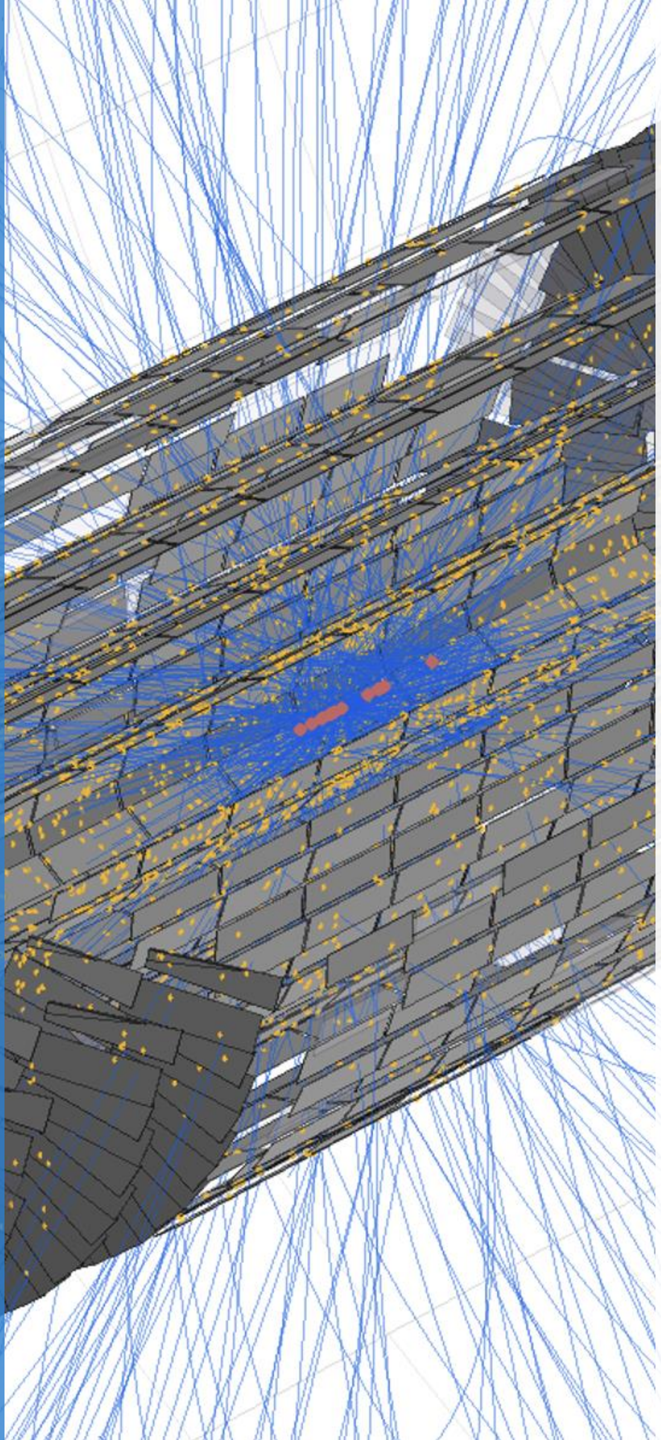
A Multi-Objective Optimization Tool for Track Reconstruction in CMS

Simone Rossi Tisbeni (CERN-UniBO-INFN)
Aya Arabiat (UTD), Adriano Di Florio (CC-IN2P3), Chris Hoang (UCSD) , Enrico Lusiani (INFN), Felice Pantaleo (CERN)

On behalf of the CMS Collaboration

CHEP 2024
27th International Conference on Computing in High Energy Physics
24 Oct 2024 - Krakow(PL)



Tracking at HLT

- **CMS trigger**: two levels Level 1 (hardware) and High Level Trigger (HLT) (software)
  Input rate: ~100kHz, **Output rate:** few kHz
- Track reconstruction is one of the first (and most important) steps of HLT
- **Full tracks** (pixel+strips) **seeded by pixel-only tracks**
- Pixel track reconstruction run on GPU with a Cellular Automaton algorithm (>40 tunable parameters)
  Three main steps:
  1. **doublets**: pairs of compatible hits
  2. **n-Tuplets**: compatible hit pairs sharing a hit
  3. **Fitting** and selection
- Tuning is complex and time consuming

Parameters

Compatibility checked (mainly) via geometrical cuts on the angle of the trajectory:

- 19 Cuts on Φ
- Cut on z0
- Hard Curvature Cut
- DCA in the inner layers
- DCA in the outer layers
- Cut on θ in Barrel region
- Cut on θ in Forward region

25 parameters in total

Metrics

Reco track is **associated** to Sim if >75% of hits originate from it

Otherwise is considered fake

- Efficiency: $\frac{N_{ass(reco \to sim)}}{N_{sim}}$
- Fake Rate: $\frac{N_{rec} - N_{ass(sim \to reco)}}{N_{rec}}$

Doublet Cuts

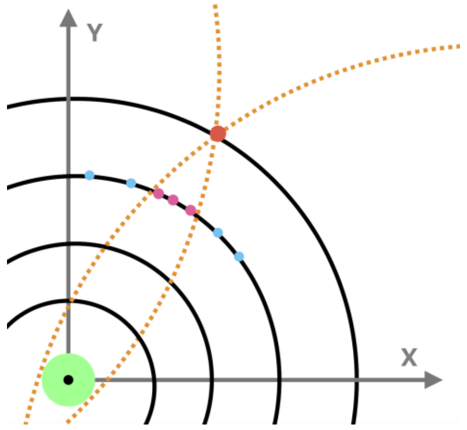


Fig 1

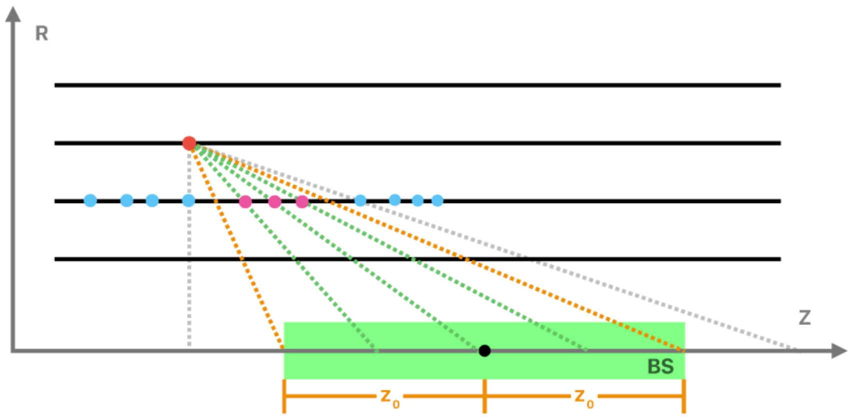


Fig 2

Doublet pairs built from outer layer:

- ● Outer layer hit
- ● Inner hits
- ● Compatible hits

1. ϕ Cuts: windows in ϕ in $X - Y$
   One per layer pair

2. z0 Cut: window in Z in $R - Z$

CMS DP-2024/084

Triplet Cuts

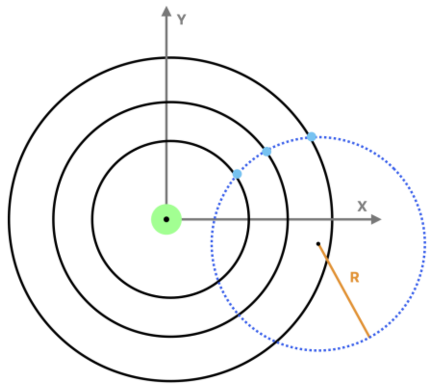
Fig 3a


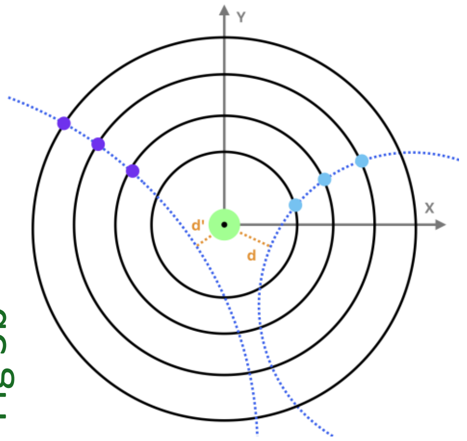
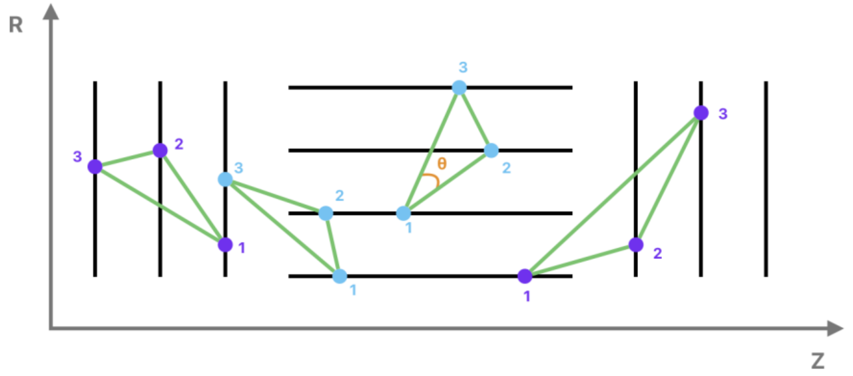
Fig 3b


Fig 4


Triplets are built from doublets sharing a hit

3a. HardCurvCut: cut on minimum p_T

3b. DCA Cut: cut on maximum
Distance of Closest Approach

4. θ Cut: cuts on maximum θ
One for Barrel, one for forward

CMS DP-2024/084

The Optimizer

Framework to support multiple optimization algorithms

Common interface to the user

Support any objective function

Optimization can be done all in one go or step by step

Support stop and resume

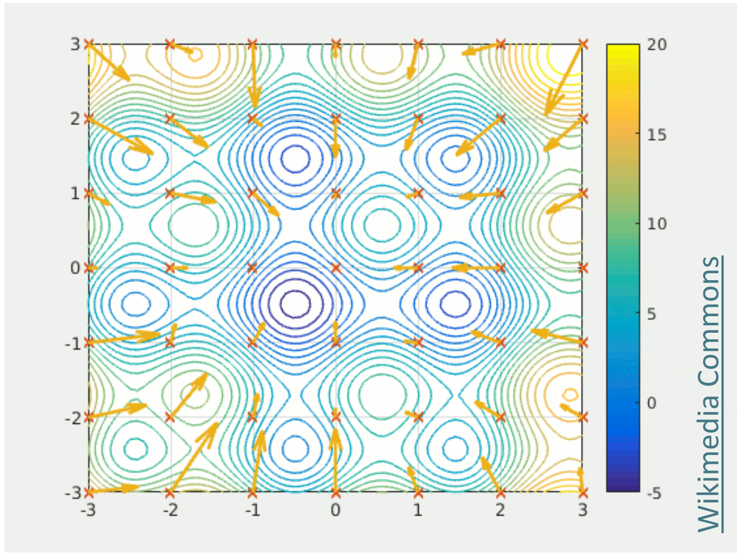
```
import optimizer

def f1(x):
    return x**2

def f2(x):
    subprocess.run(["cmsRun",...])
    eval = get_metrics(uproot.open(...))
    return eval

objectives = optimizer.Objective([f1, f2])
mopso = optimizer.MOPSO(objectives,
                        lower_bounds = [...],
                        upper_bounds = [...],
                        num_particles, ...)
mopso.optimize(num_iterations)
print(mopso.pareto_front)
```

https://github.com/cms-patatrack/The-Optimizer

Particle Swarm Optimization

PSO is a **population-based** optimization technique inspired by the social behaviour of birds flocking or fish schooling

Agents move through the parameters space, adjusting their positions based on local and global best position



- $v_{id}^{t+1} = w \times v_{id}^{t} + c_1 \times r_1^t \times (p_{id}^t - x_{id}^t) + c_2 \times r_1^t \times (g_d^t - x_{id}^t)$
- $x_{id}^{t+1} = x_{id}^t + v_{id}^t$

w, c_1, and c_2 regulate **exploration** (finding new solutions) and **exploitation** (fine-tuning local solution)

Multi-Objective Optimization

Our problem requires optimizing two objective function at once (efficiency and fake rate)

An adaptation of PSO for multi-objective problems (two or more)

Conflicting Objectives: two or more objectives cannot be optimized simultaneously w/o compromising one or more of the other

There is **no single optimal solution**, but a **set of trade-off solutions**

Maintains a set of solutions (Pareto front) each iteration

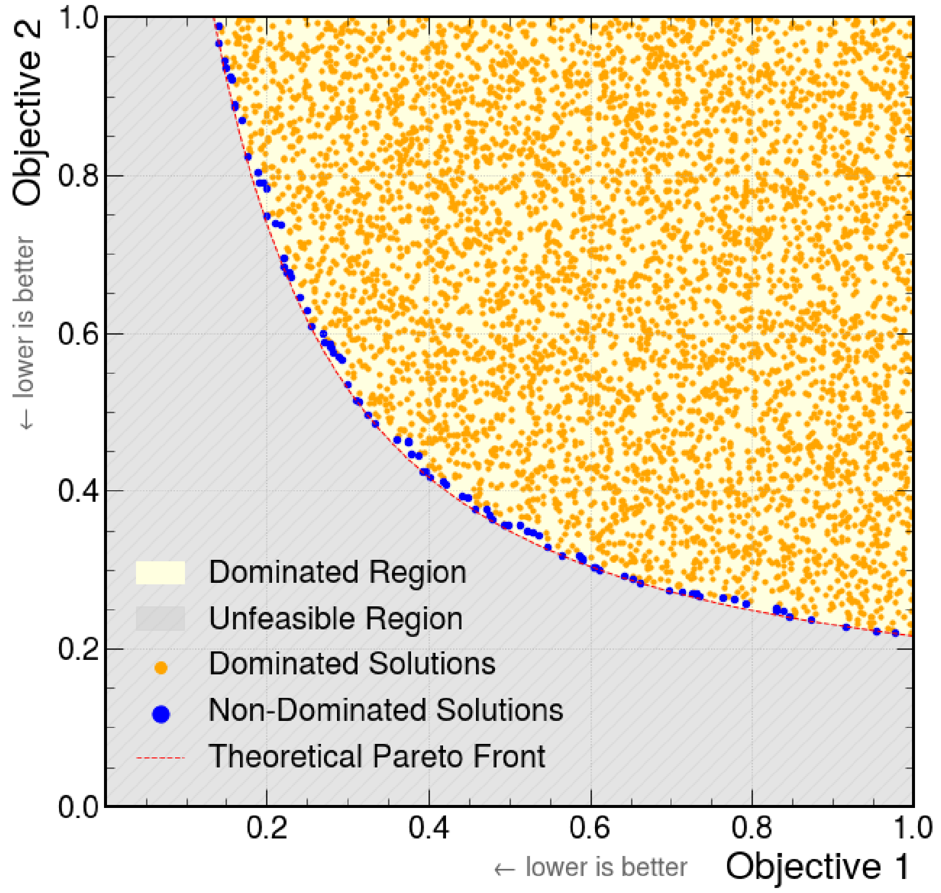
Global and local best determined taken from collection of non-dominated solutions

Pareto Front

Set of **non-dominated** solutions in multi-objective optimization

Divides the solution space into a **dominated region** and an **unfeasible region**

A dominates *B* if is equal or better than *B* in all objectives and strictly better in at least one

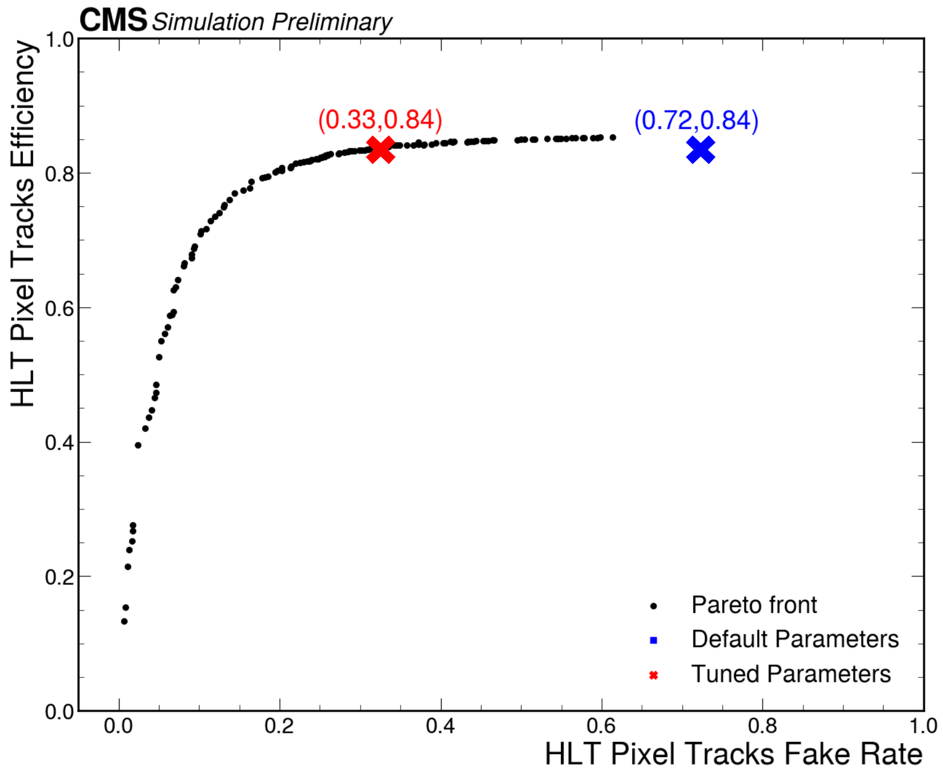


Optimization results

Simulated $t\bar{t}$ sample (100 events) with superimposed pileup
PU range $[\sim\mathbf{15}; \sim\mathbf{85}]$ with $< \mathbf{PU} > \ \sim\mathbf{50}$
to mimic early 2023 conditions

The Pareto front (in black) obtained running the optimization with:

- 25 parameters
- **total** pixel tracks **efficiency** and **fake rate** as objective
- 150 iterations (**~5 h**)
- 200 agents
- $w, c_1, c_2 \ = 1$



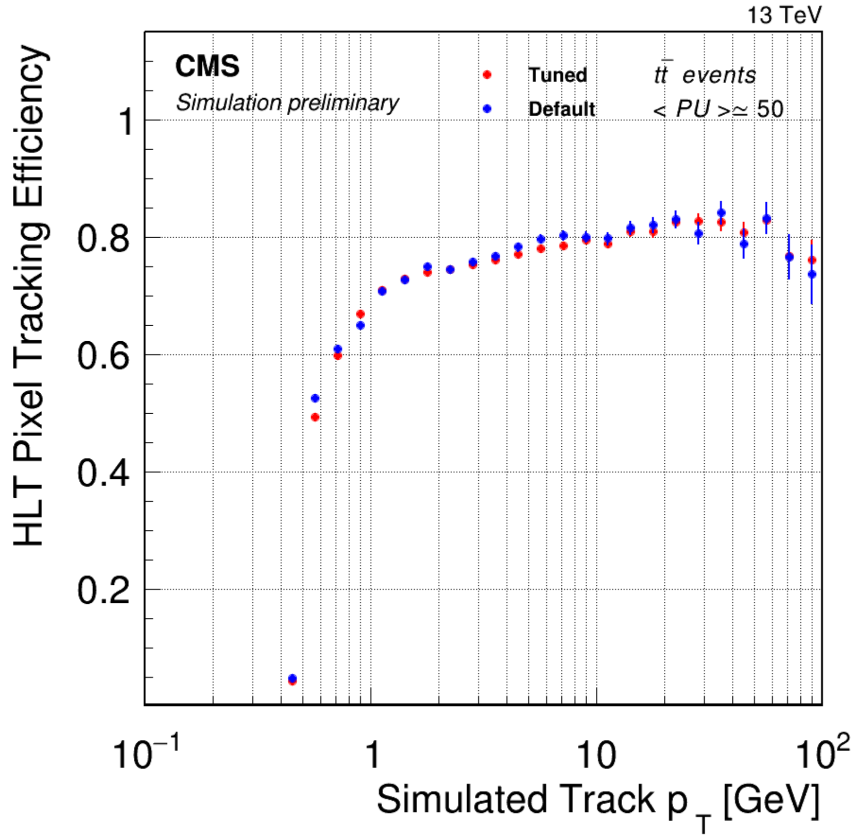
Same efficiency, ~half the fake rate

CMS DP-2024/084

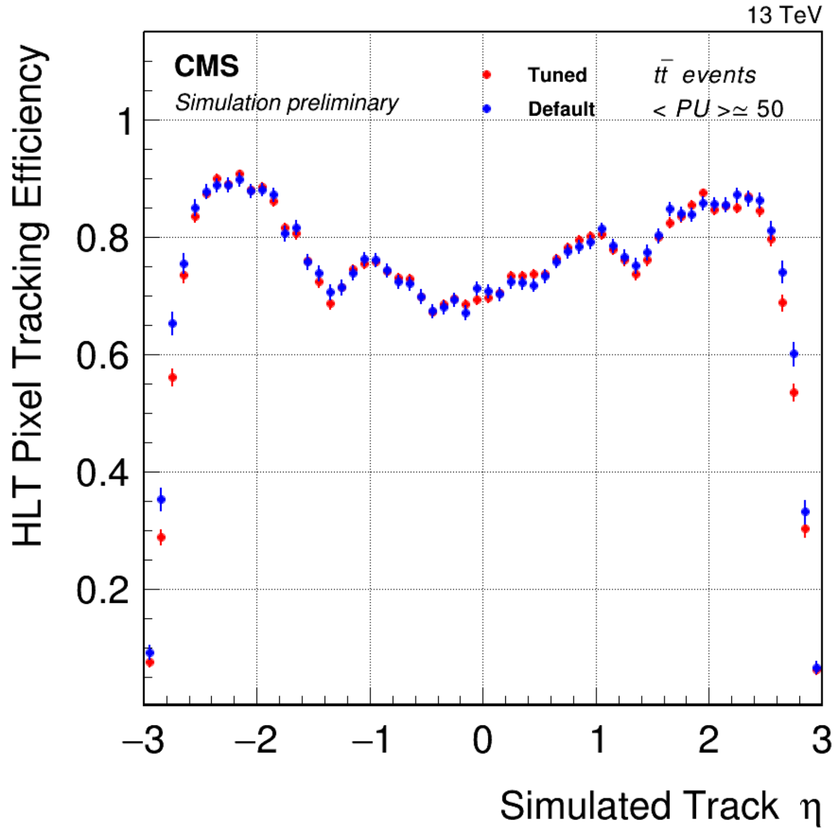
Efficiency

HLT pixel tracking efficiency for the default HLT pixel tracks (blue) and the tuned HLT pixel tracks (red)

Efficiency roughly the same as expected



Efficiency vs sim tracks $\mathbf{p_T}$ with $\mathbf{p_T > 0.4\ GeV}$ and $|\boldsymbol{\eta}| < \mathbf{3.0}$



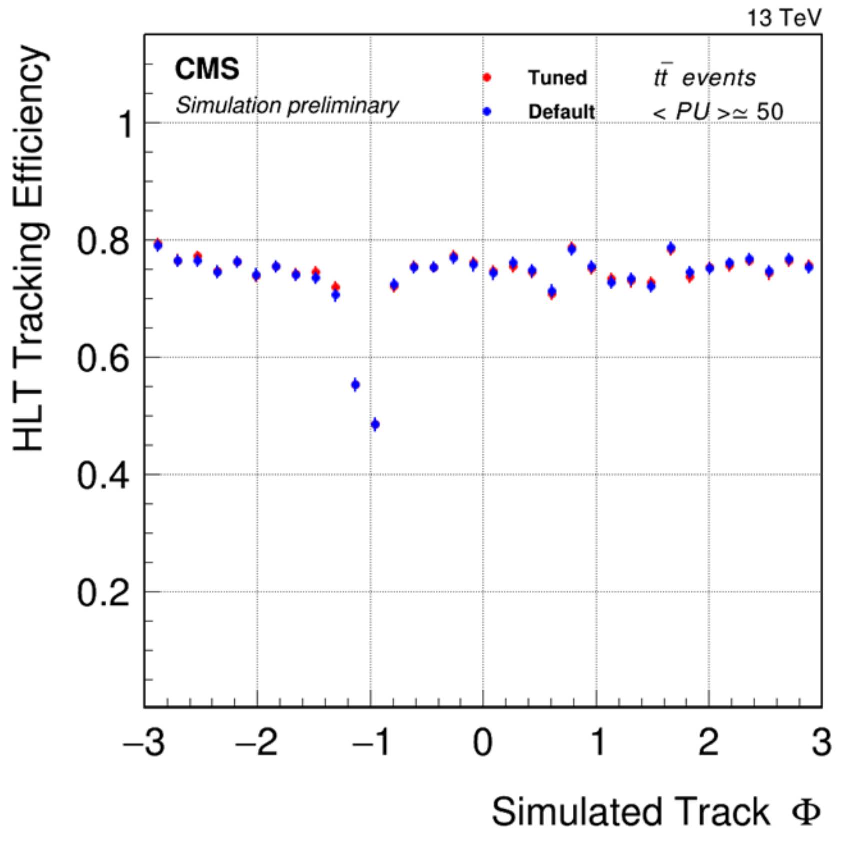
Efficiency vs sim tracks $\boldsymbol{\eta}$ with $\mathbf{p_T > 0.9\ GeV}$ and $|\boldsymbol{\eta}| < \mathbf{3.0}$

CMS DP-2024/084

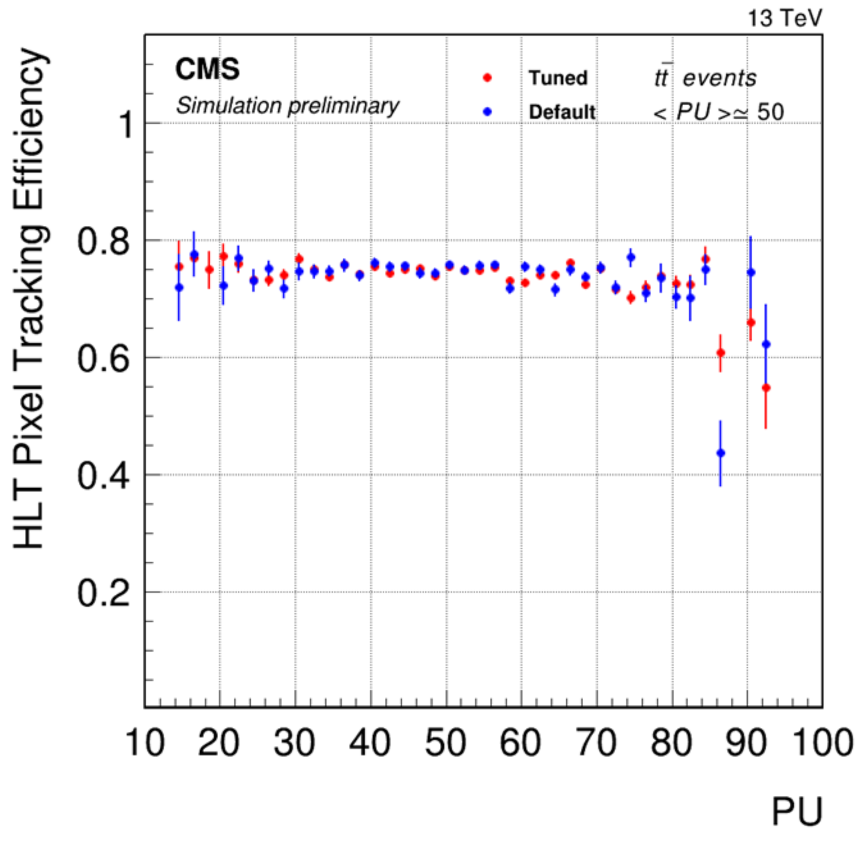
Efficiency (contd.)

HLT pixel tracking efficiency for the default HLT pixel tracks (blue) and the tuned HLT pixel tracks (red)

Efficiency roughly the same as expected



Efficiency vs sim tracks ϕ
with $p_T > 0.9$ GeV and $|\eta| < 3.0$
Drop in the region around $\phi = -1$ for the masking (simulated) of BPix3 and BPix4 modules in 2023 (see backup)



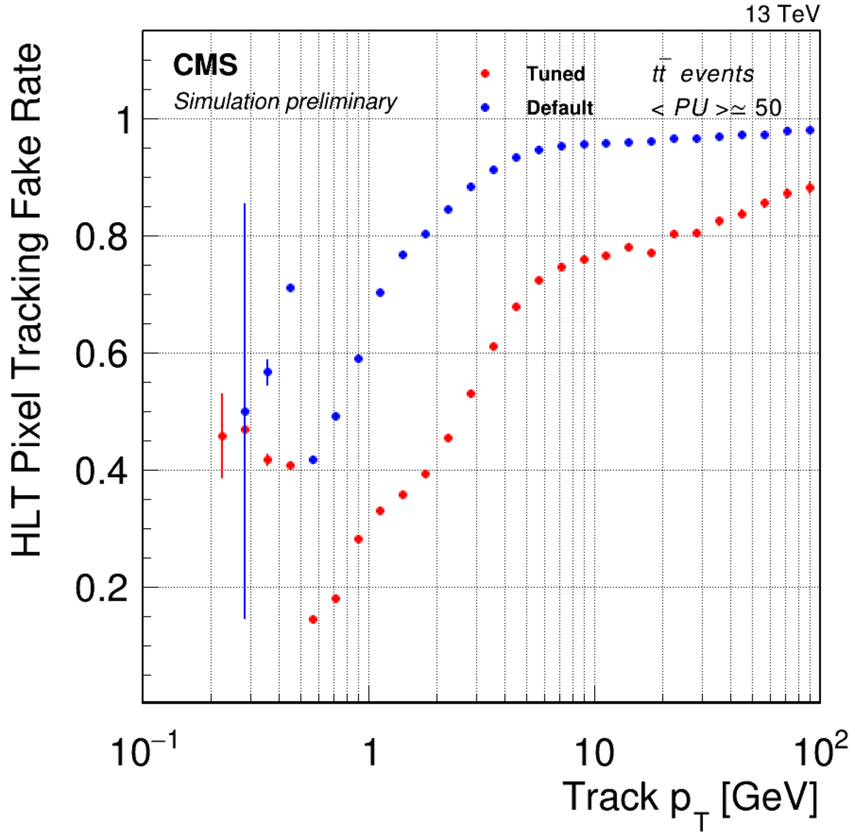
Efficiency vs sim tracks PU
with $p_T > 0.9$ GeV and $|\eta| < 3.0$

CMS DP-2024/084

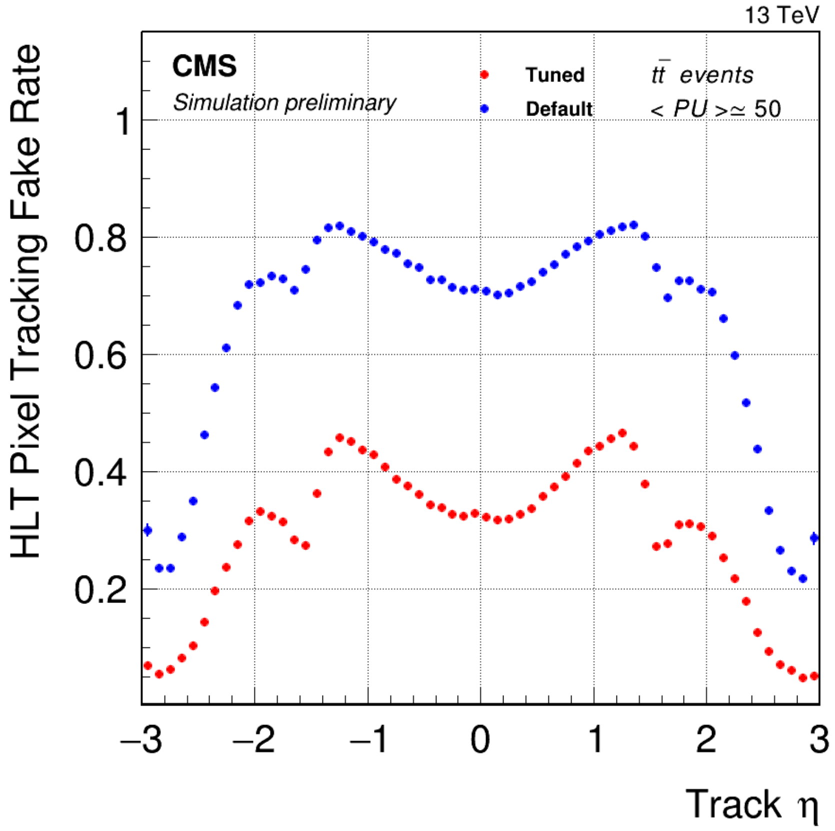
Fake rates

HLT pixel tracking fake rates for the default HLT pixel tracks (blue) and the tuned HLT pixel tracks (red)

~2x Fake Rate reduction, as expected



Fake rate vs sim tracks $\mathbf{p_T}$



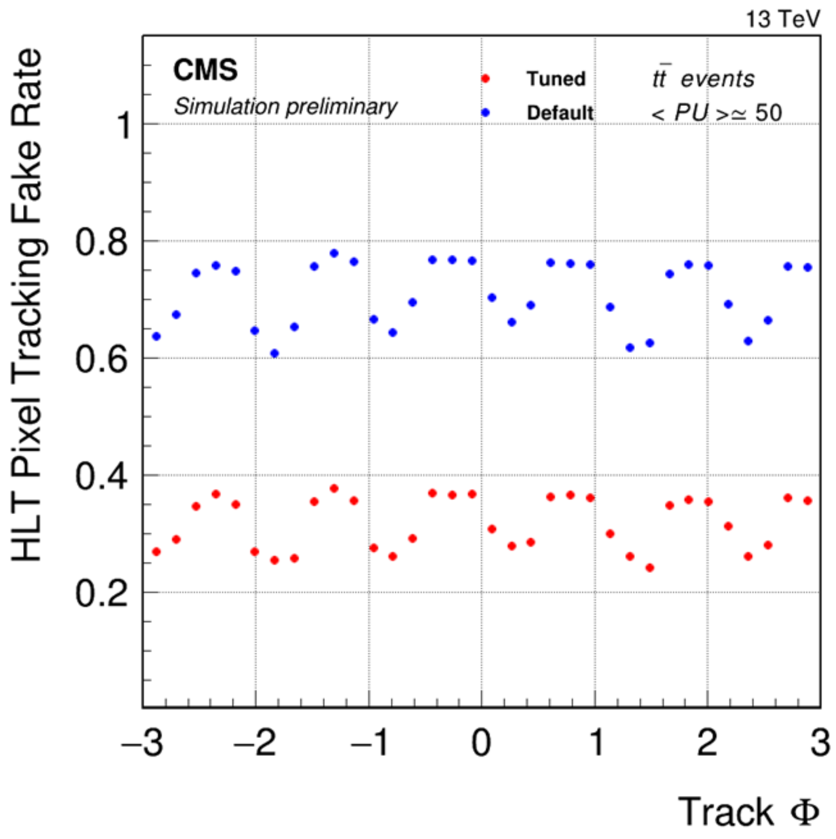
Fake rate vs sim tracks $\boldsymbol{\eta}$

CMS DP-2024/084

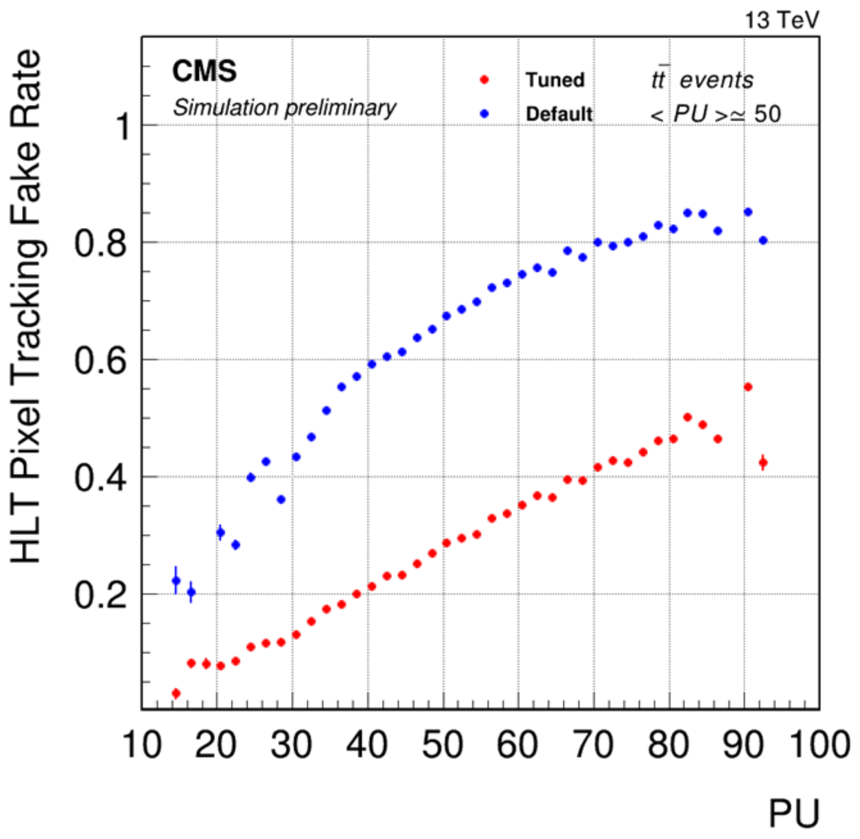
Fake rates (contd.)

HLT pixel tracking fake rates for the default HLT pixel tracks (blue) and the tuned HLT pixel tracks (red)

~2x Fake Rate reduction, as expected



Fake rate vs sim tracks ϕ
Oscillating behaviour due to overlap on pixel modules



Fake rate vs sim tracks $\mathbf{PU}$

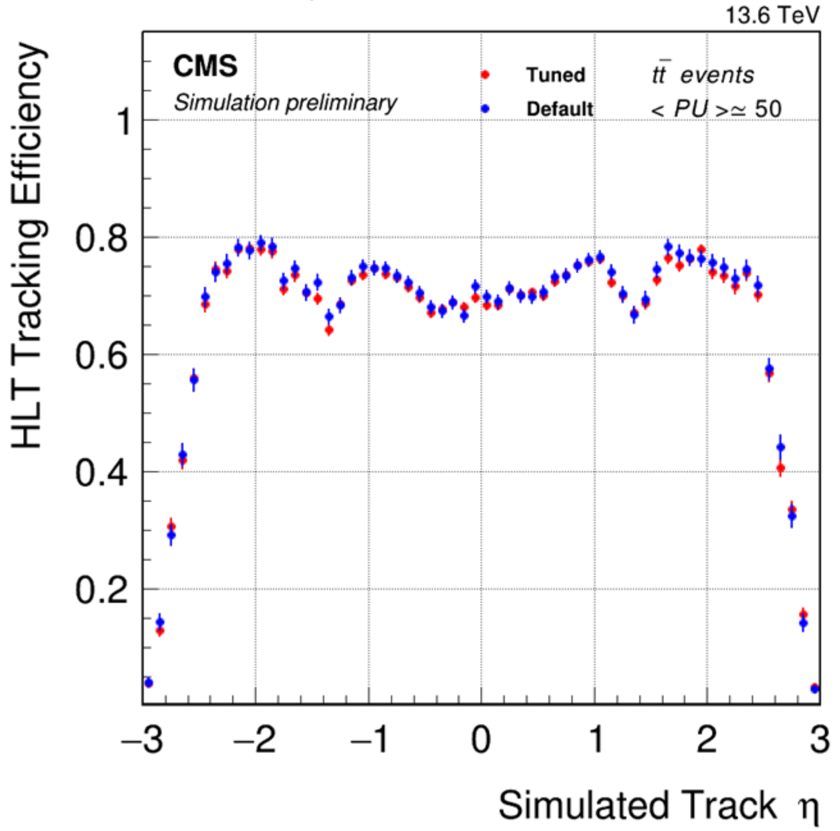
CMS DP-2024/084

Full HLT Tracks

Pixel tracks used as seed for Full tracks (pixel+strip)

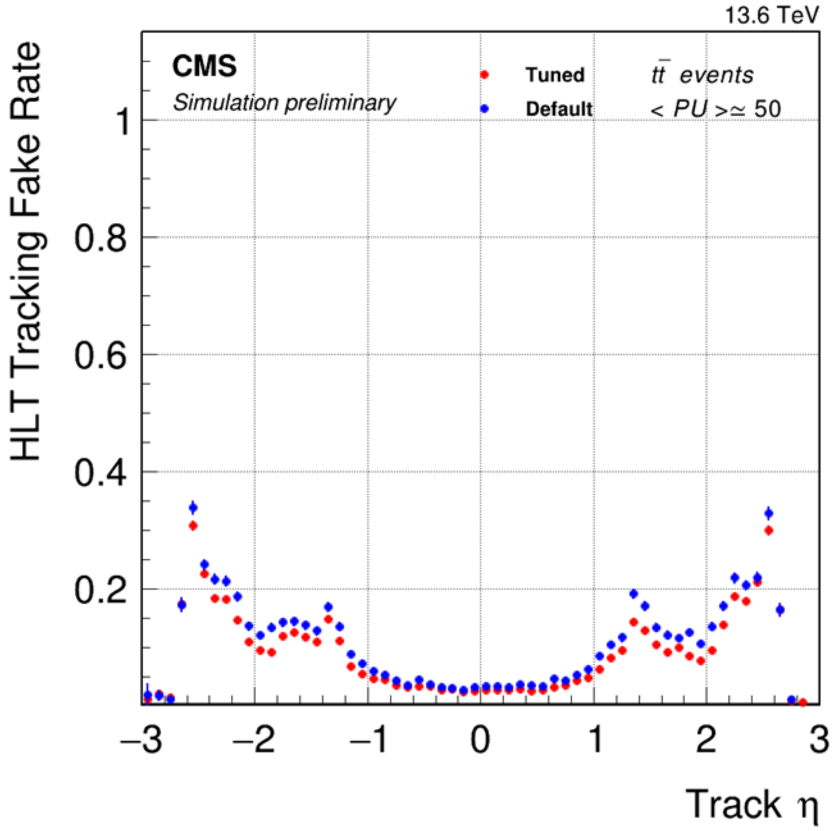
Full track quality selection applied:
Larger reduction of fake rate on seeds leads to smaller reduction of fake rate on full tracks

Efficiency



Efficiency vs sim tracks η with $p_T > 0.9\ \mathrm{GeV}$ and $|\eta| < 3.0$

Fake Rate



CMS DP-2024/084

Results

- **Tuned** results have **compatible efficiency** WRT **default**
- **Fake rate is reduced by 50% for pixel tracks.**
- Almost identical performance for Full Tracks
- Preliminary timing measurement on >10000 events
  8 concurrent jobs, 32 threads 24 streams
  - 2 x AMD EPYC 7763 “Milan” CPUs
  - 256 GB of system memory
  - 2 x NVIDIA T4 GPUs

Sample late 2023 Run $< \mathrm{PU} > \cong 61.5$

Speedup of ~9.5% for pixel track reconstruction
Speedup of ~5.5% for the full HLT online reconstruction

Other use-cases

- Tested for mkFit (CMS-DP-2022-018) scoring and filtering on Run3 results. Plan to use for Phase 2
- Initial work on Line Segment Tracking (CMS-DP-2024-014) reconstruction optimization. Preliminary results shown at 16th Patatrack Hackathon.
- Application of MOPSO to TICL (Talk on Tuesday by W. Redjeb) in HGCAL. To be extended with the reconstruction in the Barrel region.
- Testing Pixeltrack on TrackML (different detector geometry) with promising results
- Testing Reinforcement Learning algorithm to learn when to avoid "useless" evaluation of the objective functions

Conclusion

MOPSO allows parameters tuning in short amount of time

Preliminary results for HLT pixel tracking are compatible with validation

Optimization result in a **~continuous set of solutions**

Same optimizer can be used for different use cases

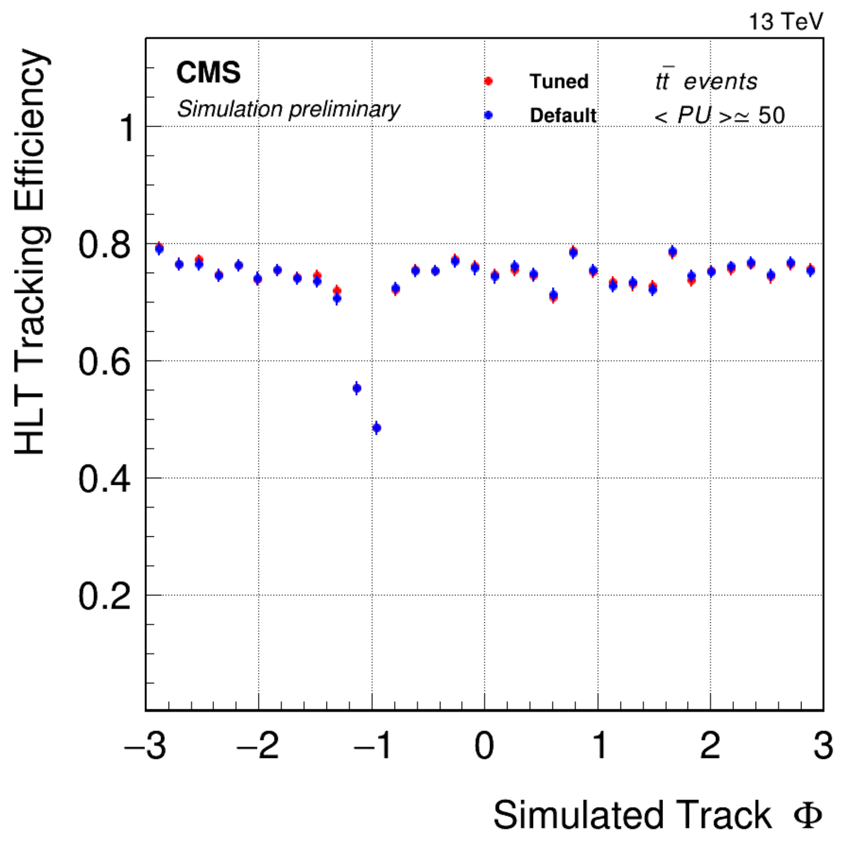
Optimization can be used to adapt algorithms to new problems without rewriting them

If interested contact me or the CMS Patatrack team

https://github.com/cms-patatrack/The-Optimizer

Thank you

Detector conditions



The detector conditions are simulated to include an issue affecting the Barrel Pixel Layers 3 and 4 (BPix3 and BPix4) since TS1 in 2023

27 modulesin BPix3 and BPix4 became inoperable due to a problem distributing the LHC clock to the modules

These modules have been turned off since this incident

To better cope with this issue an additional pixel doublets recovery iteration has been added to the HLT tracking sequence

CMS DP-2024/084

Default vs Tuned

	default	tuned	Relative change
CAThetaCutBarrel	0.002	0.001180	-41%
CAThetaCutForward	0.003	0.001997	-33%
hardCurvCut	0.0328407225	0.0302418577150	-8%
dcaCutInnerTriplet	0.15	0.599431310369406556	+300%
dcaCutOuterTriplet	0.25	0.373806768941585710	+50%
cellZ0Cut	12.0	8.846	-26%
phiCuts	[522, 730, 730, 522, 626, 626, 522, 522, 626, 626, 626, 522, 522, 522, 522, 522, 522, 522, 522]	[661, 598, 595, 718, 707, 858, 656, 832, 677, 411, 650, 767, 703, 682, 637, 601, 675, 745, 797]	[+27%, -18%, -18%, +38%, +13%, +37%, +26%, +59%, +8%, -34%, +4%, +47%, +35%, +31%, +22%, +15%, +29%, +43%, +53%]