GLOBAL VDET PATTERN RECOGNITION FOR ALEPH

ANDREW BAZARKO[a], DAVID BROWN[b], PASCAL GAY[c],
JEAN-FRANÇOIS PUSZTASZERI[a], PAUL E. RENSING[a]

[a]*European Laboratory for Particle Physics (CERN), 1211 Geneva 23, Switzerland*

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

[c]*Laboratoire de Physique Corpusculaire, Université Blaise Pascal, IN^{2}P^3-CNRS, Clermont-Ferrand, 63177 Aubière, France*

With the current reliance on high-precision vertex detectors to provide very accurate information about the primary and secondary vertices in an event, the accuracy and efficiency of the assignment of vertex detector hits to tracks has become crucial. This paper discusses new software written for ALEPH which attempts to make this assignment in a global manner using integer programming techniques.

1 Introduction

Much of the physics being done at LEP relies heavily on finding primary and secondary vertices in an event. This procedure requires accurately knowing the positions of tracks near the interaction point. Large errors can be made if tracks are assigned the wrong vertex detector hits. Currently, ALEPH performs the pattern recognition via an algorithm which deals with only one track at a time. An effort has been underway for the past year to write a new algorithm which attempts to determine the optimal assignment over the whole event.

The ALEPH Detector is described in detail in [1]; only a brief description of the relevant tracking elements will be given here. The tracking system consists of a large time projection chamber (TPC), a cylindrical drift chamber (ITC), and a high-resolution silicon strip vertex detector (VDET). The VDET features two layers of 300 μm thick silicon wafers at average radii of 6.3 and 10.8 cm. The spatial resolution for r–ϕ coordinates is 12 μm and varies between 12 and 22 μm for z coordinates.

2 Sequential Algorithm

The sequential algorithm currently used by ALEPH to perform the pattern recognition for the VDET starts with the tracks found in the TPC and ITC. The tracks are extrapolated to the VDET and are ordered by their extrapolation footprints in the r–ϕ direction. Then, working from smallest to biggest footprint, the best combination for each track is found. The measure used is the χ^2 of the track fit, with a penalty of 15 for each missing hit. Hits can be used more than once provided their pulse height is large enough, but no effort is made to optimize the assignment over more than one track.

This algorithm in fact works quite well because of the favourable conditions of ALEPH. There are less than 0.034 radiation lengths of material and about 5 cm in radius between the innermost wire of the ITC and the outer layer of the VDET. Ordering the tracks by their extrapolation footprint allows the tracks which typically experience the least multiple scattering, and therefore are the best measured, to get hits first.

3 Global Model

The model used is again based on the χ^2 of the track fit. The "cost coefficient" for track i is given by

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

where χ_i^2 is the increase in track fit χ^2 due to the VDET hits, N_ϕ is the number of missing hits, $N_{1\mathrm{D}}$ is the number of hits which do not have a matching hit in the other view, and P_ϕ and $P_{1\mathrm{D}}$ are the associated penalties. Currently, best values of the penalties seem to be

$$P_\phi = 11, \qquad\qquad P_{1\mathrm{D}} = 4. \tag{2}$$

This maintains the penalty of the sequential algorithm (*i.e.* 15) to drop a hit which is matched with a hit in the other view, but makes it somewhat easier to drop a hit which is not matched. For a given assignment, the "objective function" to be minimized is simply the sum of O_i over the tracks.

The minimization is done with a number of constraints on the use of the hits. Most hits can only be assigned to one track. However, if the hit's pulse height, normalized by the angle of the track crossing the VDET wafer, is larger than a given threshold, it may be assigned twice. In addition, when a double assignment is attempted, the pulse height of the double hit must be larger than either matching hit in the other view.

4 Global Algorithm

The input to the VDET pattern recognition is the set of tracks previously found in the TPC plus ITC. Prior to the start of the pattern recognition, searches are made for V^0s, kinks (decays in flight), and nuclear interactions. Tracks which have a high probability of having originated outside the VDET are flagged and only considered for hit assignment after the pattern recognition for the other tracks has been completed.

The first step of the pattern recognition is to find all likely patterns of hits for each track. Each track is extrapolated to the two VDET layers and lists are made of all hits which fall within a 5σ road. Various steps are used to find all physical patterns with a reasonable value for the objective function. The final step is to fit each remaining pattern using ALEPH's track fitter, based on a Kalman filter. Patterns are saved if their objective function is within 30 of the best pattern found for the track; the pattern with no hits assigned to the track is always saved as a backup. Typical tracks have only a few reasonable patterns, but some can have over 100 patterns.

The next step is to divide the event into sets of tracks ("components") which are in conflict for hits. Any two tracks are in conflict if they share any hit in their list of possible combinations. Therefore, in terms of determining the assignment of hits to tracks, each component is logically independent from all others and can be solved by itself. Unfortunately, it was found that directly solving components as just defined did not provide good solutions. Tracks with very large extrapolation footprints have a large number of possible patterns and very small χ^2 for many combinations, allowing them to "steal" hits from tracks with small footprints. This problem reflects a lack of understanding about the correct function to minimize. The current solution is to remove tracks from a component if their footprint is more than 3 times larger than the smallest in the component. The removed tracks are left to get hits in the next round of the pattern recognition. Work is continuing to find a more sophisticated component generating algorithm.

Roughly 80–90% of the components in $q\overline{q}$ events consist of only one or two tracks and can be easily solved by exhaustive search. Some components, however, can have up to 10 tracks and billions of possible combinations. These are solved by a "branch and bound" algorithm, described below. The algorithms find the "best" solution plus a small number of solutions which are possibly ambiguous with the best.

5 Branch and Bound Algorithm

The key to this pattern recognition method is the branch and bound algorithm, taken from Integer Programming[2], which is used to solve the large combinatorial problems. The technique is described in detail in reference [3]. The problem is to determine the optimal assignment between a set of tracks, I, and four disjoint sets of hits, J, K, L, and M, and is therefore called a "5D assignment problem". The problem is expressed as a function to be minimized and a set of constraints

$$\begin{aligned}
O(x_{ijklm}) &= \sum_{ijklm} C_{ijklm} x_{ijklm} && \text{with } x_{ijklm} = 0, 1 \\
\sum_{jklm} x_{jklm} &= 1 && \text{for all tracks } i \\
\sum_{iklm} x_{iklm} &= \begin{cases} 1 & \text{(single hits)} \\ 1, 2 & \text{(otherwise)} \end{cases} && \text{for all hits } j
\end{aligned} \tag{3}$$

with similar constraints on the other sets of hits. The values C_{ijklm} are given by equation 1. The variables x_{ijklm} are called the "assignment variables"; a value of $x_{ijklm} = 1$ means that track i is assigned hits j, k, l, and m, while a value of 0 indicates no assignment. A track 0 and hits 0 (with no constraints applied) are added to absorb noise hits and allow missing hits in the assignment.

The branch and bound algorithm proceeds by building a binary tree. As the tree is built, upper and lower limits on the optimal value of the objective function are maintained. At each node of the tree, the current problem is solved with the variables x_{ijklm} continuous on the interval $[0, 1]$ using a linear program (LP) solver.[4] Based on the solution, the node is either "pruned" (*i.e.* no more leaves added) or more leaves are added to the tree. If the solution of the LP is entirely integer, then this is a valid solution of the assignment problem, and the objective value obtained gives a new upper limit on the optimal value. If the solution of the LP contains some non-integer values, the solution of the LP gives a lower limit on the best solution to be found in this branch. The algorithm proceeds by generating two branches off some node, one with $x_{ijklm} = 1$, the other $x_{ijklm} = 0$, for some choice of i, j, k, l, and m. The branch and bound is complete when all nodes of the tree have been pruned or evaluated, and the solution is the best of the valid solutions found.

Good commercial and free software exists for solving linear programs,[5] but the commercial packages are significantly better for integer programming. These packages use various techniques to reduce the time needed to solve the many LPs, but reducing the number of LPs solved is much more important. The branch and bound algorithm is not guaranteed to complete within any reasonable bound on time. However, since one has an upper and lower limit on the objective function, the algorithm can be terminated once some limit on accuracy or compute time has been reach. This is usually not a loss, as most of the CPU time is spent proving the optimality of a solution found early in the algorithm.

6 Results

Figure 1 shows the CPU time taken to solve the combinatorial problems in 1000 $q\overline{q}$ events on a DEC Alpha 3000/300 ($\sim$ 76 SPECint 92). The transition

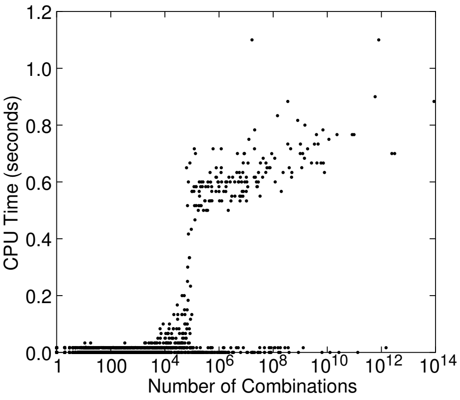

Figure 1: The CPU time used to solve a component versus the number of combinations in a component.

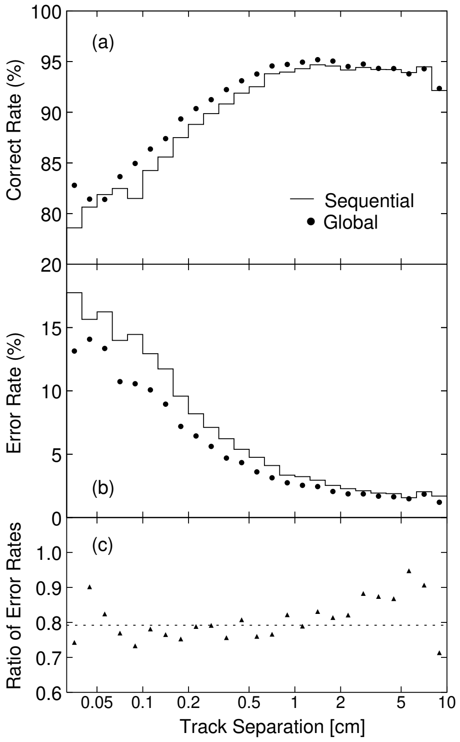

Figure 2: The (a) correct and (b) incorrect assignment rates for the sequential and global algorithms as a function of track separation at the VDET ($q\overline{q}$ Monte Carlo). Figure (c) shows the ratio of the error rate from the global algorithm to that from the sequential one; the dashed line indicates the average value of 0.79.

between the exhaustive search and branch and bound algorithms is clearly visible at 10^5 combinations. No component seen so far has taken more than about 1 CPU second to solve. The entire VDET pattern recognition takes about 6 CPU seconds on average for $q\bar{q}$ events, about 80% of which is spent fitting hits to determine the χ^2 for each pattern.

Figure 2 shows the rates of (a) correct and (b) incorrect assignment of hits to tracks as a function of the separation between a track and its nearest neighbour at either layer of the VDET. Figure 2(c) shows the improvement of the error rate for the global algorithm with respect to the sequential one. The figures clearly show that, while the efficiency has increased by a small amount, the error rate has been reduced by about 20%. Tests have been performed on ALEPH data using vertex fits and impact parameter resolution. These tests indicate an improvement of around 10%.

7 Conclusion

A global algorithm to perform the pattern recognition for ALEPH's VDET has been successfully implemented. A branch and bound algorithm, taken from integer programming, is used to solve the large combinatorial problems encountered, allowing all problems to be solved to optimality. Monte Carlo and data show evidence for a 10–20% decrease in the error rate, but work is continuing on refining the minimization function.

Acknowledgments

We wish to thank our colleagues in the CERN accelerator divisions for the successful operation of LEP. We are indebted to the engineers and technicians in all our institutions for their contribution to the excellent performance of ALEPH. Those of us from non-member countries thank CERN for its hospitality.

References

1. ALEPH collaboration, D. Decamp *et al.*, *Nucl. Inst. Meth.* **A294**, 121 (1990).
2. George L. Nemhauser and Laurence A. Wolsey, *Integer and Combinatorial Optimization*, (John Wiley & Sons, New York, 1988).
3. Jean-François Pusztaszeri, Paul E. Rensing and Thomas M. Liebling, *Tracking Elementary Particles near their Primary Vertex: A Combinatorial Approach*, submitted to *Operations Research.*
4. Michel Berkelaar, *lp_solve*, ftp://ftp.es.ele.tue.nl/pub/lp_solve
5. John W. Gregory, *Linear Programming - Frequently Asked Questions List*, http://www.skypoint.com/subscribers/ashbury/linear-programming-faq.html