Automated placement of analog ICs using a priority-based constructive heuristic

Mathematical optimisation

Capstone project

Nicolò Ermanno Millo

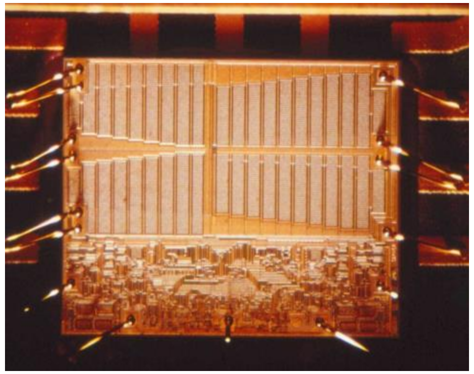
Introduction to the issue

Analog and Mixed-Signal ICs contain devices of **different** size, voltage and freely-chosen position.

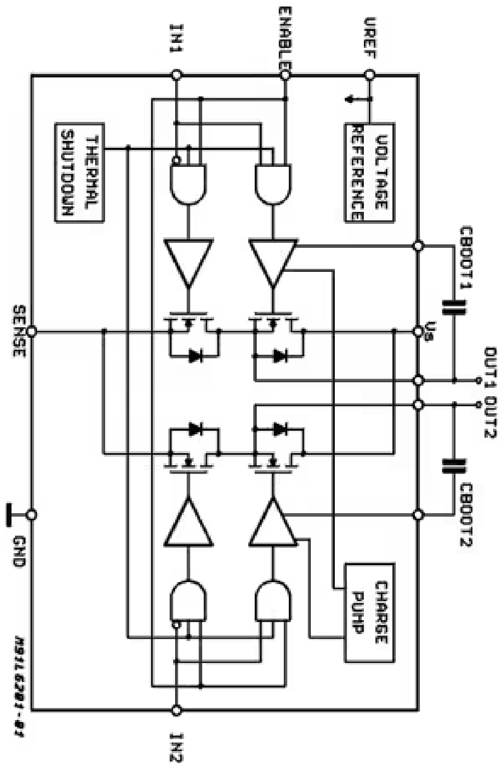
This generates a complex set of constraints for mitigating noise and process variations.

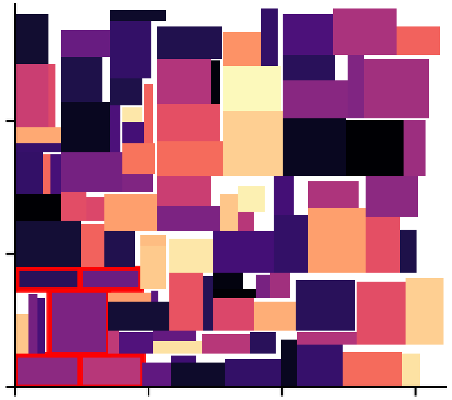
Is it possible automize the placement, reducing also effort and human error?

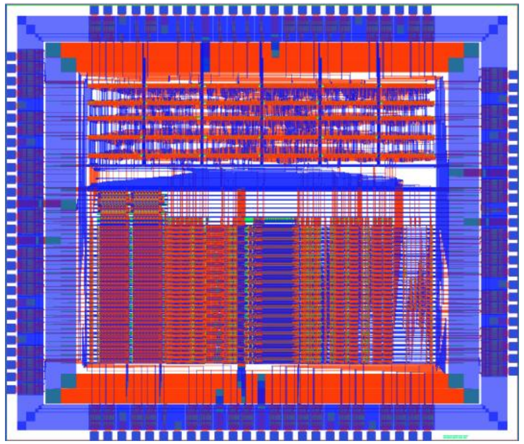
IC physical design



Post-layout
Verification
...







Placement

Routing

BCD placement

The placement phase could be seen as an extension of the rectangle packing problem.

The constraints are imposed by the problem, but also by the chosen manufacturing technology: **BCD** (Bipolar CMOS DMOS)

i.e. symmetric groups, pocket merging, blockage areas, connections, interfaces and proximities

MILP model definition

Instance

Rectangles are single devices or topological structures (i.e. lattice-ladder)

Each can assume several variants (spatial configurations).

Most of them have **connections**

Many are involved into **symmetric groups** or in **proximity bounds** and few are **interfaces**

I rectangles indices

G symmetry group

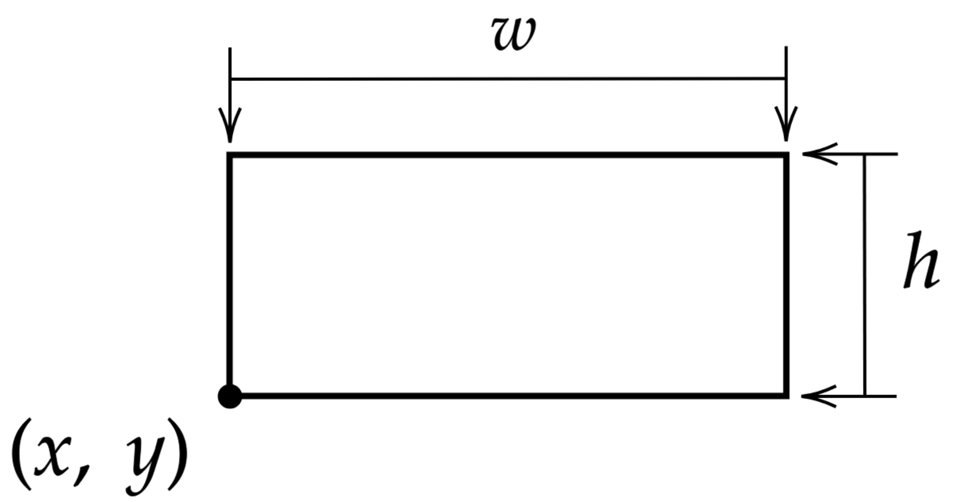
R variants dictionary

E connection dictionary

$a_{i,j}$ distance between rectangles

u_R, l_R min / max ratio bounds

Variables | Rectangles

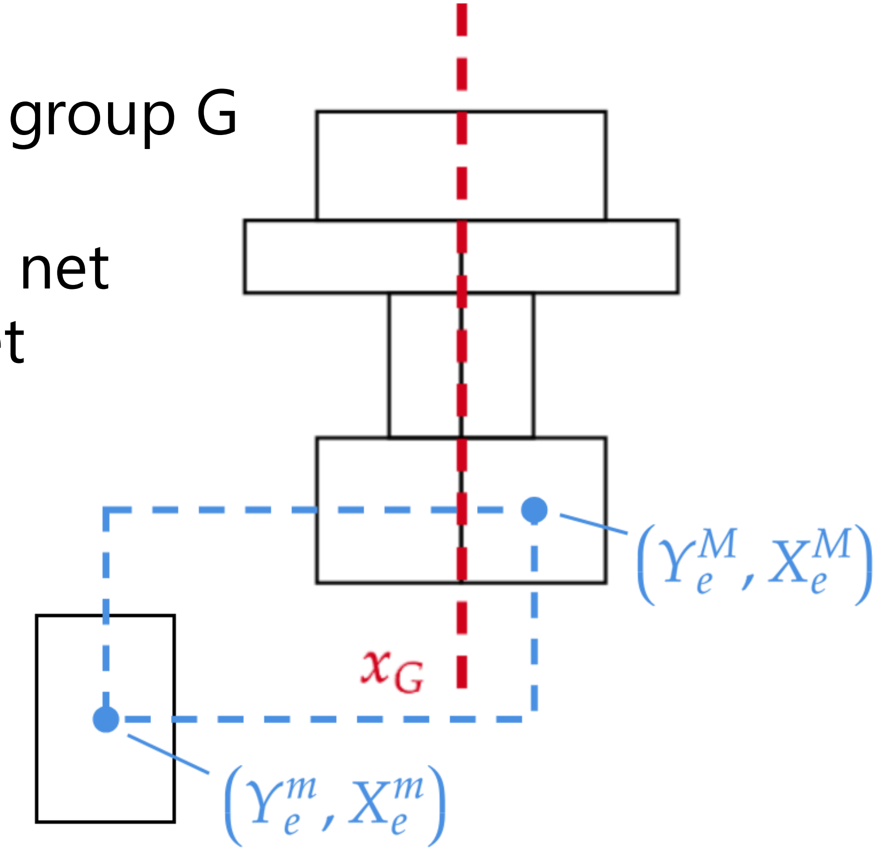


x, y left-bottom corner of rectangle
w, h width / height of rectangle
s^k_i flag of k variant of i rectangle is selected [binary]
$r^k_{i,j}$ spatial relationship between rectangles [binary]

W, H placement max width / height

Variables | Connections & symmetry groups

x_G	vertical coordinate axis of symmetry of group G
r_R	handle non-convex solution space
X_e^M, X_e^m	Centroid horizontal extreme points of e net
Y_e^M, Y_e^m	Centroid vertical extreme points of e net



Constraints

$$x_i + w_i \leq W, \quad y_i + h_i \leq H \qquad \forall i \in \mathcal{I}$$

$$\sum_{k=1}^{m_i} s_i^k = 1 \qquad \forall i \in \mathcal{I}$$

$$w_i = \sum_{k=1}^{m_i} w_i^k \cdot s_i^k \,, \quad h_i = \sum_{k=1}^{m_i} h_i^k \cdot s_i^k \qquad \forall i \in \mathcal{I}$$

$$\sum_{k=1}^{4} r_{i,j}^k \geq 1 \qquad \forall i \forall j \in \mathcal{I} \, : \, i < j$$

$$x_i + w_i + a_{i,j} \leq x_j + M(1 - r_{i,j}^k) \qquad \forall i \forall j \in \mathcal{I} \, : \, i < j \, \forall k \in \{1, 2, 3, 4\}$$

Each rectangle is defined by the dimensions of
one variant and placed within boundaries.

Placement should respect relative positions and minimum distances with other rectangles.

Constraints

$$2 \cdot x_G = x_i + x_j + w_i \qquad \forall (i,j) \in G$$

$$2 \cdot x_G = 2 \cdot x_i + w_i \qquad \forall (i,-) \in G$$

$$w_i = w_j, \quad h_i = h_j, \quad y_i = y_j \quad \forall (i,j) \in G$$

Symmetry groups have a vertical symmetry axis, along which are placed pair or self symmetric rectangles. Pairs should have the same variants.

$$0 \leq l_R \leq R \leq u_R \leq 1$$

$$l_R \cdot W \leq H \leq u_R \cdot W + M \cdot (1 - r_R)$$

$$l_R \cdot H \leq W \leq u_R \cdot H + M \cdot r_R$$

Placement has aspect ratio boundaries.

Constraints

$$X_e^M \geq x_i + w_i/2 \quad \forall i \in L_e \; \forall e \in E$$
$$X_e^m \leq x_i + w_i/2 \quad \forall i \in L_e \; \forall e \in E$$
$$Y_e^m \leq y_i + h_i/2 \quad \forall i \in L_e \; \forall e \in E$$
$$Y_e^M \geq y_i + h_i/2 \quad \forall i \in L_e \; \forall e \in E$$

Set the extreme points of the nets.
They are used in objective function.

Objective function

Minimize weighted normalized multi-criteria objective function

$$\mathcal{L} = c_{area} \cdot \mathcal{L}_{area} + \frac{c_{conn}}{S_{conn}} \cdot \mathcal{L}_{conn} + \frac{c_{face}}{S_{face}} \cdot \mathcal{L}_{face} + \frac{c_{prox}}{S_{prox}} \cdot \mathcal{L}_{prox}$$

Connectivity criterion:

$$\mathcal{L}_{conn} = \sum_{\forall e \in E} c_e \cdot (X_e^M - X_e^m + Y_e^M - Y_e^m) \qquad S_{conn} = \sum_{\forall e \in E} c_e$$

Area criterion:

$$\mathcal{L}_{area} = W + H$$

Objective function

Minimize weighted normalized multi-criteria objective function

$$\mathcal{L} = c_{area} \cdot \mathcal{L}_{area} + \frac{c_{conn}}{S_{conn}} \cdot \mathcal{L}_{conn} + \frac{c_{face}}{S_{face}} \cdot \mathcal{L}_{face} + \frac{c_{prox}}{S_{prox}} \cdot \mathcal{L}_{prox}$$

Proximity criterion:

$$\mathcal{L}_{prox} = \sum_{\forall (i,j): c_{prox}^{i,j} \neq 0} c_{prox}^{i,j} \cdot d_{i,j}$$

$$\mathcal{S}_{prox} = \sum_{\forall (i,j)} \left| c_{prox}^{i,j} \right|$$

Interface criterion:

$$\mathcal{L}_{face} = \sum_{\forall i \in F: c_{face}^{i} \neq 0} c_{face}^{i} \sum_{\forall f} d_{i,f} \cdot \chi(f)$$

$$S_{face} = \sum_{\forall i \in F} c_{face}^{i}$$

Proposed solution

Constructive heuristic

Given a set of rectangles, it adopts an **integer**, **greedy** and **priority-modulated** approach to create a feasible solution

Integer: each rectangle is placed on the corners or projection points of other rectangles

Greedy: heuristic reduces for each iteration the current multi-criteria

Priority-modulated: change order to place considering cost of connection

Pipeline

Genetic Algorithm → CMA-ES → Local Search Sequence → Local Search Layout → Gurobi LP

Genetic Algorithm	CMA-ES	Local Search Sequence	Local Search Layout	Gurobi LP
Random initiated Constructive Multithreading DQN enhanced	GA best initiated Constructive Multithreading	Deterministic Constructive	Deterministic Constructive	Real fast from constructed solution

Evolutionary metaheuristics

EAs explore encoded solution space (chromosome) trying to improve their fitness.

- Chromosome: $N \times$ (position, variant, direction) + priority module
  All of them are coded in [0,1]
- x, y position is decoded via **limited Cantor pairing function**
- Fitness: minimize multi-criteria

Genetic algorithm

Key features:
- Highly Customizable (quality and time effort)
- Robust to Local Optima

Hyperparameters:
- Childs
- Crossover probability
- Mutation probability
- Population size
- Tournament size

GA | Deep Q Learning

Q-learning where action-value function is NN-approximated.

Applied to GA, it is hugely expensive with no return of improvement **assured** [2].

But sometimes could be rewarding.

X = "population" $\Rightarrow$ $\boldsymbol{Q} \in \mathbb{R}^{\text{pop\_size}\times 3\ +1}$

$U_{\text{tour\_ratio}} = \{ -0.02, -0.05, -0.1, -0.2 \}$

$U_{\text{cross\_prob}} = \{ -0.3, -0.8 \}$

$$r_t = \frac{\text{best}_{t-1} - \text{best}_t}{\text{best}_1}$$

CMA-ES

Key Features:

- Self-adaptive and nearly hyperparameter-free
- Capture variable dependencies

Hyperparameters:

- Sigma
- Population size

Powered by the **cma** Python module

Local search over sequences

From metaheuristics best solution, we try to change all other variants for each rectangle (**neighbourhood**).

With constructive heuristics we evaluate new solutions.

If the **local optima** is better, it will be passed on to the next LS

Local search over layout

From sequence LS solution, we try to re-place each rectangle (changing its variants), keeping all others fixed.

Even though, both LS do not improve solution at any time, they are effective when metaheuristics were not.

Gurobi

It solves the formerly defined MILP, but

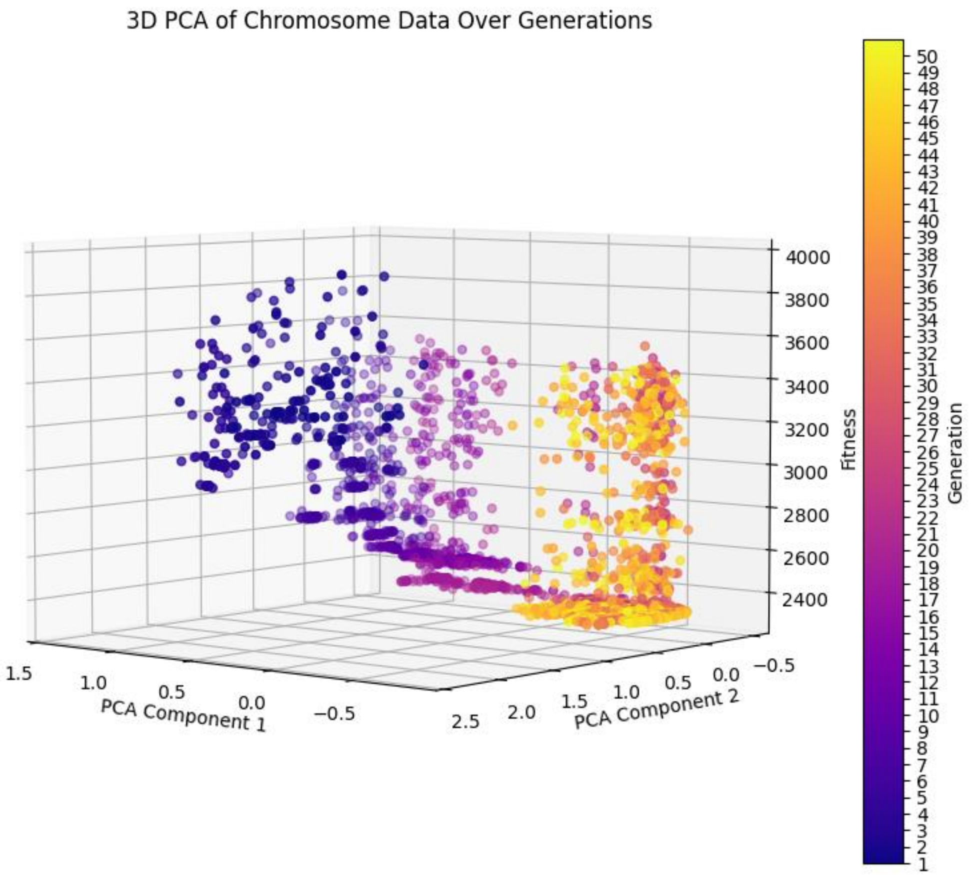
1. It starts from best solution of layout LS
2. Relative position of rectangles $r^k_{i,j}$ are **fixed**
3. Variants s^k_i are **fixed**

So, it is a LP

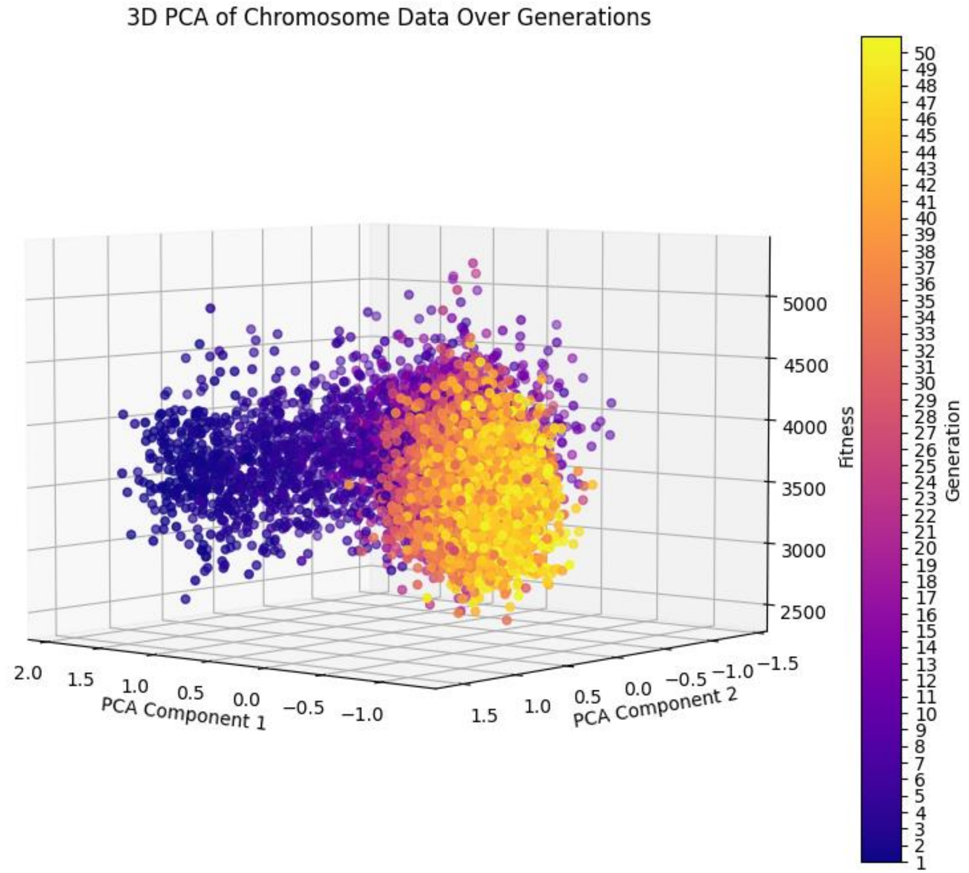
This guarantees that the optimal solution is reached in no time, even though it is **suboptimal** problem-wise.

Results

Results | Metaheuristics convergence

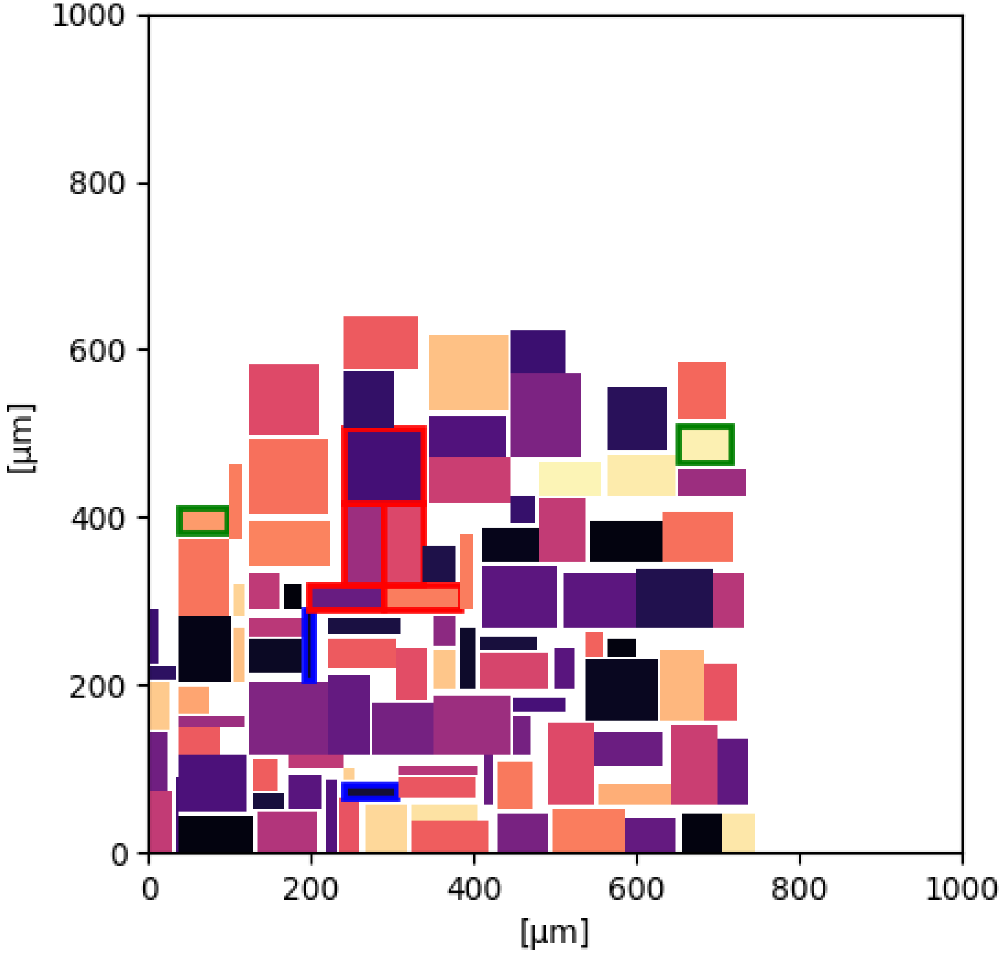


GA

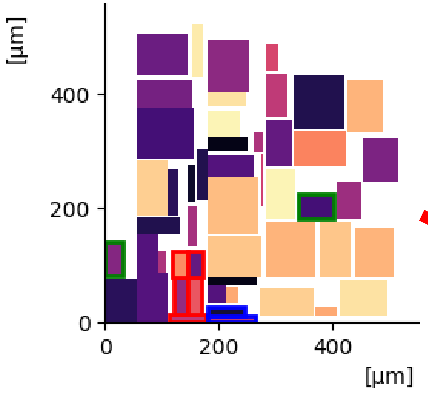


CMA-ES

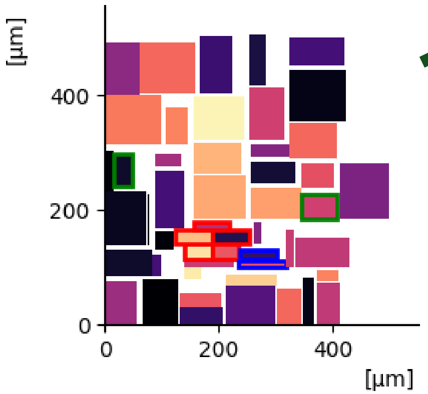
Results | Placement



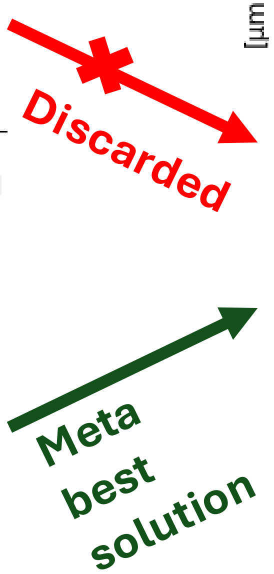
Results | Placement optimization

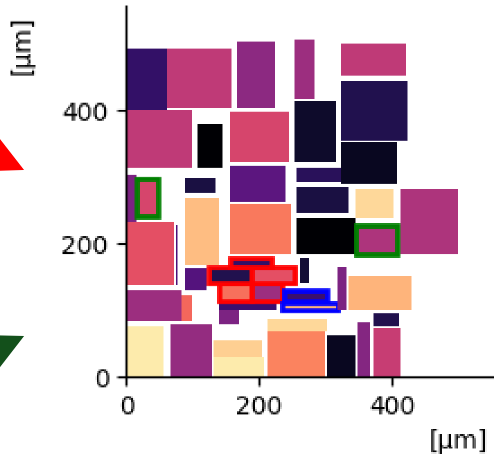


GA

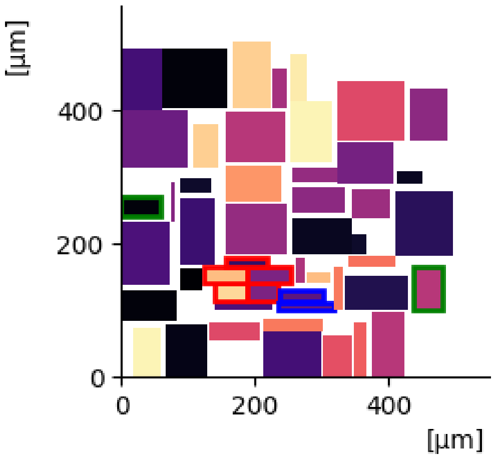


CMA-ES

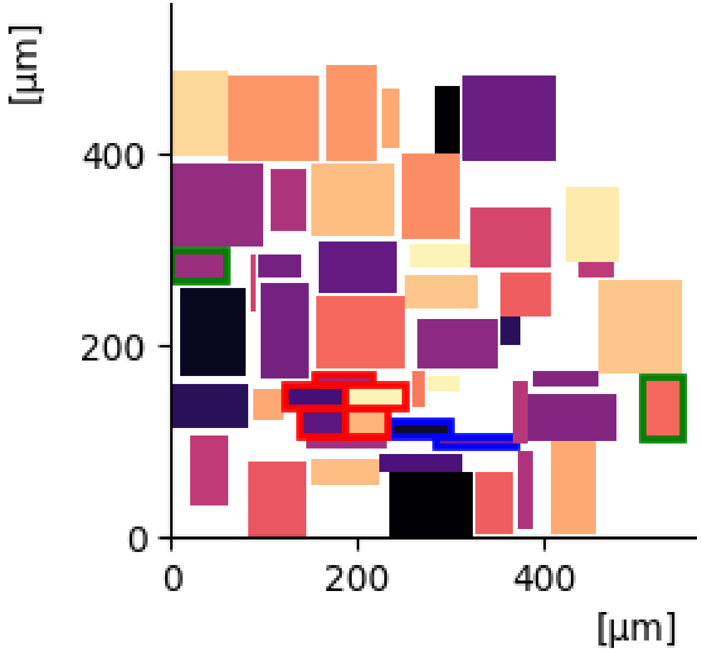




Local search
over sequences



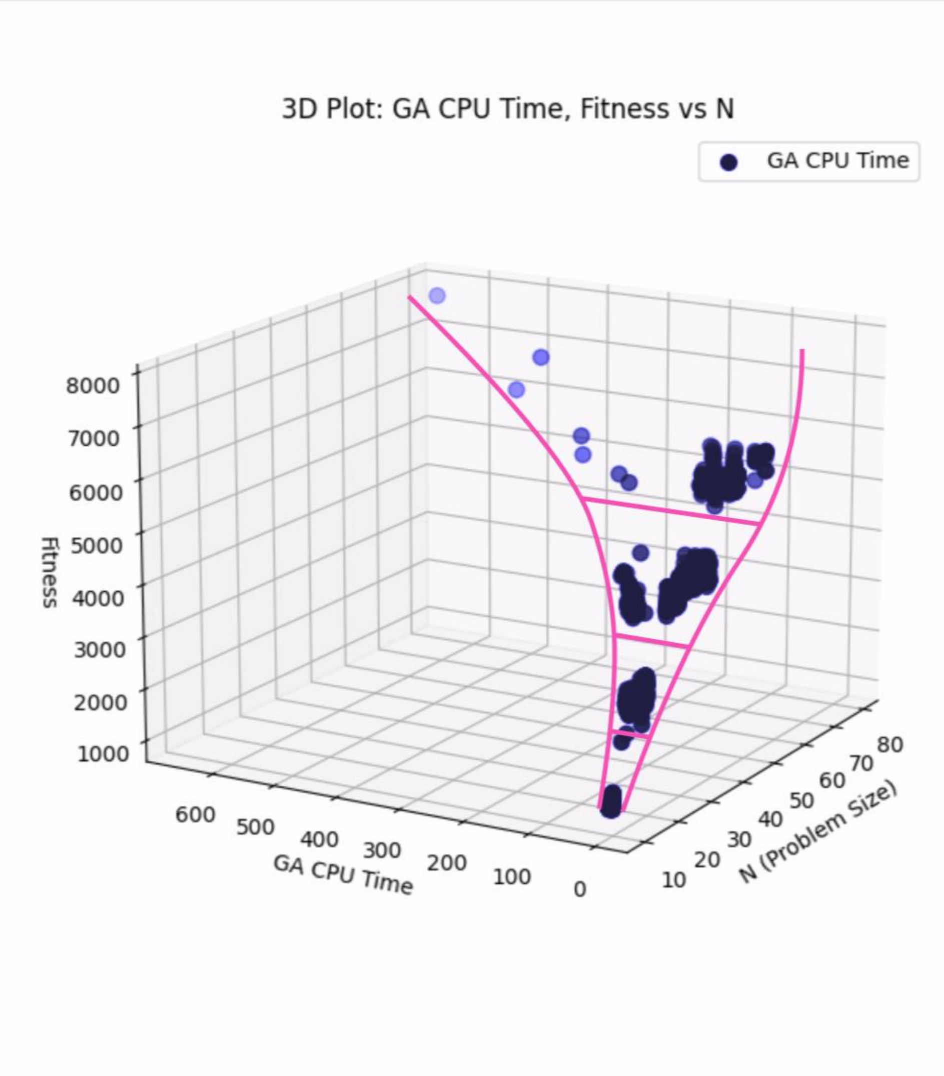
Local search
over layout

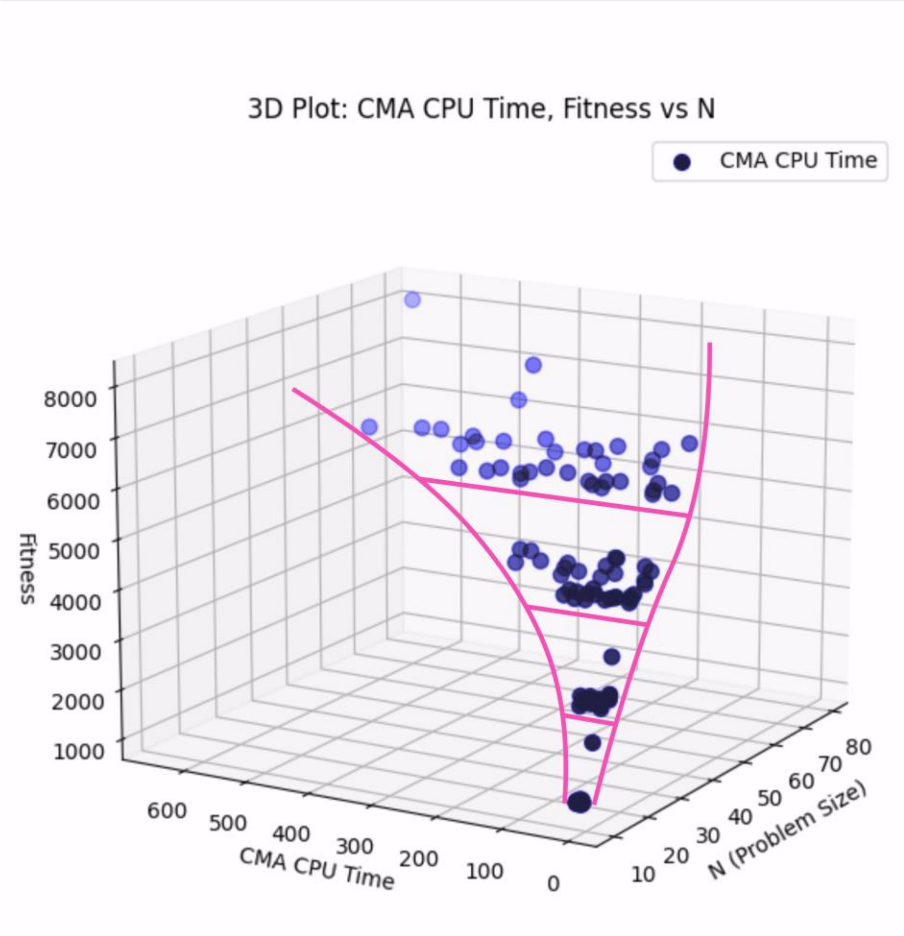


Gurobi LP

Scalability Analysis

Empirical analysis on benchmarks





Empirical analysis on benchmarks

Instance size N influences **CPU time**,

but metaheuristics rely also on hyperparameters.

Whereas it is **not** so for LSs and Gurobi.

Fitness and **CPU time analysis** could lead the choice of hyperparameters based on their dependencies.

GA scalability analysis | CPU time

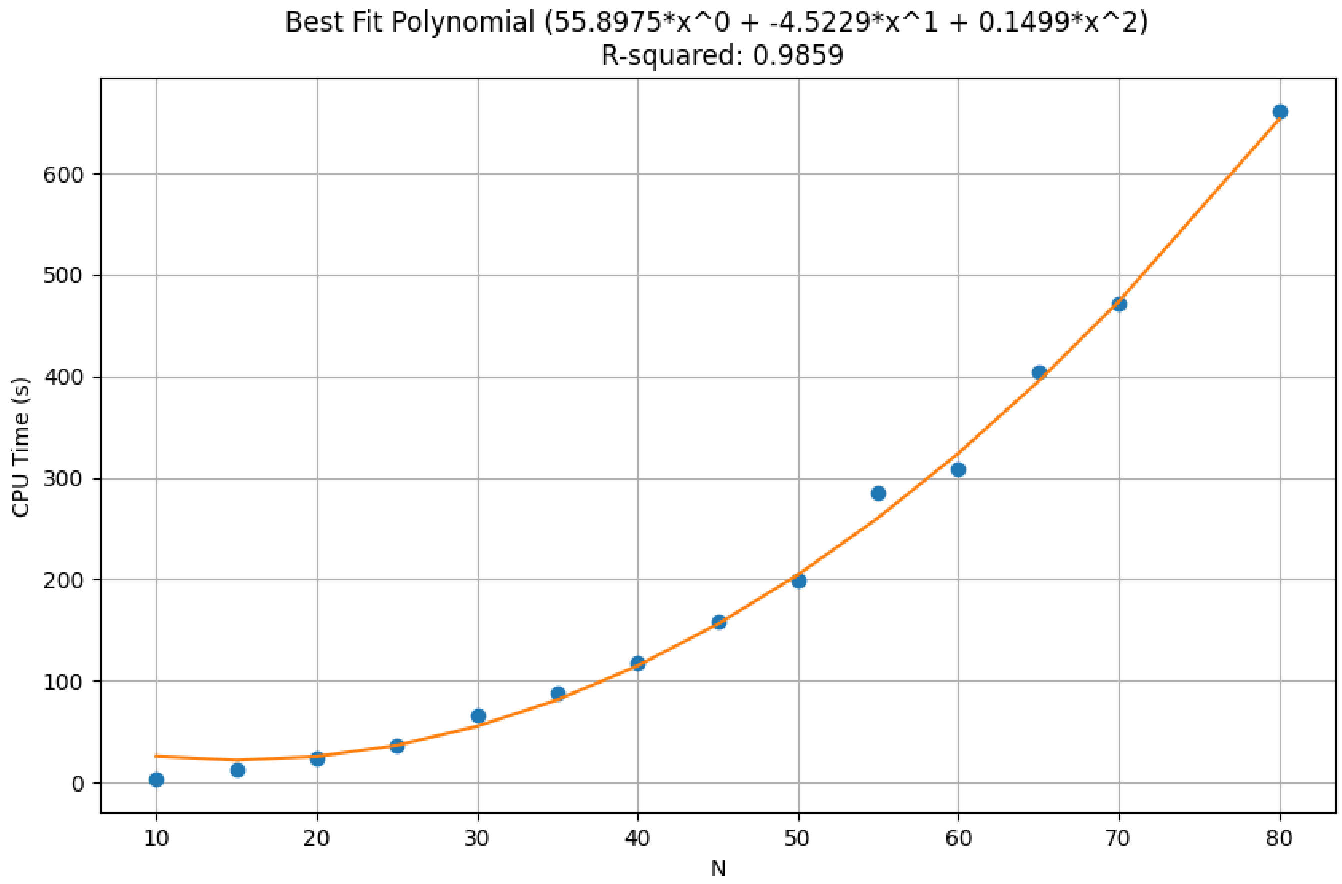
To grasp dependance on CPU time, it was performed a linear GLS regression respect to CPU time.

CPU time is affected the most by:

1. **N:** 1.3600 (p=0.000)
2. **Childs:** 0.8910 (p=0.000)
3. **Population size:** 0.1015 (p=0.000)

Model shows a strong fit (**R-squared:** 0.922)

GA scalability analysis | CPU time



Polynomial regression over cross validation

Fixing hyperparameters, N increases following a **degree 2** polynomial law: **O**(N^2)

So, for each fixed hyperparams configuration

GA scalability analysis | Fitness

Fitness did not show clear relationships even on higher poly degree, outside of N.

It does not fit well (R-squared: 0.532)

Maybe the high unpredictability of fitness is the reason for lower effectiveness of DQN.

CMA-ES scalability analysis | CPU time

Similarly as GA, GLS regression on CMA CPU time showed what is expected.

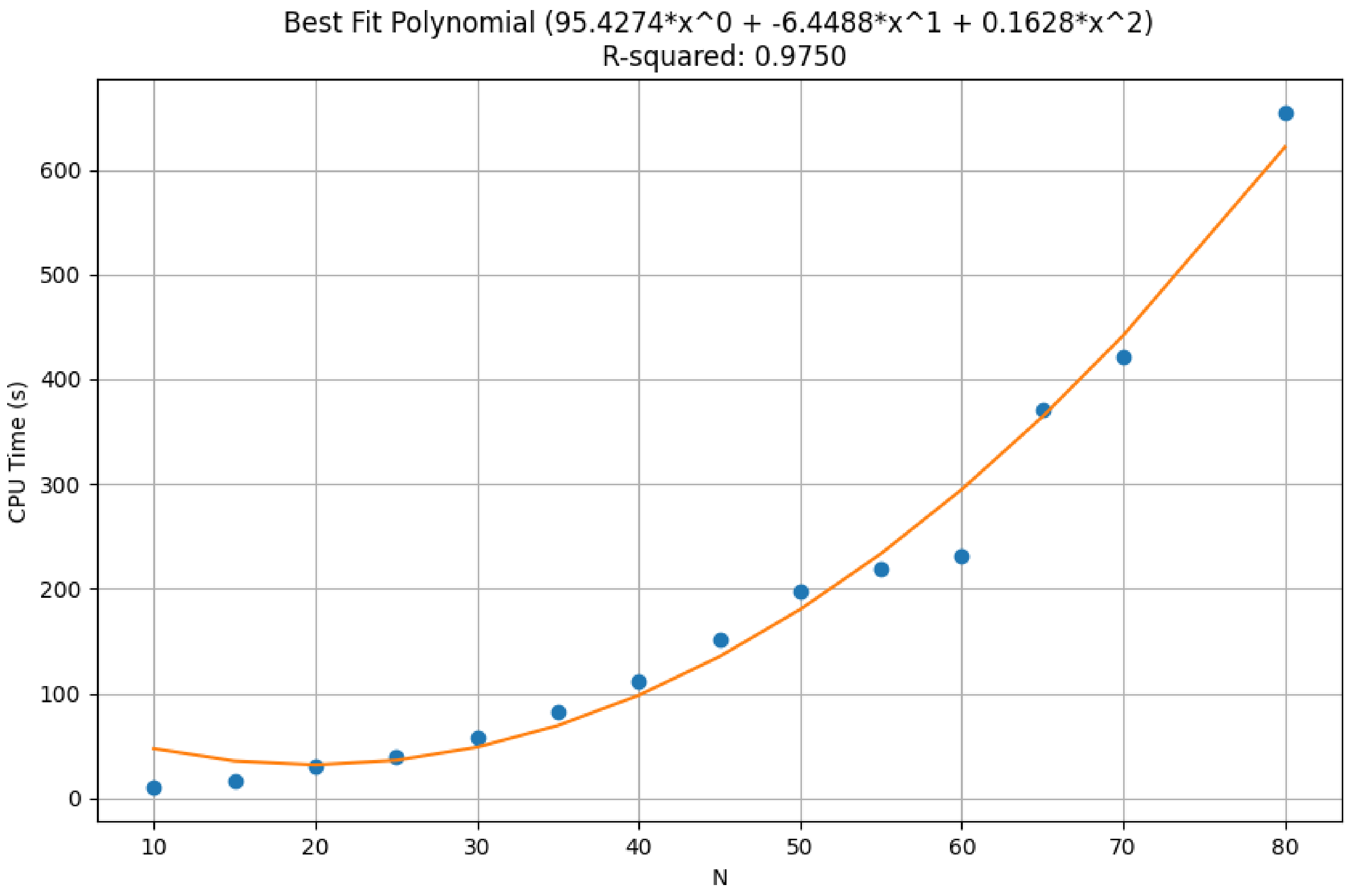
Time is affected the most by:

1. **N:** 0.8372 (p=0.000)
2. **Population size:** 0.4029 (p=0.000)
3. **Sigma:** -0.0625 (p=0.000)

Model shows a strong fit (**R-squared:** 0.889)

CMA-ES scalability analysis | CPU time

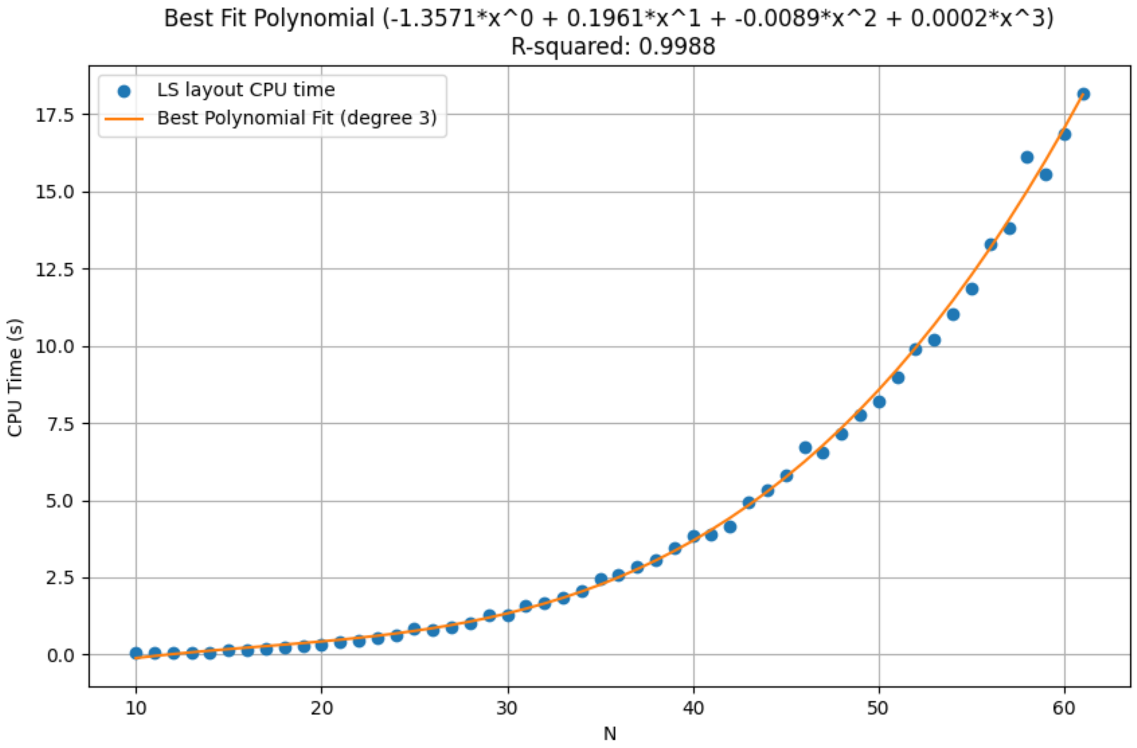
$O(N^2)$

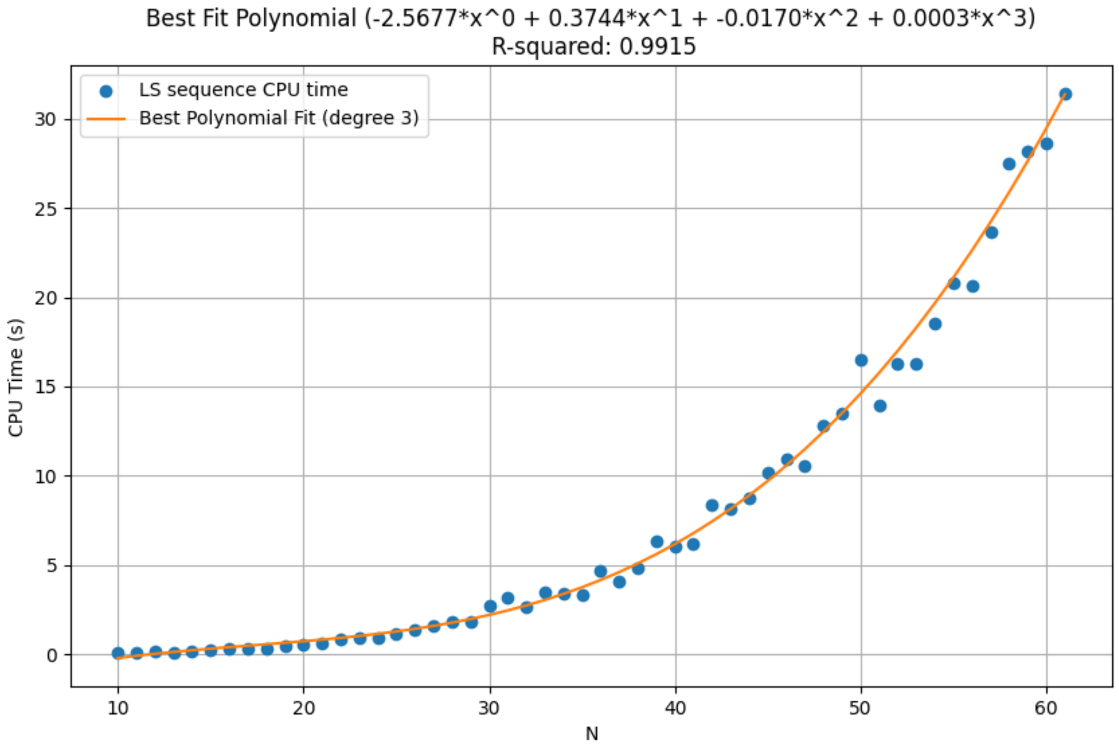


CMA-ES scalability analysis | Fitness

As GA Fitness model, CMA-ES one does not fit well and does not show clear relationships outside N

Local searches | CPU time



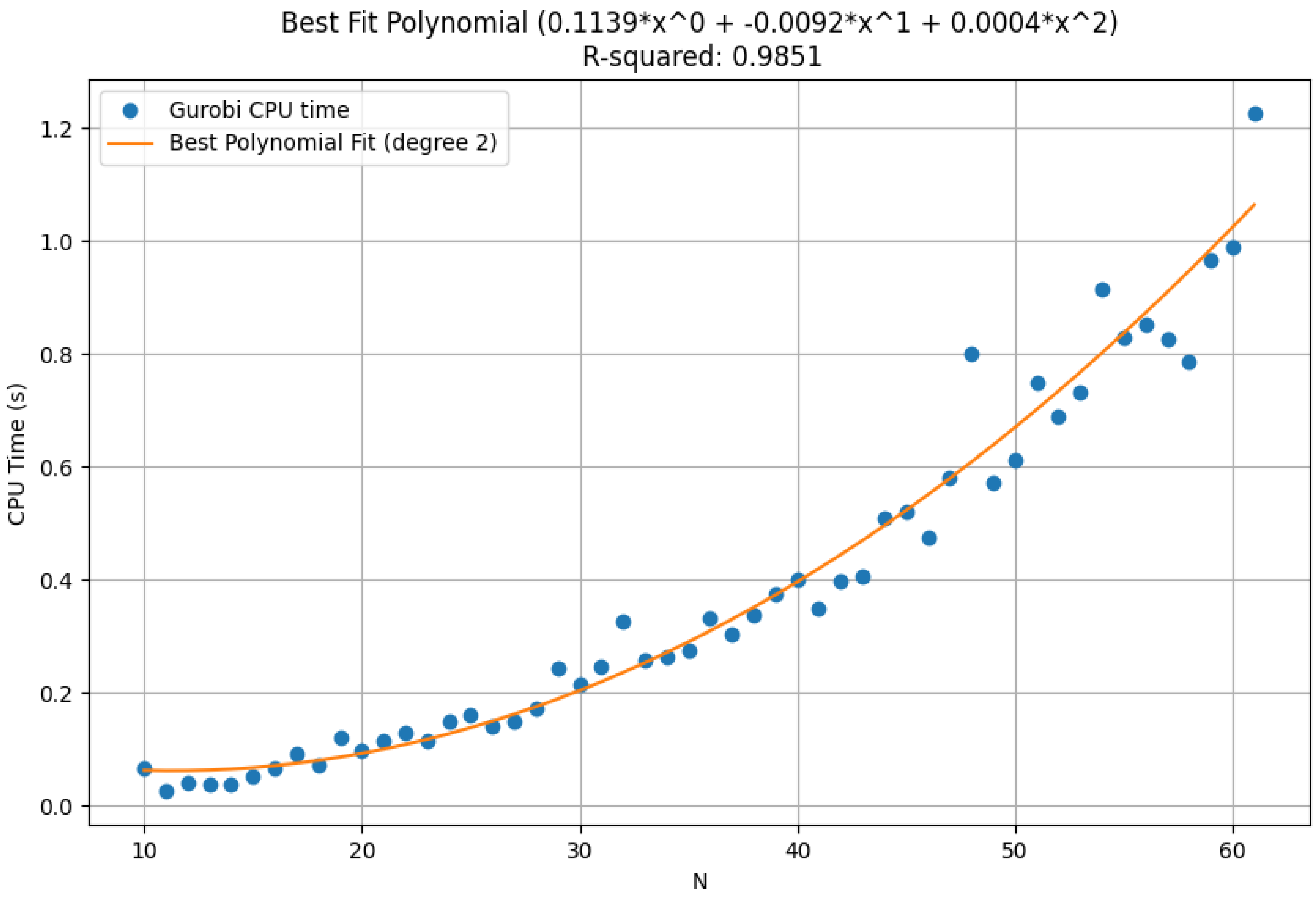


Local search over layout

$O(N^3)$

Local search over sequences

Gurobi | CPU time



Additional improvements

Profiling metaheuristics

This probe aims to catch the **cumulative time** heaviest function of the constructive heuristic.

For this reason, exploiting low level optimization is pivotal:

1. rewriting bottlenecks of constructive heuristic in **C** (**`cffi`**) reduce CPU time at least of 55%
2. EA fitness computation supports multithreading, but there are still plenty of room to further parallelization

Both are hardware dependant.

Multiobjective hyperparams tuning

What combination of **criteria costs** benefits overall the most ?

NSGA-II helps a little to produce meaningful solutions, ***but***

- No clear optima reaching
- Hugely expensive
- It stressed again fitness unpredictability.

Conclusions

Is it possible automize the placement, reducing also effort and human error?

Yes, the solutions computed by the pipeline seems almost always solid.

But fitness improving hyperparameters tuning is not cheap and sometimes not effective at all.

Bibliography

[1] Grus, J., Hanzálek, Z., Barri, D., Vacula, P., *Automatic placer for analog circuits using integer linear programming warm started by graph drawing*. *Proceedings of the 12th ICORES*, SciTePress, 2023

[2] Grus, J., Hanzálek, Z., *Automated placement of analog integrated circuits using priority-based constructive heuristic. Computers & Operations Research*, 2024

Thank you

Further details:
https://github.com/ermannomillo/AMSIC_placement_solver