Global VDET Pattern Recognition for ALEPH

Andrew Bazarko, Jean-François Pusztaszeri,
Paul E. Rensing

European Laboratory for Particle Physics (CERN),
1211 Genève 23, Switzerland

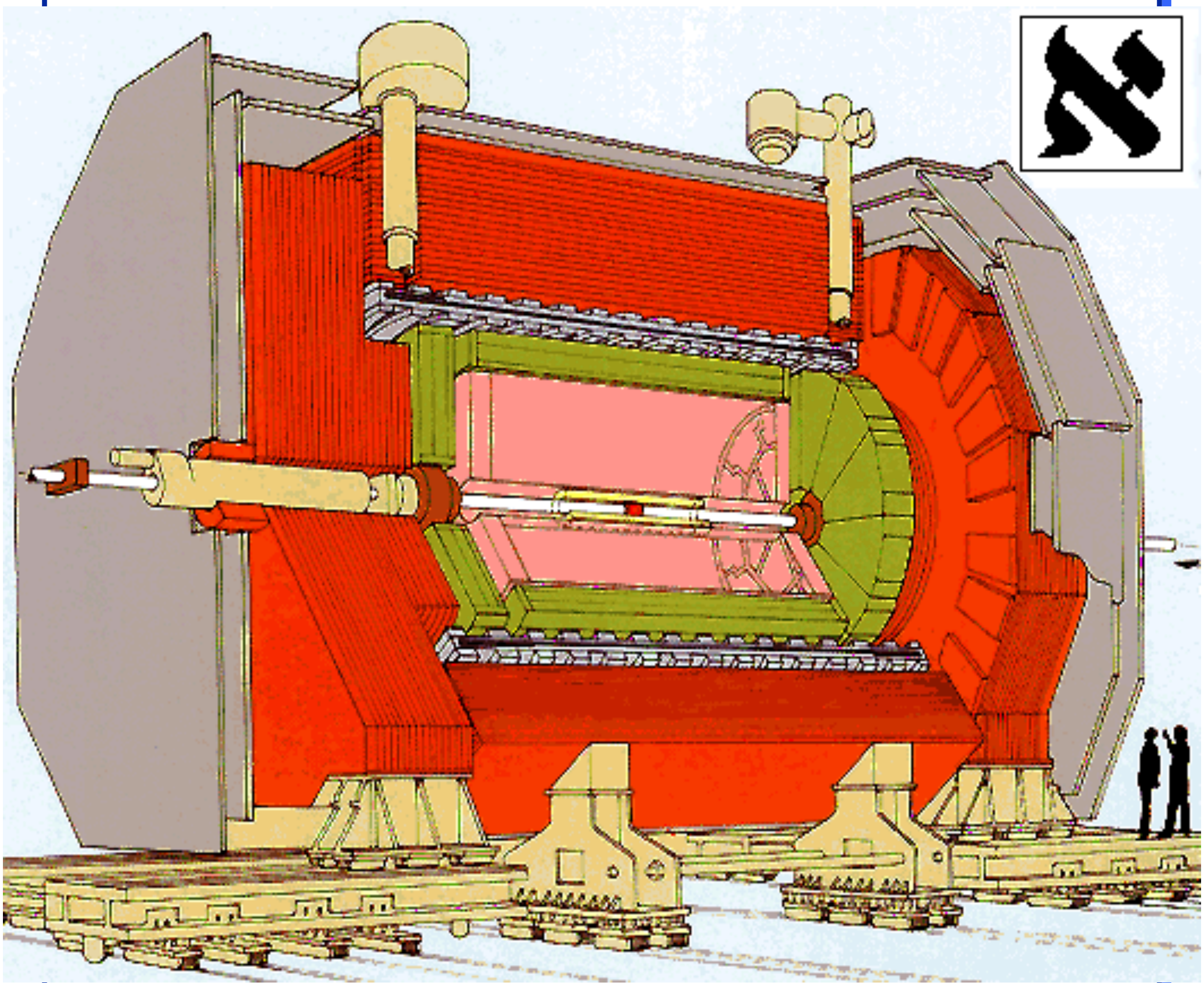
David Brown

Max-Planck-Institut für Physik, Werner-Heisenberg-Institut,
80805 München, Fed. Rep. of Germany

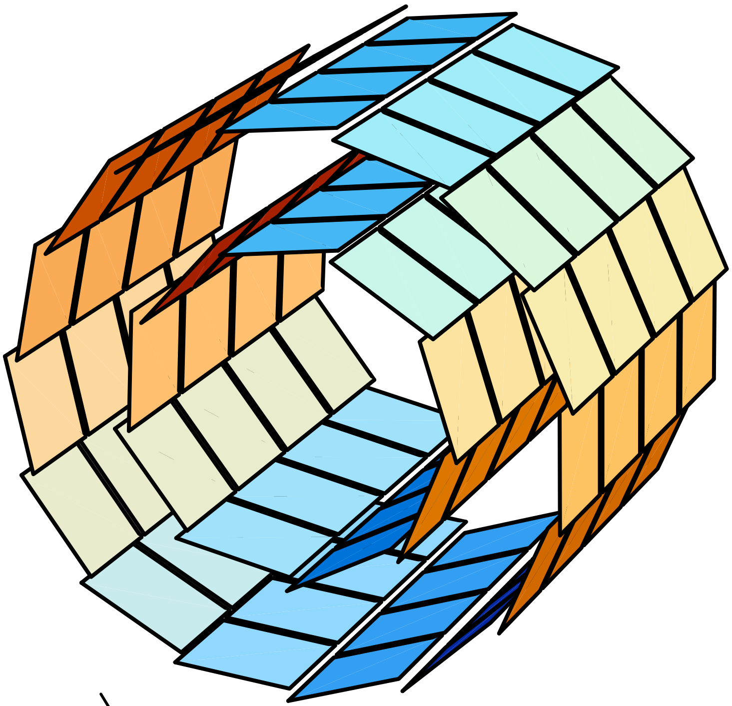
Pascal Gay

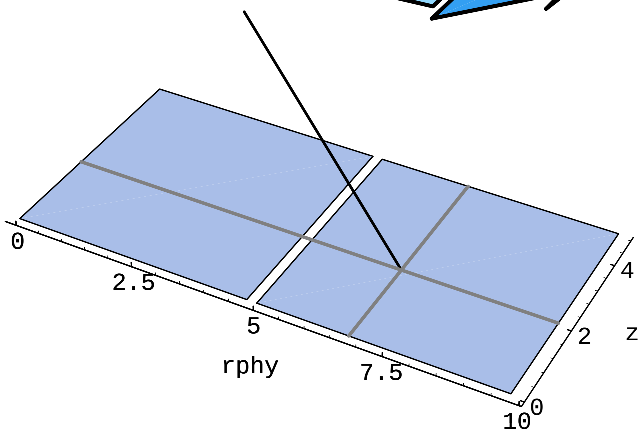
Laboratoire do Physique Corpusculaire,
Université Blaise Pascal, IN2P3–CNRS,
Clermont-Ferrand, 63177 Aubière, France

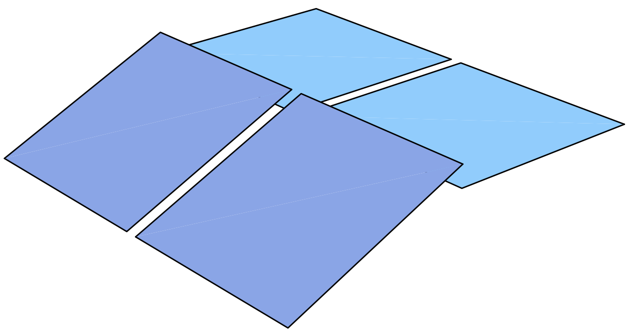
The ALEPH Detector



The ALEPH Vertex Detector







Paul E. Rensing 19/9/95

Sequential Method

- Order tracks by extrapolation footprint.
- For each track, starting with the smallest:
  - Look at all hits in 5σ road.
  - Choose set of hits with smallest increase in track fit χ^{2}.
  - Penalty of 15 for dropping a hit.
- Remove hits from consideration and go on to next track.

- Double assignment is allowed if pulse height is big enough.

- This actually works fairly well.
  - Rate of assigning hits correctly is ~ 90%
  - Error rate is ~ 5%

Global Method

- We wish to create a more global algorithm to handle hard, dense problems better.
- Need a tool to solve large combinatoric problems:
  - "integer programming"
- But, no need to solve entire event at once:
  - Most tracks are never in competition with others for hits.
  - Spatial nature of event naturally divides problem into many small "components" which can be solved separately.

- General Algorithm
  - Find all reasonable hit patterns for each track.
  - Divide event into sets of tracks which are in competition for hits.
  - Solve each set separately.

Minimization Model

- For each track, i, the "cost coefficient", O_i , is

  $$O_i = \chi_i^2 + P_\phi \, N_{\phi,i} + P_{1D} \, N_{1D,i}$$

- χ_i^2 is the increase of track fit χ^2 due to VDET hits

  $\chi_i^2 = \chi_i^2(\text{VDET+ITC+TPC}) - \chi_i^2(\text{ITC+TPC})$

- Add penalties for missing hits

  $P_\phi = 11$ for each missing hit

  $P_{1D} = 4$ for each hit with no matching hit in other view

  $\Rightarrow$ hard to drop one hit from matched pair, easier to drop second hit.

- Total "objective function", O, to be minimized is

  $$O = \sum_{\text{tracks}} O_i$$

- Double hits
  - a hit can be assigned twice only iff
    1) pulse height is above a threshold,
    2) pulse height is larger by some amount than corresponding hit in other view.

Track Selection and Pattern Generation

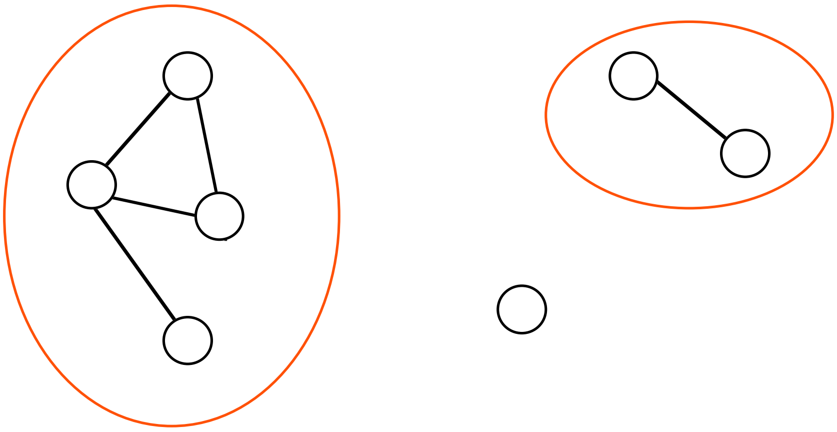
- Tracks which are thought to be from V^0s, kinks, or nuclear interactions outside the VDET are not considered in the first round.

- For each track being considered in this round, find all hits within a 5σ road.
- Generate all patterns of hits with reasonable O_i, using a simple matrix calculation.
- Fit all patterns and cut on O_i.

- Overlap regions:
  - Tracks in overlap regions can get 2 hits in each view within one layer.
  - Overlap patterns are considered as one logical hit, but all permutations are generated.

Component Generation

- Want to find all sets of tracks which are in competition for hits.
- Tracks are connected if they share any hit in the list of possible patterns.



- Component pruning
  - Algorithm was not successful unless tracks with large extrapolation footprints were removed from components.
  - Remove track if extrapolation uncertainty is 3 times larger than best track in component.
  - Removed tracks are handled in subsequent round of pattern recognition.
  - Need reflects lack of understanding of minimization function.

Solution of Combinatorics

- For small problems, solve by exhaustive search.
- For big problems, use "branch and bound" algorithm.
- Keep a small list of ambiguous solutions.
  - Useful for study of systematic errors.
- Use one solution for final track fit.
  - Pick solution with most matched pairs of VDET hits (small improvement).

Branch and Bound

- We want to assign set of tracks I to sets of hits J, K, L, M (2 layers, 2 views).
  - sets of hits are disjoint
  - assignment must be done in one step to exploit correlations between views and layers.

$\Rightarrow$ "5D assignment problem"

- Problem is expressed as a function to be minimized:

  $O = \Sigma_{ijklm} C_{ijklm} x_{ijklm}$ where $x_{ijklm} = 0$ or 1

  and a set of constraints:

  $$\Sigma_{jklm} x_{ijklm} = 1 \text{ for all tracks } i$$

  $$\Sigma_{iklm} x_{ijklm} = 1 \text{ for all known single hits } j$$

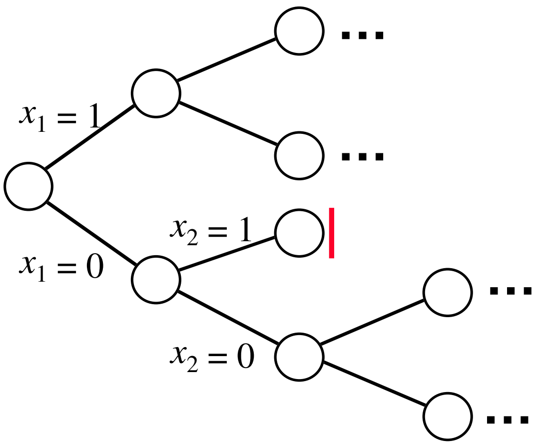
  $$1 \leq \Sigma_{iklm} x_{ijklm} \leq 2 \text{ for all possibly double hits } j$$

  and similarly for hits k, l, m.

- Track 0 and hits 0 are used to absorb noise hits or allow missing hits.
- Overlap hits add some additional bookkeeping in the constraints, but the problem is the same.

Branch and Bound

- The integer minimization problem is found by building a tree.



- Minimum and maximum limits on O are maintained throughout.
- At each node, solve the problem with x_{ijklm} continuous [0,1] using a linear programming (LP) solver.
- Based on solution, either "prune" the tree or continue building it.

Branch and Bound

- If LP does not succeed, there is no solution in this branch.

  $\Rightarrow$ prune.
- If all x_{ijklm} of solution are 0,1, this is valid solution.
  - O^{LP} of solution provides upper limit on final value of O.

    $\Rightarrow$ update $O^{max} = \min(O^{LP}, O^{max})$.
  - prune this branch.
- If some x_{ijklm} is non-integer,
  - O^{LP} provides lower limit on best O in this branch.

    $\Rightarrow$ prune if $O^{LP} > O^{max}$
  - Else, add branches with $x_{ijklm} = 0, 1$ for some choice of i,j,k,l,m.

- Process is finished when all branches are pruned.
- Solution of integer problem is best of the valid solutions.

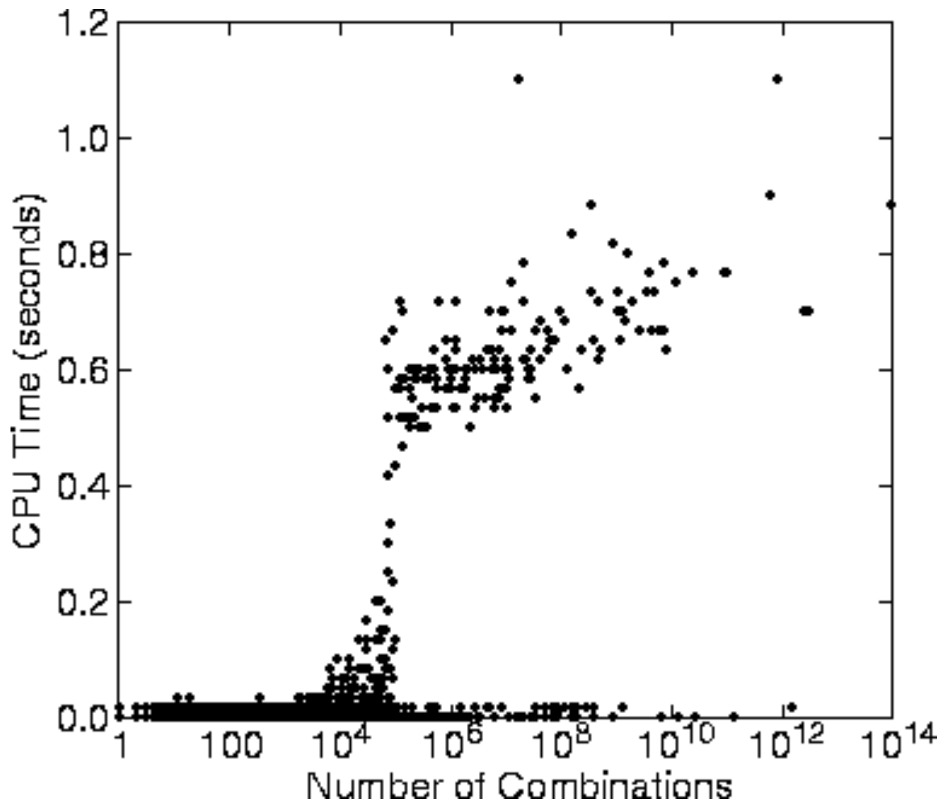
Branch and Bound

- There are many tricks used to reduce the amount of computation done in solving the LPs.
- There are different schemes for choosing the next unpruned node on which to add new branches.
- Same technique can solve problems with general integer variables or a mixture of integer and continuous variables.

- CPU time can become exponential.
- Since you have a lower and upper limit on O, process can be terminated after some accuracy or CPU limit is reached (not needed for this application).

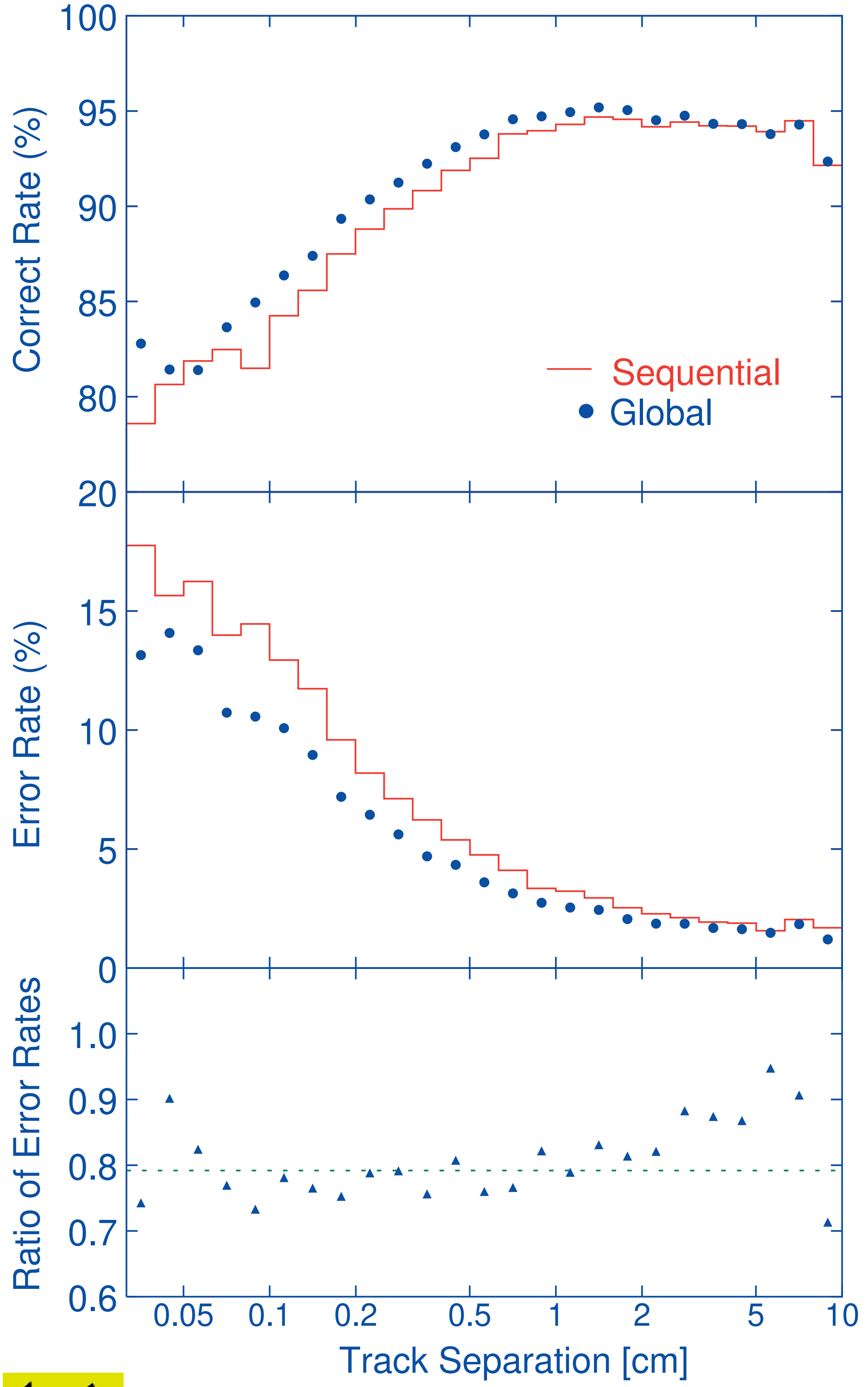
- References:
  - FAQ http://www.skypoint.com/subscribers/ashbury/linear-programming-faq.html
  - George L. Nemhauser and Laurence A. Wolsey, *Integer and Combinatorial Optimization*, (John Wiley & Sons, New York, 1988)
- Good commercial and freeware solvers exist.
  - lp_solve from ftp://ftp.es.ele.tue.nl/pub/lp_solve
  - commercial is much better for integer programming

CPU Time



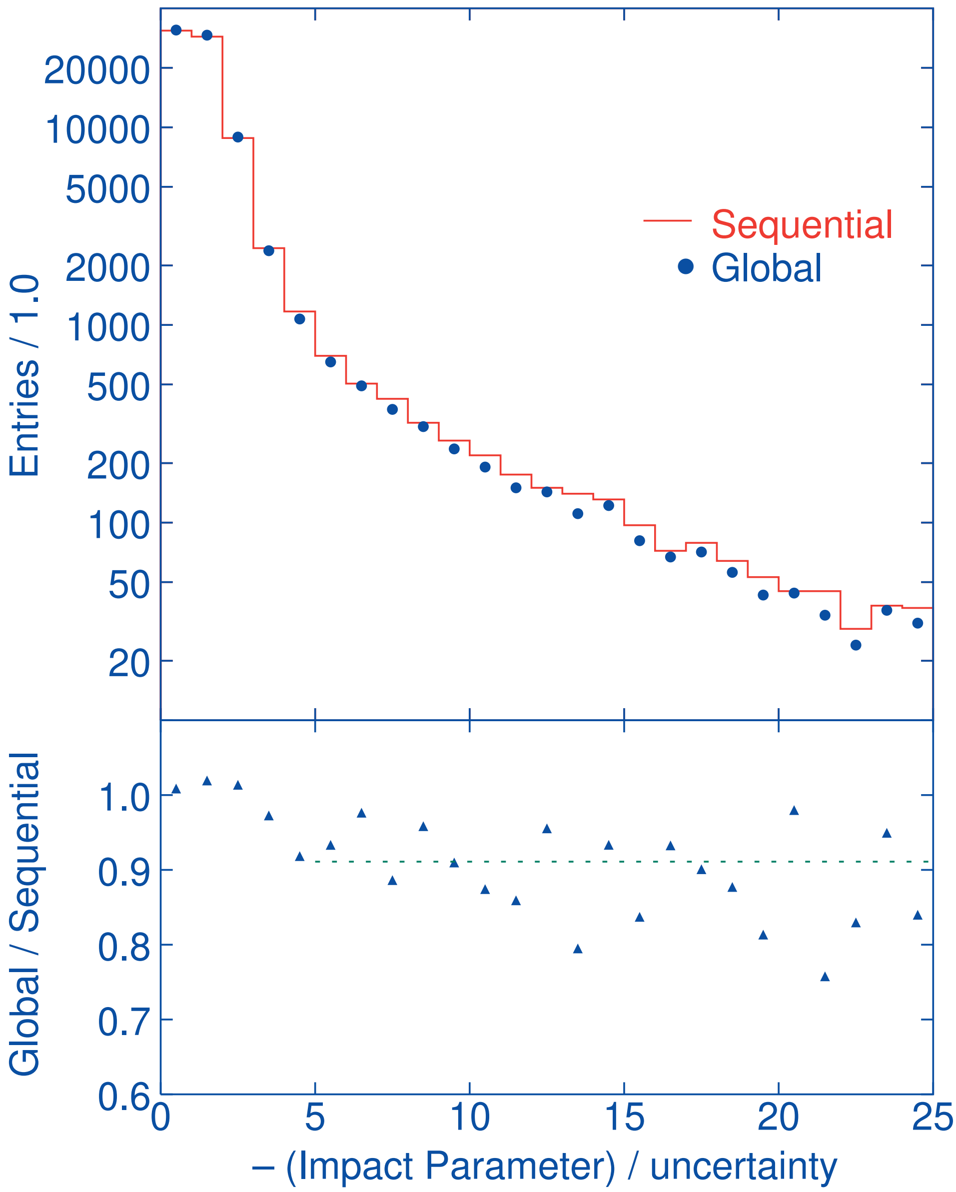
- Benchmark on DEC Alpha 3000/300 (~ 76 SPECInt 92)
- No component has taken more than 1 CPU second to solve.
- On average, $q\bar{q}$ events takes ~6 seconds for complete procedure.
- ~80% of time is spent fitting the tracks with different hit patterns.

Assignment Rate (MC)



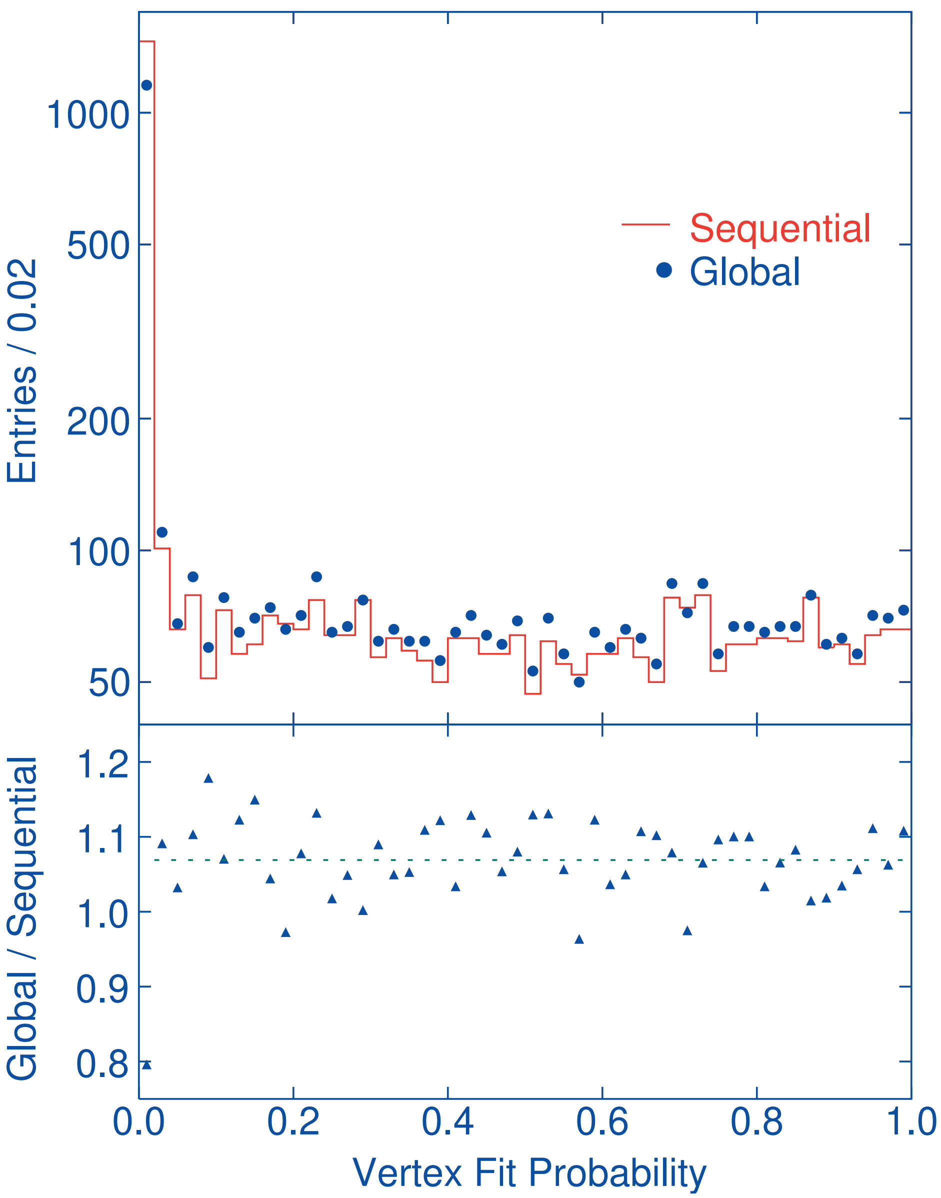
Paul E. Rensing 19/9/95

Impact Parameter (MC)



Paul E. Rensing 19/9/95

Vertex Quality (τ 3-prong)



Lessons Learned

- It is worth solving the entire problem at once, if it can be done.
  - Initial algorithm tried to solve the two views separately and combine.
  - Reduced one combinatorial problem, but combining the two views was a large problem by itself.
  - Ignoring correlation between the views caused problems.
- Better understanding of the measure of assignment is needed.
  - χ^2 is not really a measure of the probability of assignment.
  - Possibly need to use a joint probability.

Conclusions

- We have implemented a global algorithm to perform the pattern recognition for ALEPH's VDET.
- The algorithm uses the technique of "integer programming" to solve the large combinatorial problems encountered.
- CPU time is not a problem.
- The rate of assigning hits to the wrong track has been reduced by 10-20%, while correct assignments have increased.