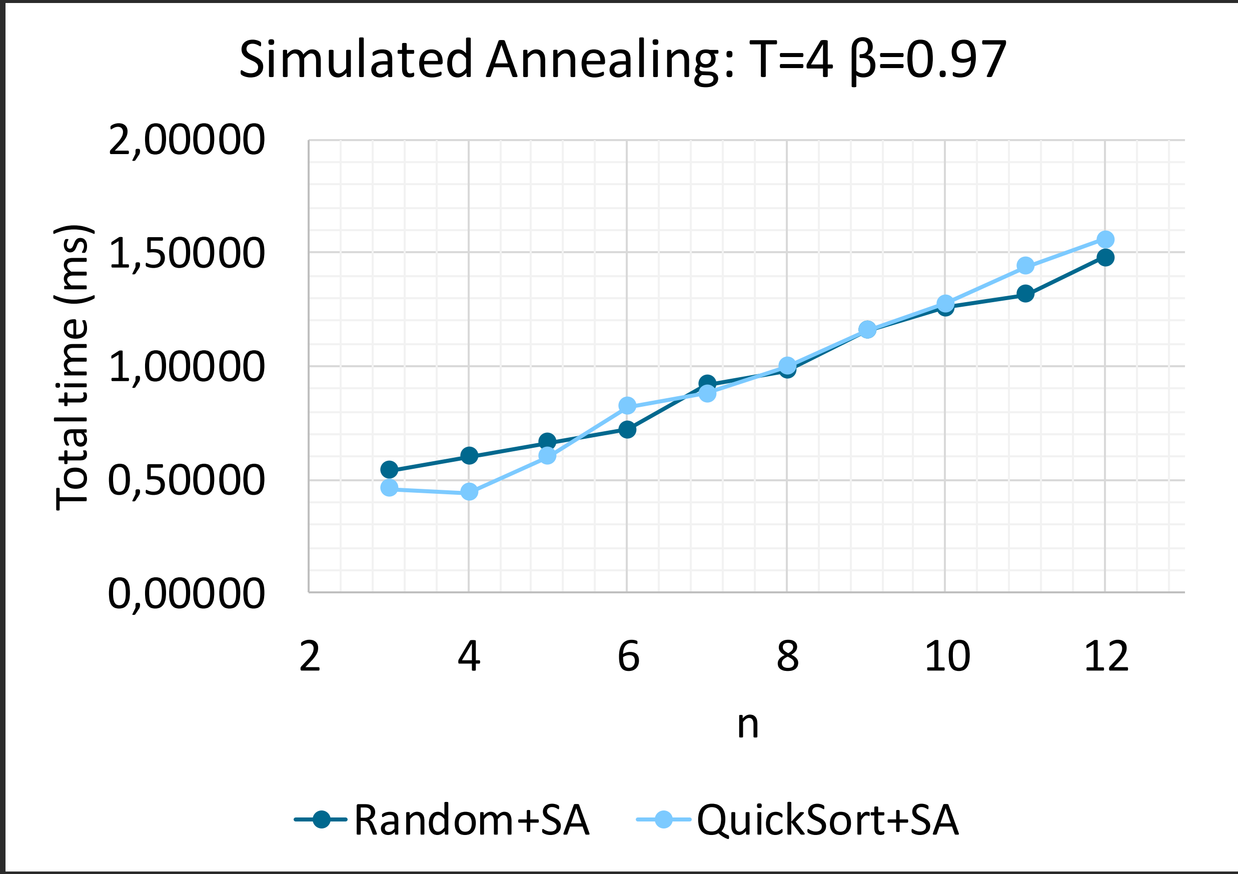




Random

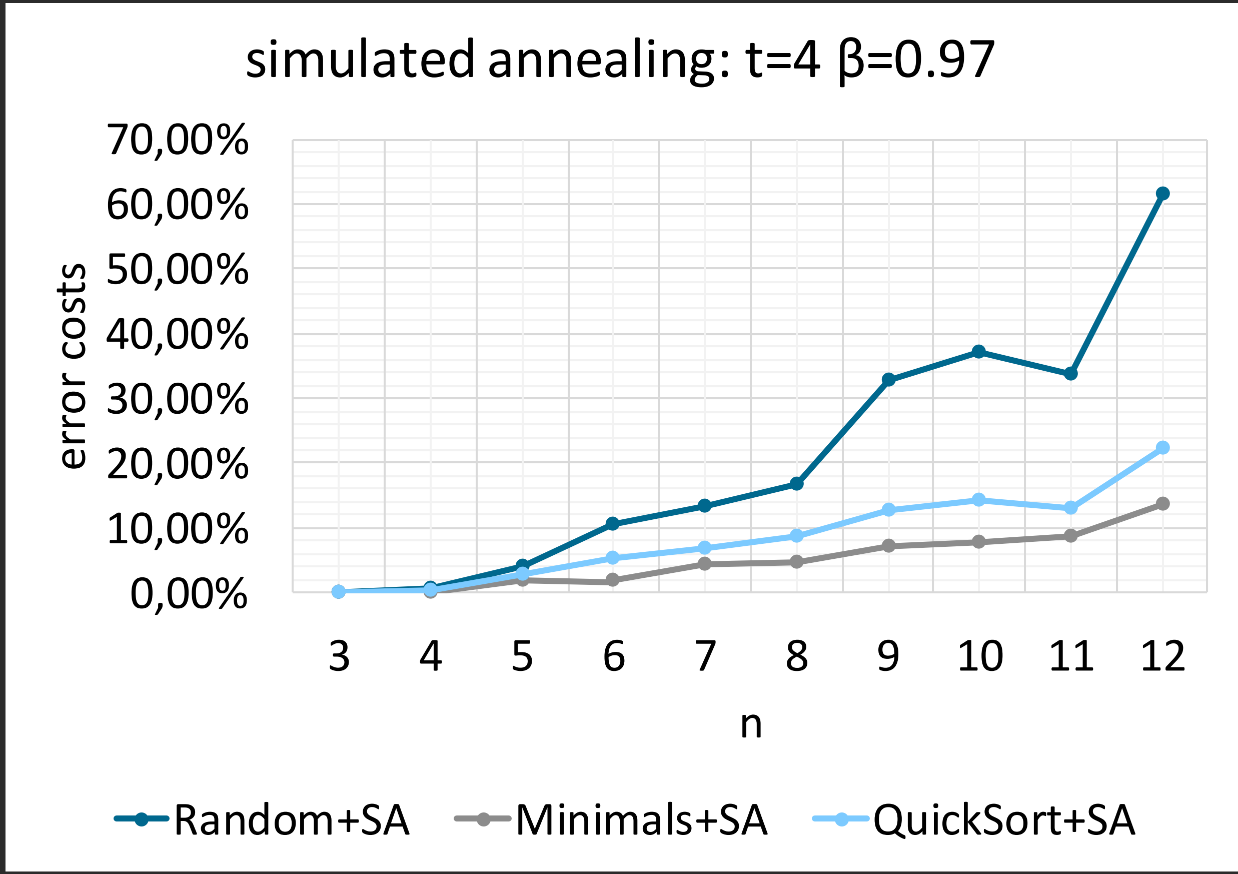
Simulated annealing: initial algorithm

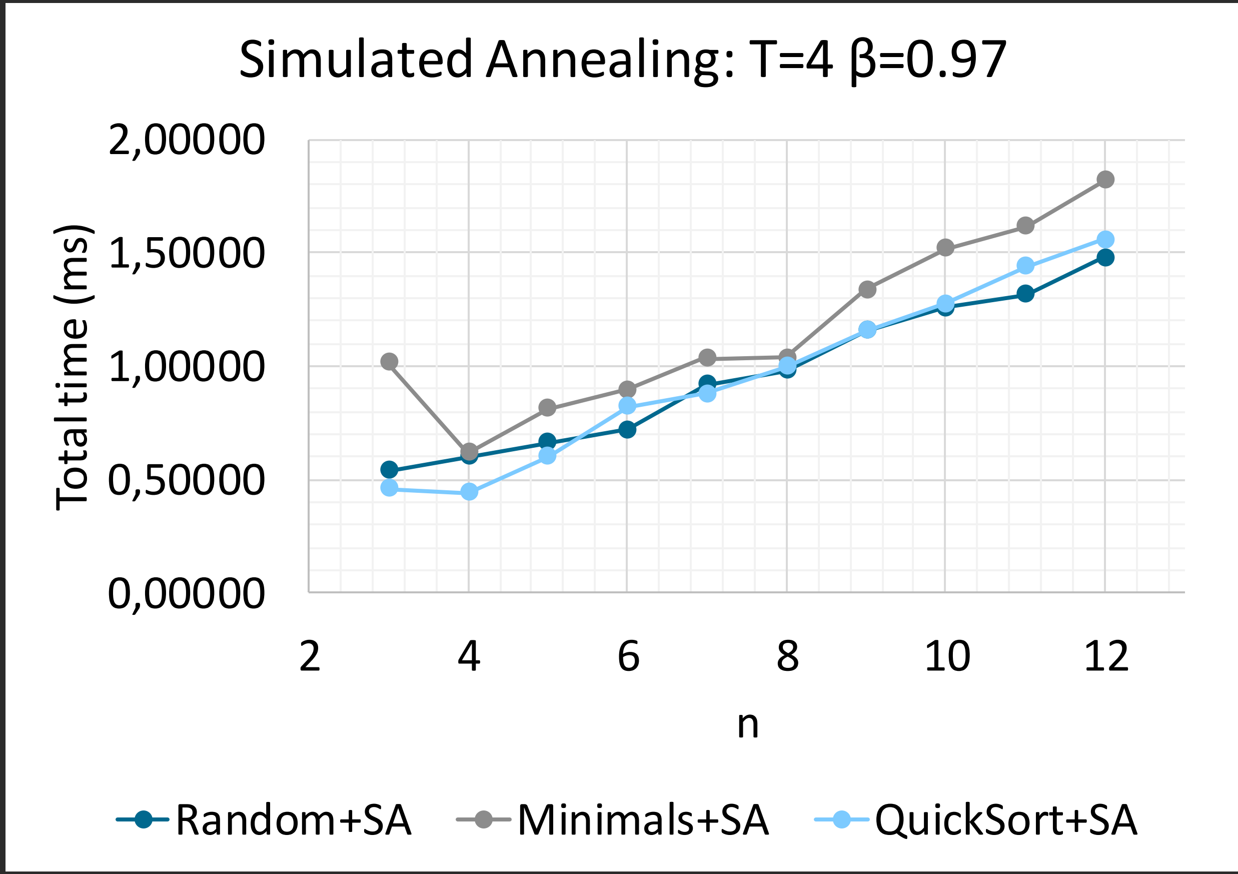




Random → quicksort

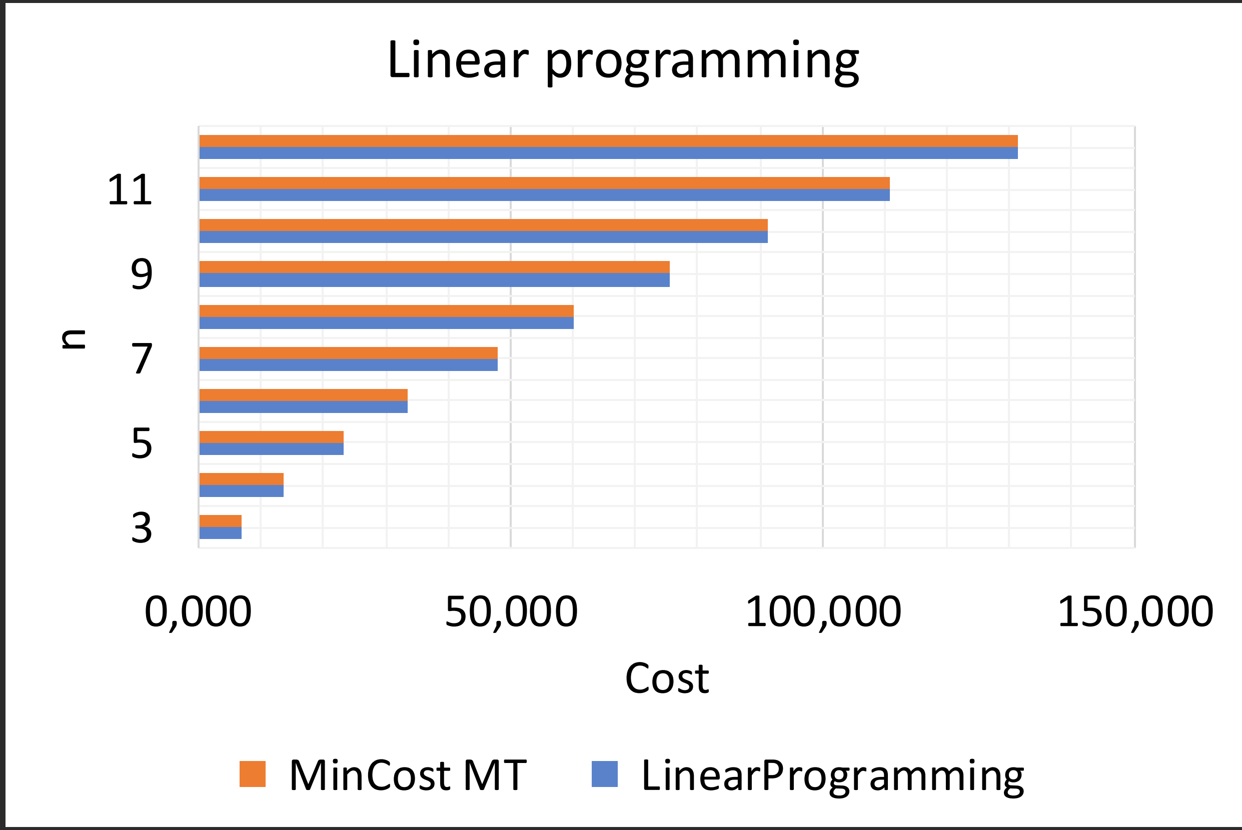
Simulated annealing: initial algorithm

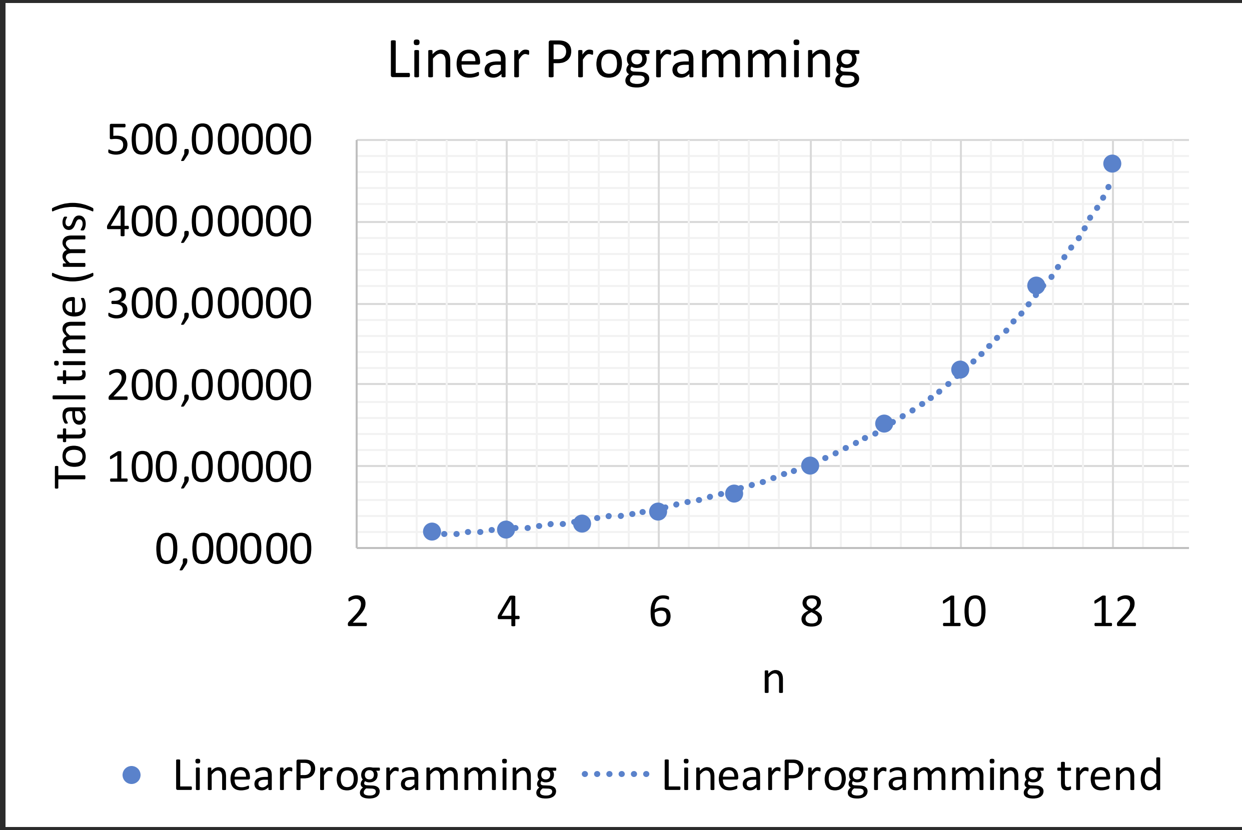




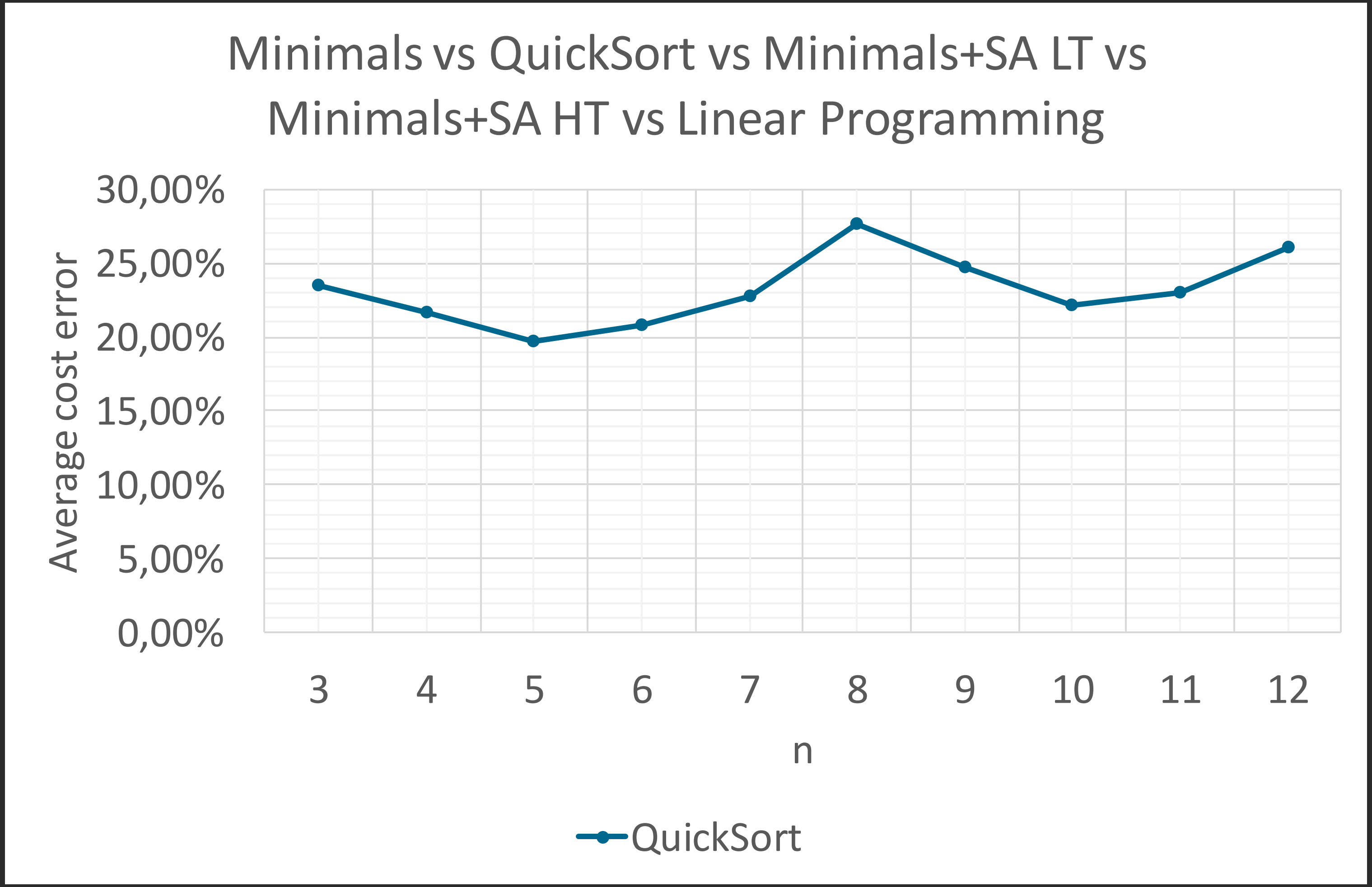
Random → quicksort → minimals

Linear programming



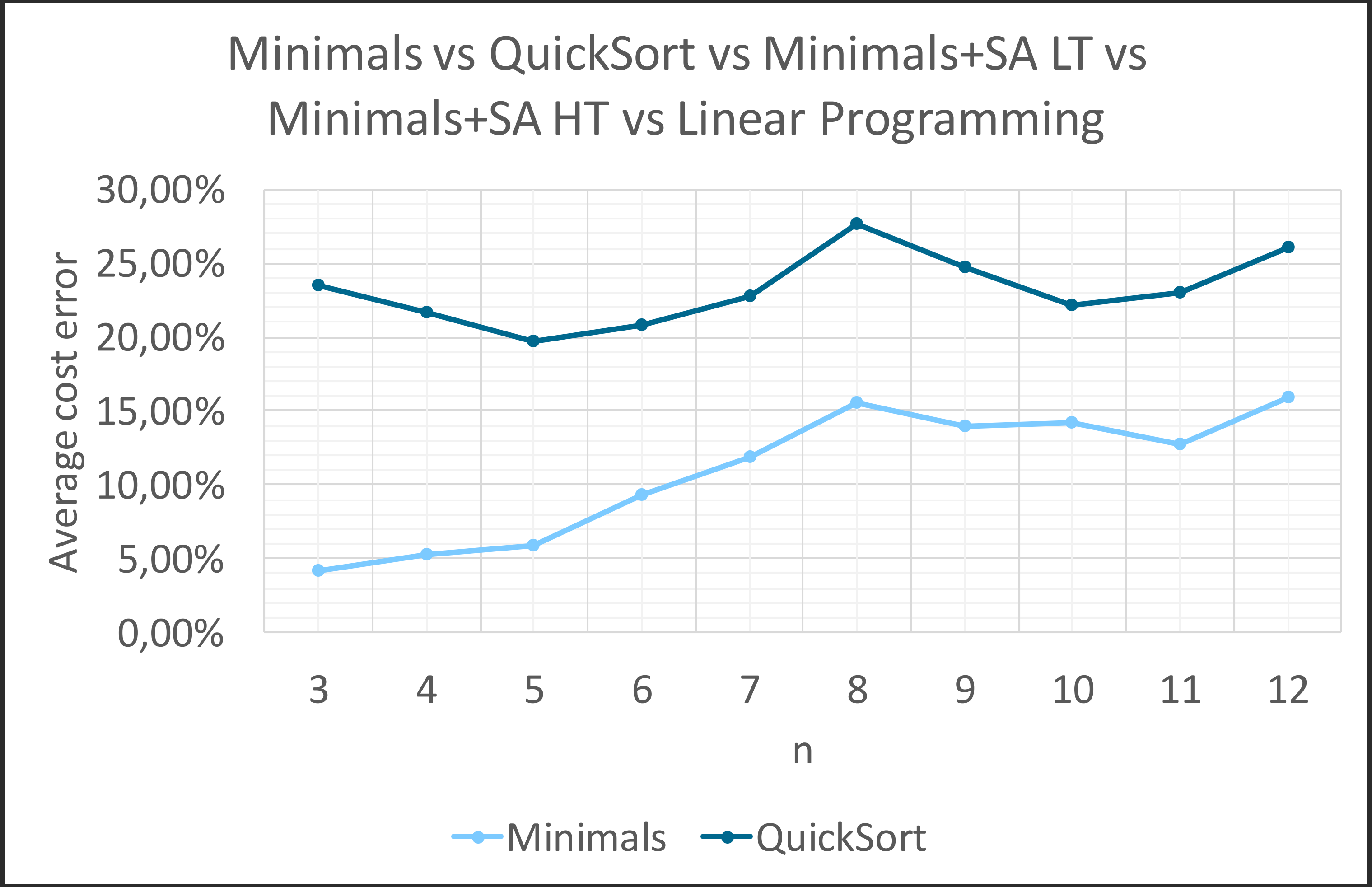


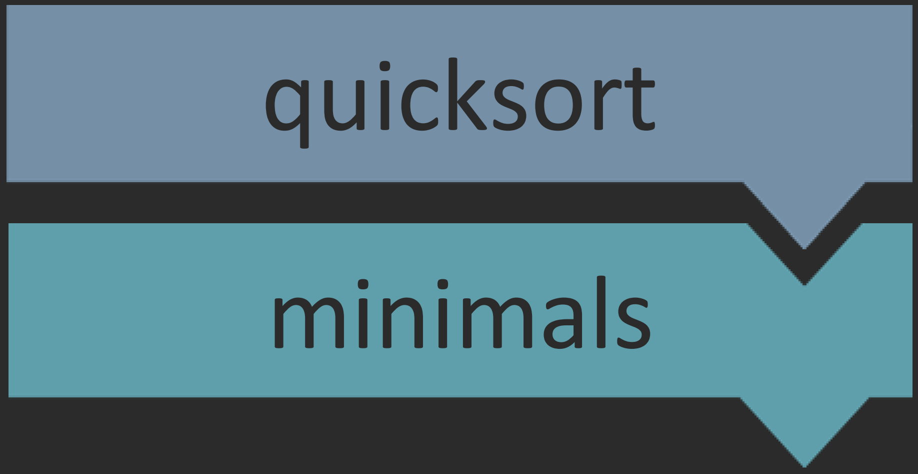
Best aggregation method



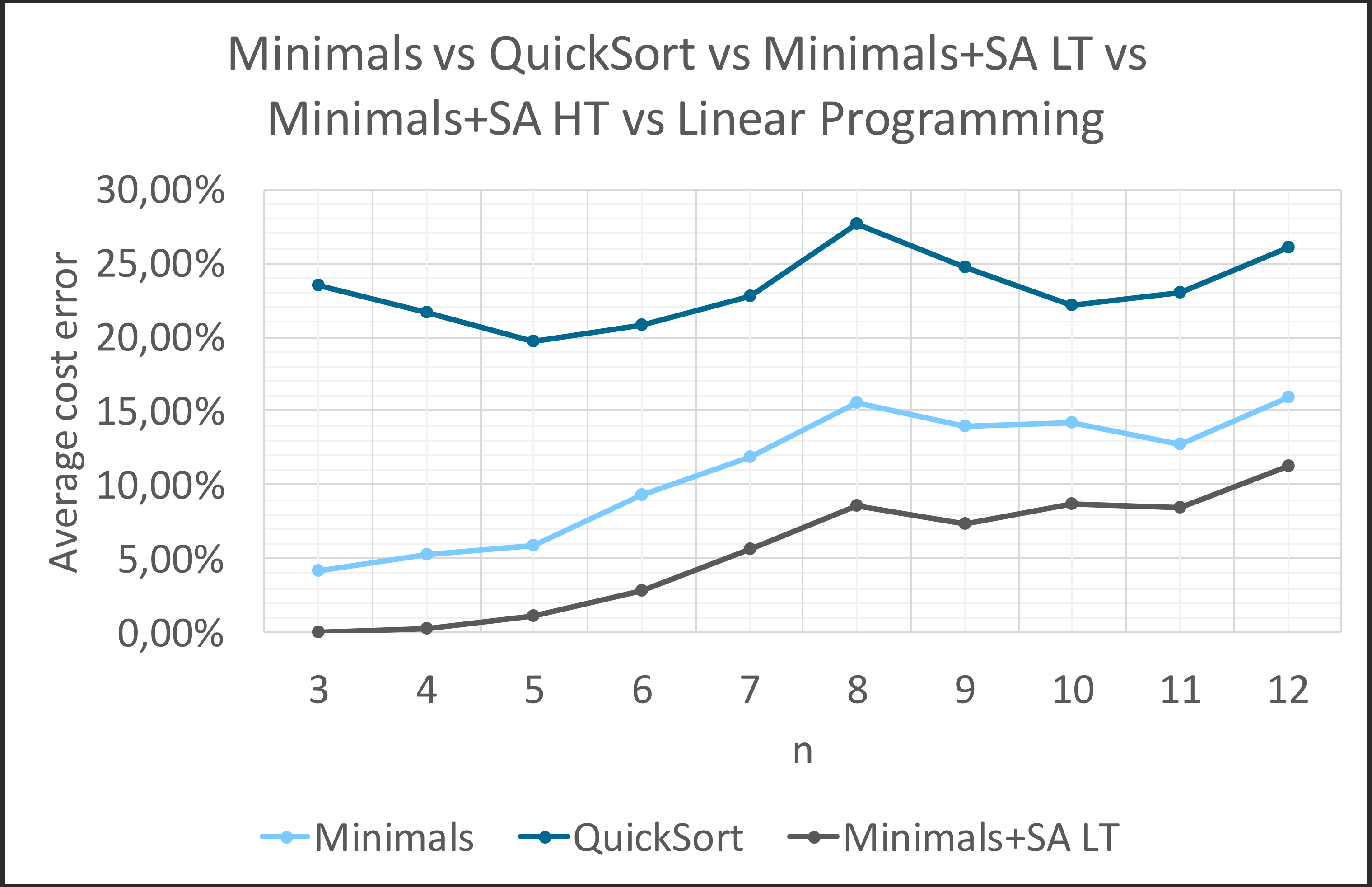
quicksort

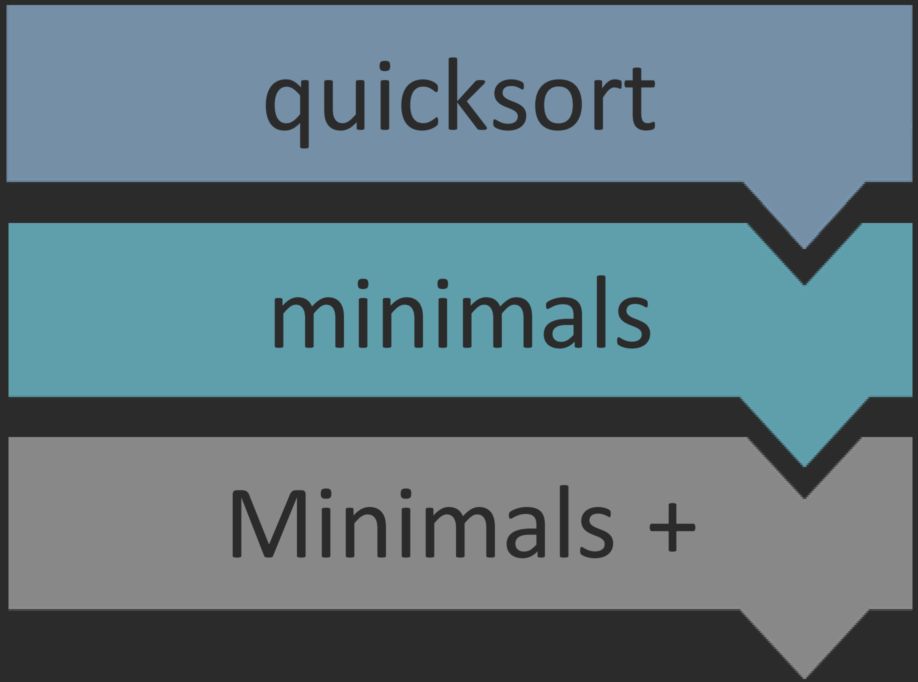
Best aggregation method



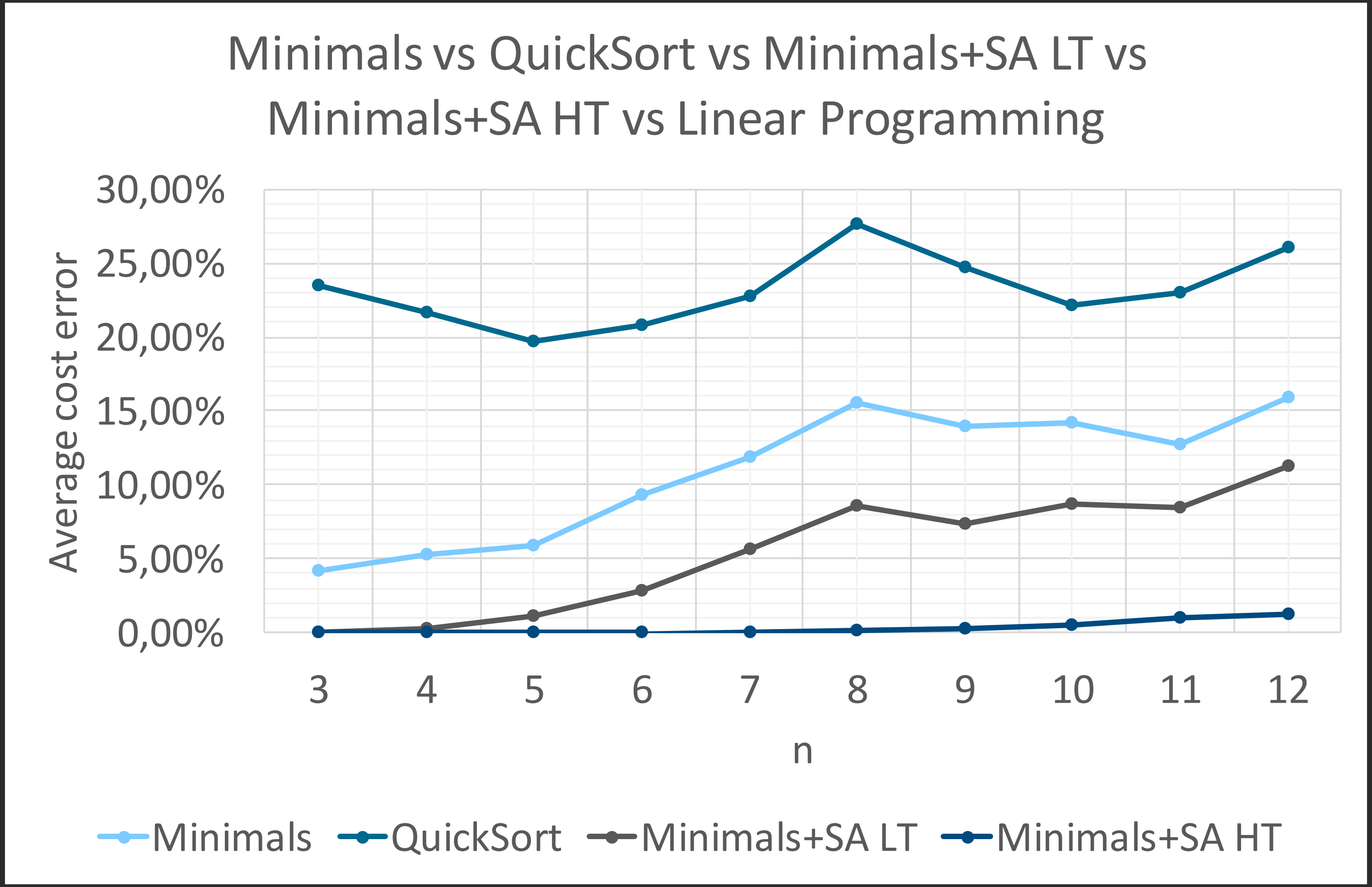


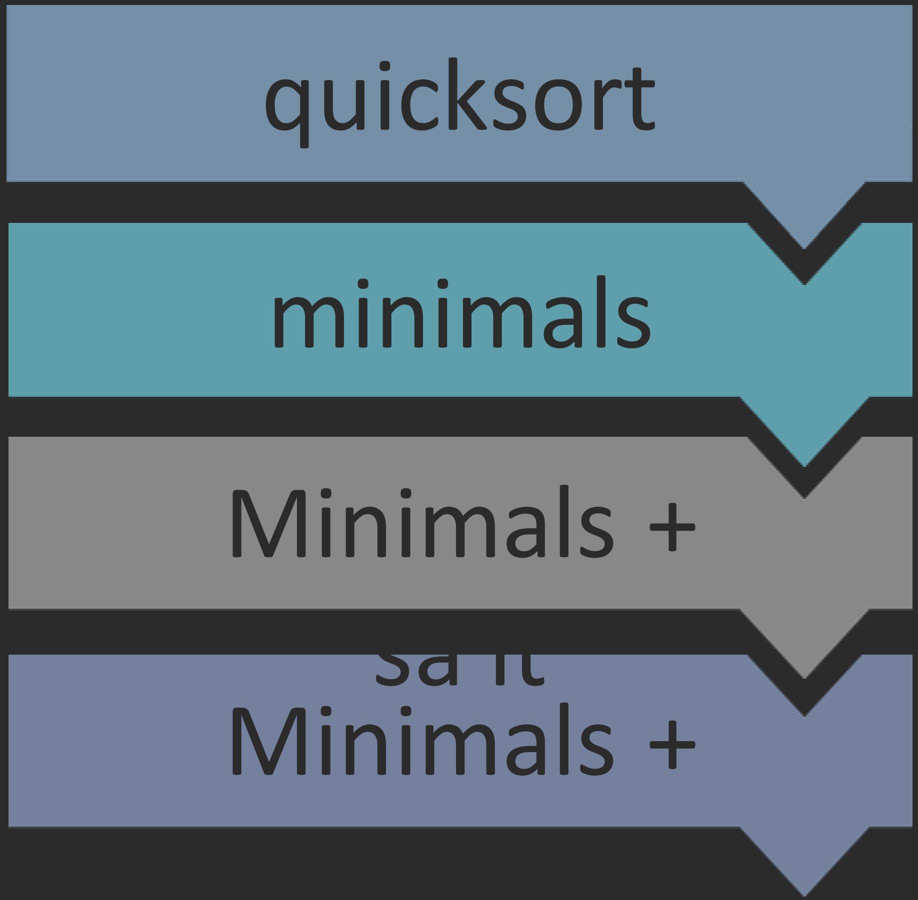
Best aggregation method



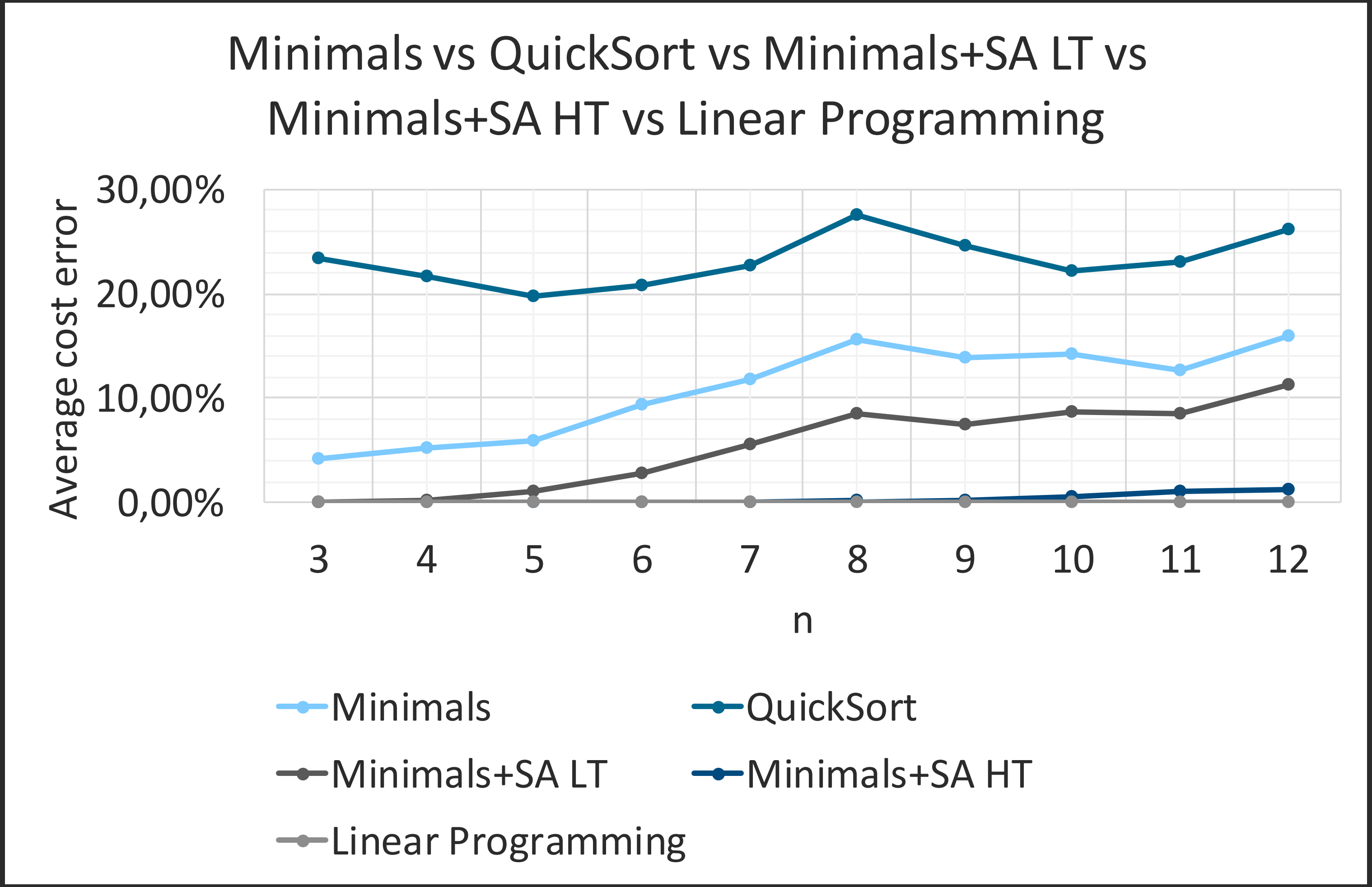


Best aggregation method





Best aggregation method



quicksort

minimals

Minimals + sa lt

Minimals + sa ht

Linear

Best aggregation method

N	Minimals (ms)	QuickSort(ms)	Minimals+SA LT (ms)	Minimals+SA HT (ms)	Linear Programming (ms)	MinCost MT (ms)
3	0,19894	0,03990	0,63184	55,24363	6963,14810	32,82010
4	0,09810	0,01943	0,52893	61,15640	6234,57310	0,97050
5	0,19574	0,01995	0,60648	79,48764	6438,74210	1,99220
6	0,19478	0,05738	0,69495	89,79747	7002,72700	12,14280
7	0,25991	0,04095	0,96700	97,62566	7868,88970	41,58960
8	0,19948	0,05933	1,00327	106,92099	9126,45090	264,87810
9	0,27718	0,05987	1,24245	134,00389	11065,30240	2569,49330
10	0,35919	0,05987	1,39723	148,50157	13921,74060	27600,38580
11	0,35117	0,09916	1,60339	158,93650	18212,21800	275991,06630
12	0,47300	0,06042	1,74745	172,52270	23393,36860	3147048,23830
Total time (ms)	2,60750	0,51625	10,42301	1104,19646	110227,16050	3453563,57700

conclusions

summary

minima
ls

Sorting
algorithm
s

Mincost
st

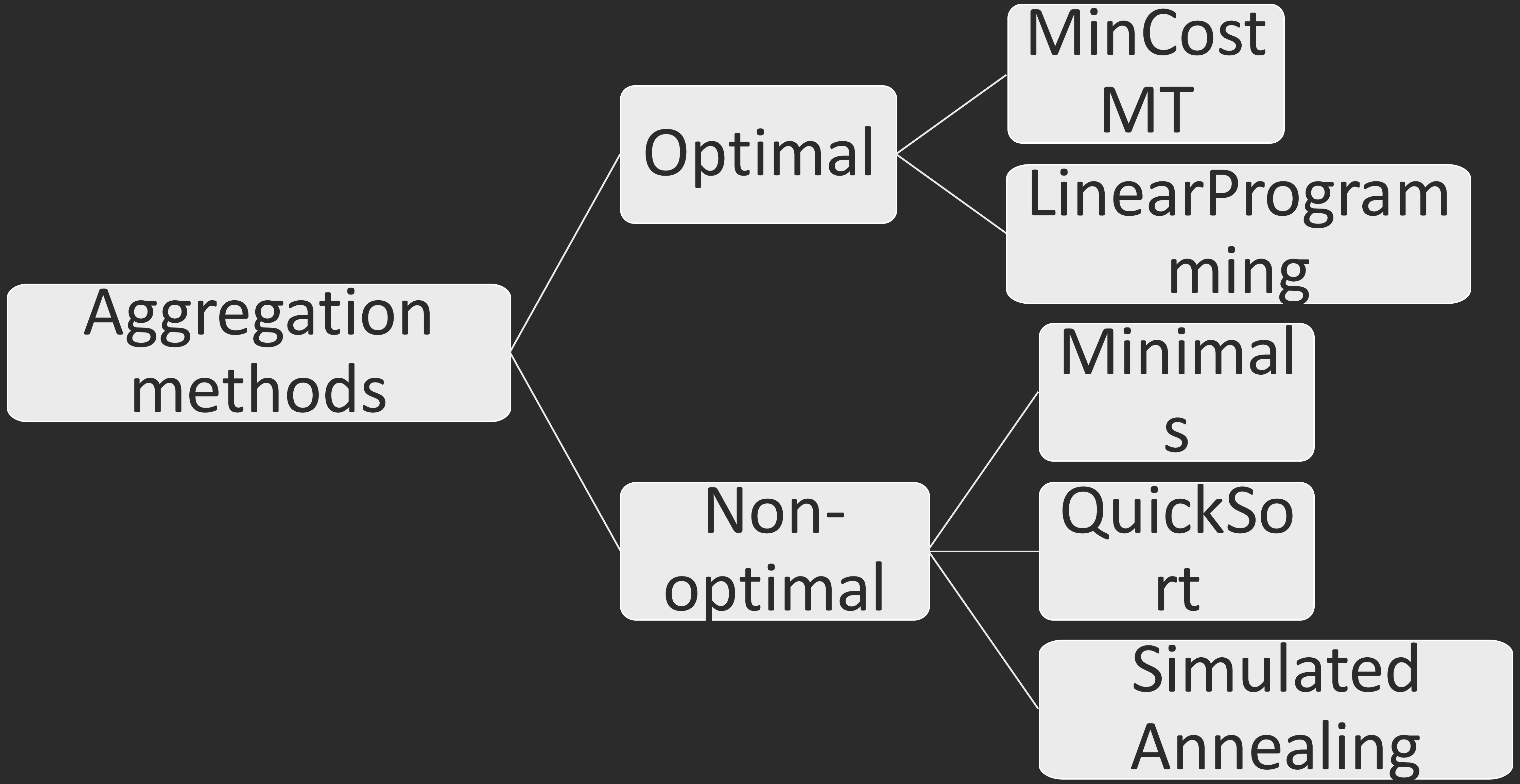
Simulat
ed
anneali
ng

Minima
ls
random

Mincost
mt

Linear
programm
ing

Two categories of algorithms



Simulated annealing

Quality of the initial solu

temperature and cooling co

Optimal algorithms

minCost mT or LinearProgra

Non-Optimal algorithms

Minimals + simulated an

What is the best aggregation method?

$?

What is the best aggregation method?

High temperature

Minimals + simulated annealing

low cooling constant

?