Improvements in the Cellular Automaton

Felice Pantaleo

EP-CMG-CO

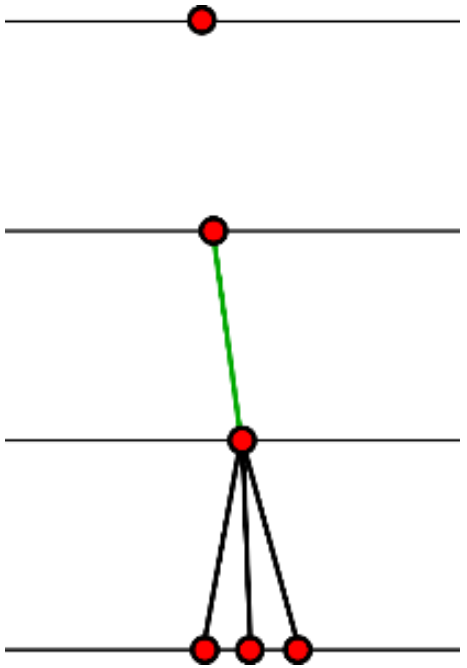
Outline

- Fake rate: xy plane cut, hard pt cut
- Timing: all in one
- Future plans

Cells creation and connection

- At the moment a Cell is constructed using the legacy Doublet Generator
  - given a region (pT_min, beamspot, LIP, TIP..) it matches two hits from different layers if they are compatible with the region
  - in the near future a cell will be constructed when a doublet is found

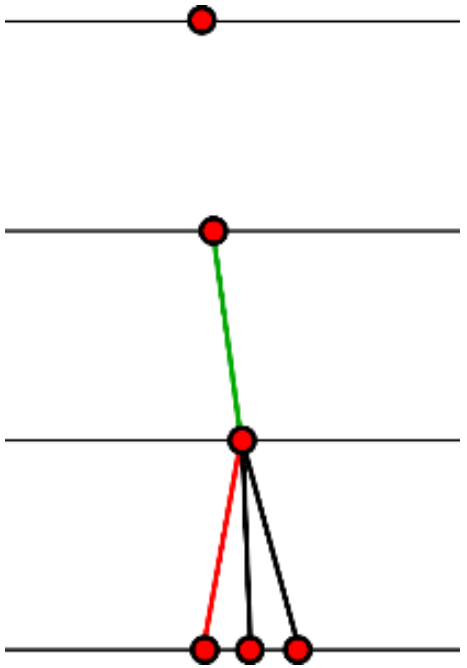
- When a Cell is created, the compatibility with all the cells in the previous layer pair that are sharing the same outer hit is checked



Cells creation and connection

- At the moment a Cell is constructed using the legacy Doublet Generator
  - given a region (pT_min, beamspot, LIP, TIP..) it matches two hits from different layers if they are compatible with the region
  - in the near future a cell will be constructed when a doublet is found

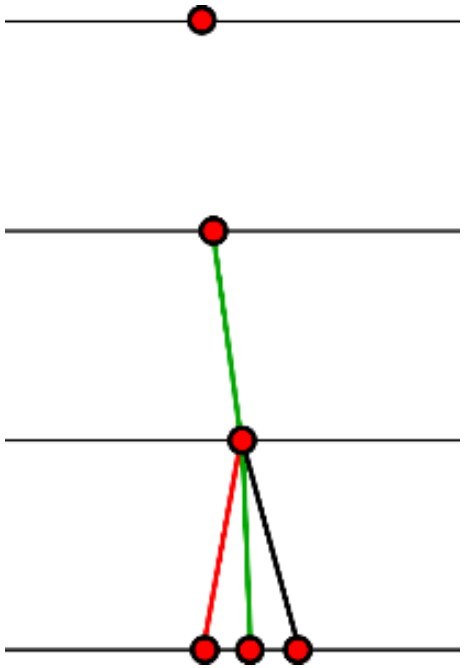
- When a Cell is created, the compatibility with all the cells in the previous layer pair that are sharing the same outer hit is checked



Cells creation and connection

- At the moment a Cell is constructed using the legacy Doublet Generator
  - given a region (pT_min, beamspot, LIP, TIP..) it matches two hits from different layers if they are compatible with the region
  - in the near future a cell will be constructed when a doublet is found

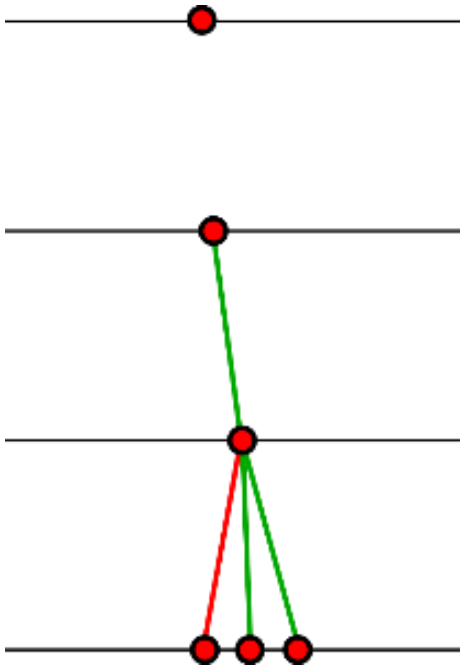
- When a Cell is created, the compatibility with all the cells in the previous layer pair that are sharing the same outer hit is checked



Cells creation and connection

- At the moment a Cell is constructed using the legacy Doublet Generator
  - given a region (pT_min, beamspot, LIP, TIP..) it matches two hits from different layers if they are compatible with the region
  - in the near future a cell will be constructed when a doublet is found

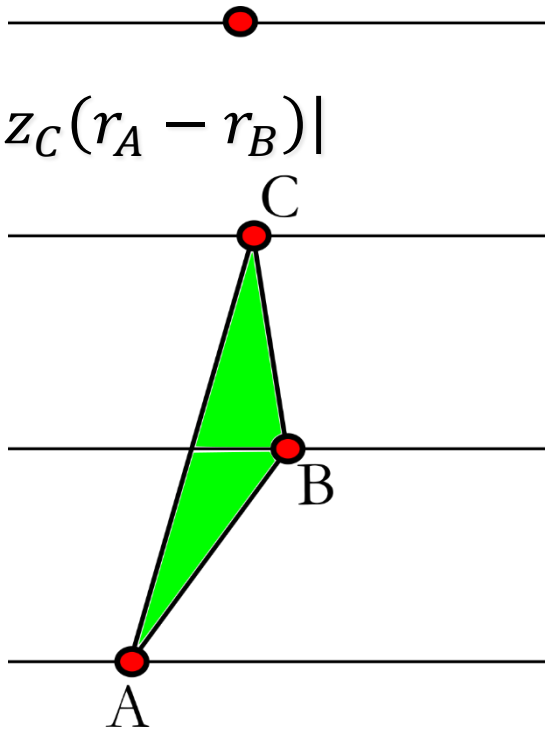
- When a Cell is created, the compatibility with all the cells in the previous layer pair that are sharing the same outer hit is checked



Compatibility

- The area of the triangle ABC in the RZ plane is given by:

$$A = |z_A(r_B - r_C) + z_B(r_C - r_A) + z_C(r_A - r_B)|$$



Hence the tangent of the angle in A is given by:

$$tg(\vartheta) = 2A/d_{AC}^2 \rightarrow \vartheta$$

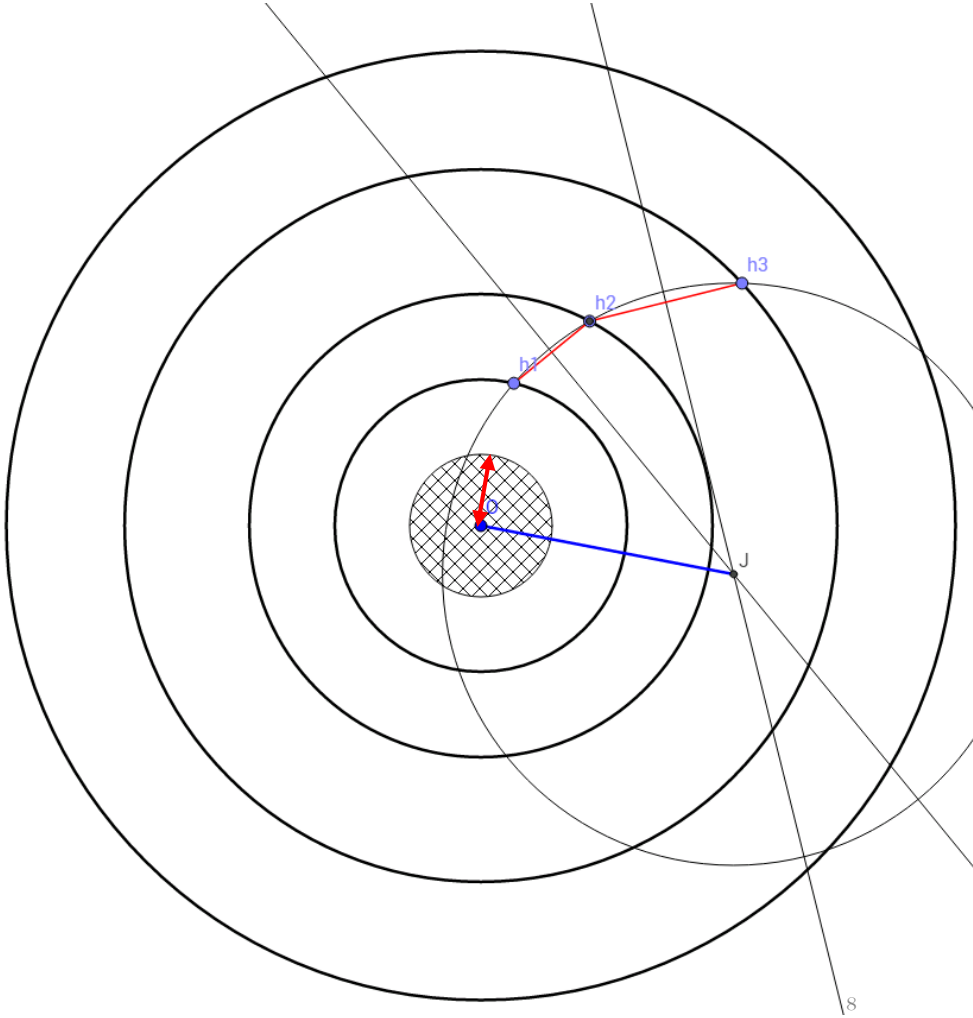
$$\theta_0 = \frac{13.6MeV}{\beta cp} z_{ch} \sqrt{\frac{t}{X_0}} \left[1 + 0.038 \ln\left(\frac{t}{X_0}\right)\right]$$

ϑ * p_{min} < cut

- If two cells are found compatible they are pushed in each others' outer and inner neighbors vectors

Compatibility

- Intersection between perpendicular bisectors of the two cells is found.
- **Radius** of the circle is then found
- No need to know where this circle and the circle given by (center=beamspot, radius = **TIP**) intersect
- They intersect if the distance between the centers d(c1,c2) satisfies:
  r1-r2 < d(c1,c2) < r1+r2
- **Hard pT cut:**
  - **If the triplet's radius is less than a threshold(pTmin), the triplet is discarded**
  - **default 0 GeV/c**



All-in-one

- The quadruplet generator was taking sets of 4 layers and run a different CA for each layer set

```
layerList = cms.vstring(
'BPix1+BPix2+BPix3+BPix4',
'BPix1+BPix2+BPix3+FPix1_pos',
'BPix1+BPix2+BPix3+FPix1_neg',
'BPix1+BPix2+FPix1_pos+FPix2_pos',
'BPix1+BPix2+FPix1_neg+FPix2_neg',
'BPix1+FPix1_pos+FPix2_pos+FPix3_pos',
'BPix1+FPix1_neg+FPix2_neg+FPix3_neg'
)
```

This would result in many doublets, checks, evolutions run twice (or more)

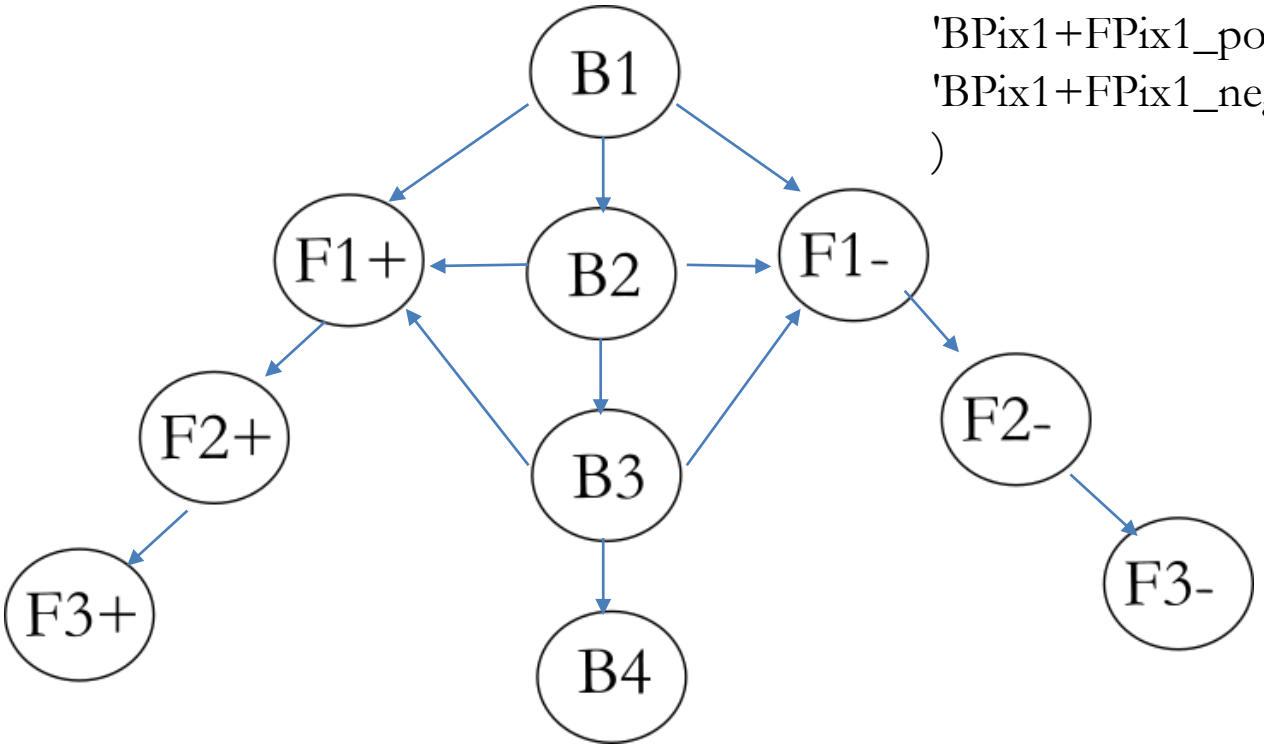
All-in-one ctd.

- In order to run only one CA for all the layer combinations, the hard dependency on the number of layers (as template parameter and in loops), had to be removed
- CAGraph was introduced to store the connections and the ordering between layers
- Given the input string from the Configuration it builds:
  - Layer Graph (vertices visitor)
  - Layer Pair Graph (edges visitor)
- Applied to out layer list it would result in...

CAGraph - CALayer

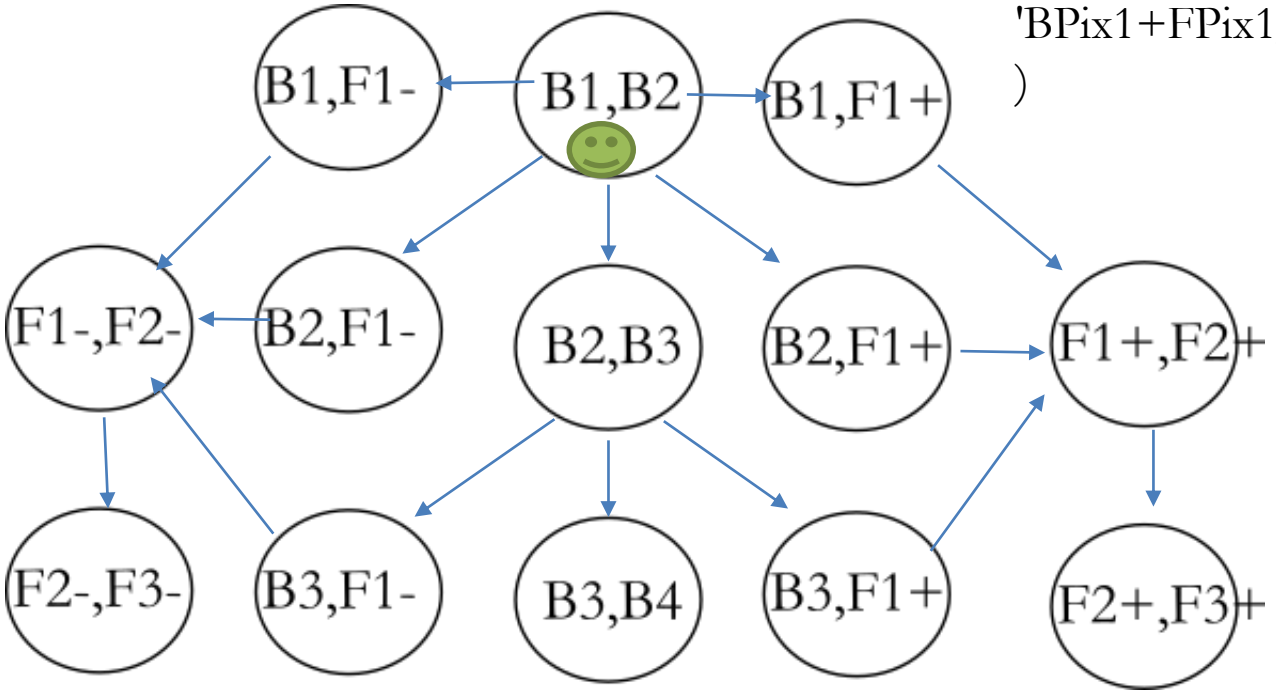
For each hit on the layer, pointers to cells having that his as outer hit

```
layerList = cms.vstring(
'BPix1+BPix2+BPix3+BPix4',
'BPix1+BPix2+BPix3+FPix1_pos',
'BPix1+BPix2+BPix3+FPix1_neg',
'BPix1+BPix2+FPix1_pos+FPix2_pos',
'BPix1+BPix2+FPix1_neg+FPix2_neg',
'BPix1+FPix1_pos+FPix2_pos+FPix3_pos',
'BPix1+FPix1_neg+FPix2_neg+FPix3_neg'
)
```



CAGraph - CALayerPair

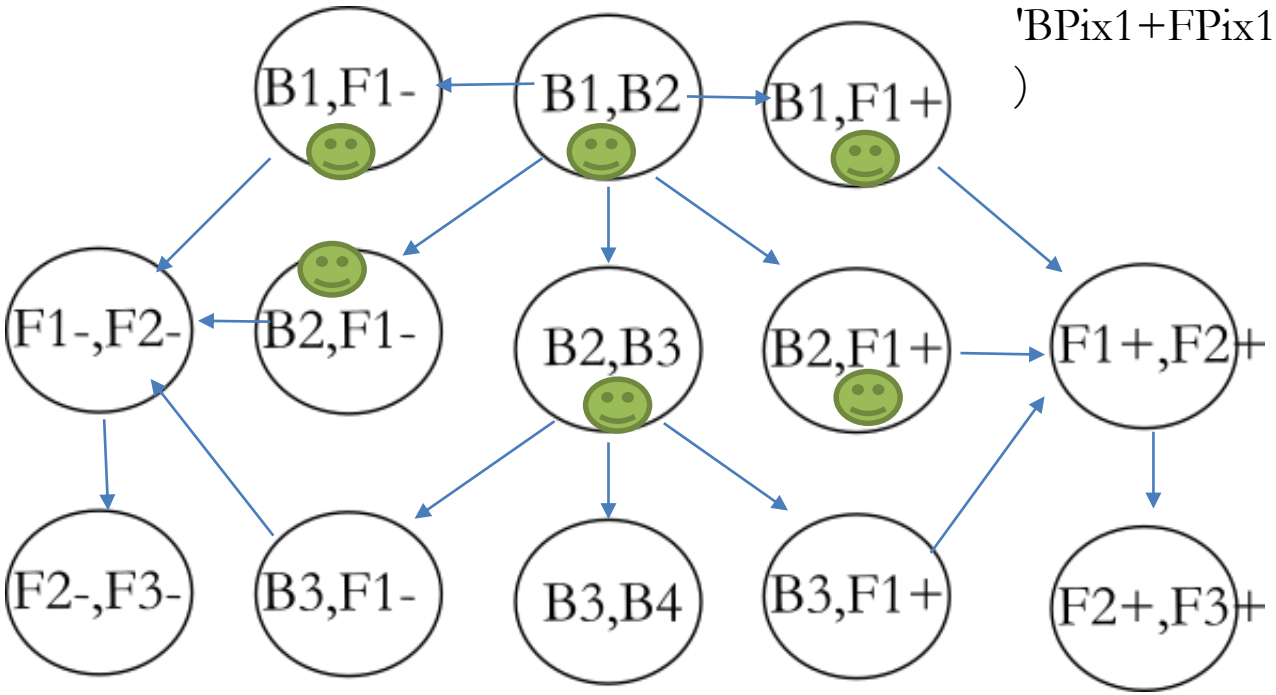
Cells are stored in a CALayerPair and are evaluated once.
Cell construction, matching, evolution is done using a BFS on this graph.




```
layerList = cms.vstring(
'BPix1+BPix2+BPix3+BPix4',
'BPix1+BPix2+BPix3+FPix1_pos',
'BPix1+BPix2+BPix3+FPix1_neg',
'BPix1+BPix2+FPix1_pos+FPix2_pos',
'BPix1+BPix2+FPix1_neg+FPix2_neg',
'BPix1+FPix1_pos+FPix2_pos+FPix3_pos',
'BPix1+FPix1_neg+FPix2_neg+FPix3_neg'
)
```

CAGraph - CALayerPair

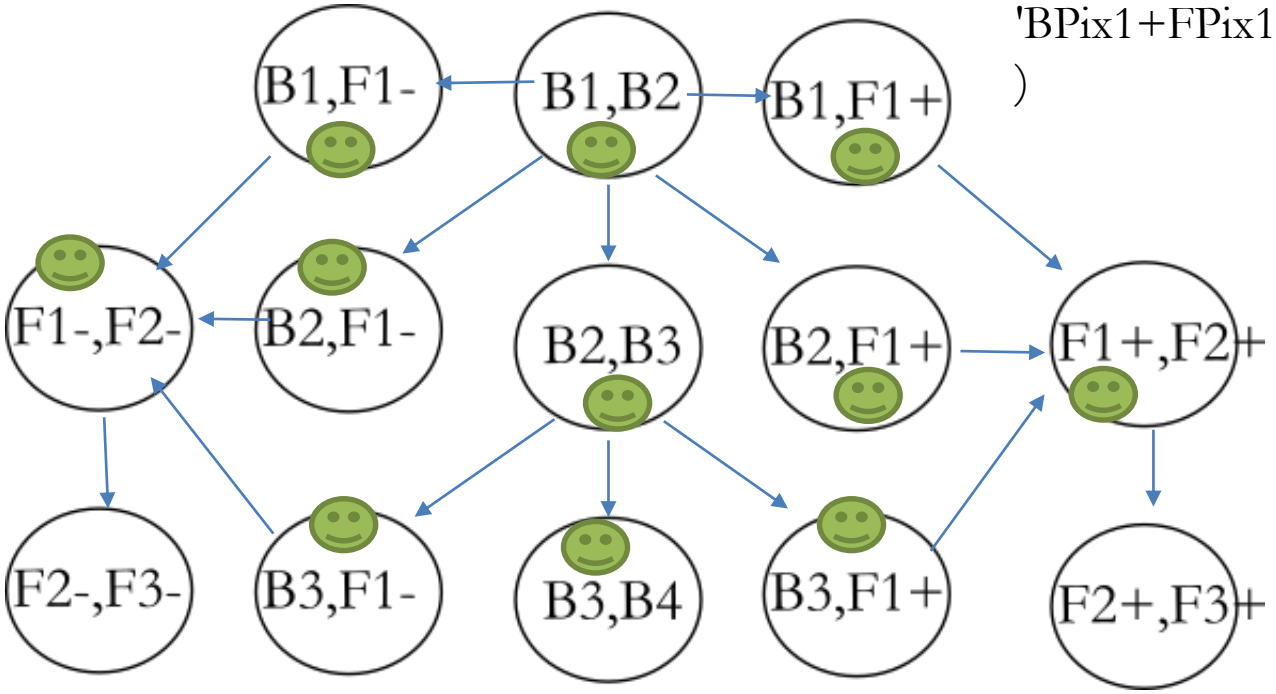
Cells are stored in a CALayerPair and are evaluated once.
Cell construction, matching, evolution is done using a BFS on this graph.




```
layerList = cms.vstring(
'BPix1+BPix2+BPix3+BPix4',
'BPix1+BPix2+BPix3+FPix1_pos',
'BPix1+BPix2+BPix3+FPix1_neg',
'BPix1+BPix2+FPix1_pos+FPix2_pos',
'BPix1+BPix2+FPix1_neg+FPix2_neg',
'BPix1+FPix1_pos+FPix2_pos+FPix3_pos',
'BPix1+FPix1_neg+FPix2_neg+FPix3_neg'
)
```

CAGraph - CALayerPair

Cells are stored in a CALayerPair and are evaluated once.
Cell construction, matching, evolution is done using a BFS on this graph.

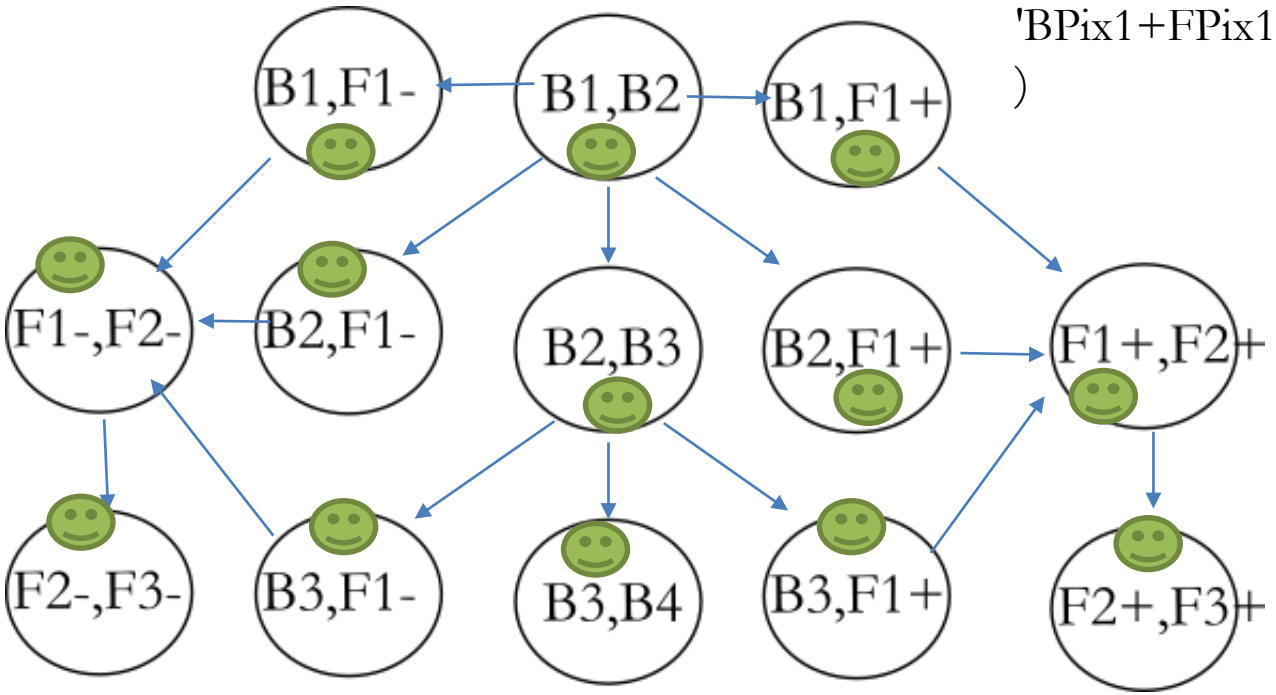



```
layerList = cms.vstring(
'BPix1+BPix2+BPix3+BPix4',
'BPix1+BPix2+BPix3+FPix1_pos',
'BPix1+BPix2+BPix3+FPix1_neg',
'BPix1+BPix2+FPix1_pos+FPix2_pos',
'BPix1+BPix2+FPix1_neg+FPix2_neg',
'BPix1+FPix1_pos+FPix2_pos+FPix3_pos',
'BPix1+FPix1_neg+FPix2_neg+FPix3_neg'
)
```

CAGraph - CALayerPair

Cells are stored in a CALayerPair and are evaluated once.
Cell construction, matching, evolution is done using a BFS on this graph.

```
layerList = cms.vstring(
'BPix1+BPix2+BPix3+BPix4',
'BPix1+BPix2+BPix3+FPix1_pos',
'BPix1+BPix2+BPix3+FPix1_neg',
'BPix1+BPix2+FPix1_pos+FPix2_pos',
'BPix1+BPix2+FPix1_neg+FPix2_neg',
'BPix1+FPix1_pos+FPix2_pos+FPix3_pos',
'BPix1+FPix1_neg+FPix2_neg+FPix3_neg'
)
```



Filtering

- Approximate independent fits
  - in the R-z plane, straight line + bending corrections
  - in the x-y plane, circumference
- Reject quadruplets whose chi2 exceeds a threshold

Performance

- See other attachment

Conclusion

- A solid and performant version of the CA is now being pushed in the release
  - https://github.com/cms-sw/cmssw/pull/15751
- Although it would have been easier to implement graphs and manipulate strings using pointers, std::set, std::maps etc, this is not portable to CUDA. Everything was implemented using integral indeces and std::vectors
- I'll work until the end of the month to port all this new implementation to CUDA, and update the hackaton branch
  - show the results at CHEP
- Run this CUDA+CMSSW prototype on different architectures
  - Minsky (NVIDIA Pascal P100+ PPC)
  - show the results at CHEP