18th International Conference on Information Fusion
Washington, DC - July 6-9, 2015

A Maximum Weight Constrained Path Cover Algorithm for Graph-Based Multitarget Tracking

Lingji Chen and Ravi Ravichandran
BAE Systems
Burlington, MA, USA

***Abstract*—In graph-based target tracking, vertices represent measurements and/or tracklets and edges represent allowable associations. Therefore a solution with a set of tracks is simply a vertex-disjoint path cover of the graph. Under certain (path independence) conditions, the tracking problem can be transformed into one of finding the maximum weight vertex-disjoint path cover of a directed acyclic graph, which can be efficiently solved using maximum weight bipartite matching or minimum cost network flow algorithms. However, attribute information often leads to path dependence. In this paper we consider an associated graph theoretic problem of finding the maximum weight vertex-disjoint path cover under the constraint that a given pair of vertices have to be on the same path. We show that the greedy algorithm is not optimal, and conjecture that the problem is nondeterministic polynomial-time hard (NP-hard). This motivates us to seek efficient algorithms for special classes of graphs that can be used as subroutines in a general algorithm. We present an efficient optimal algorithm for trellis graphs, and indicate how it can be used as a building block in a search algorithm for the general case.**

I. Introduction

Multiple Hypothesis Tracking (MHT) is an enabling technology that has been used in myriad applications with major success. In a typical (track oriented) MHT implementation, a tree/forest data structure is maintained and, as measurements come in, new leaf nodes are created and incorporated to represent possible (track) hypotheses. The best global solution in the form of a set of compatible (track) hypotheses with the best association likelihood is then obtained by solving a linear integer programming problem. The width of the tree can grow exponentially with the number of measurements in such an approach, and pruning becomes a crucial part of MHT. In recent years, Graph-Based Tracking (GBT) has drawn a lot of attention in the research community while it is being used in more and more applications; see for example [1] and the references therein. In a graph based approach, measurements are incorporated into a graph data structure, and local computations such as gating and likelihood are also carried out and recorded in the graph. Hypotheses are not explicitly represented, and the best global solution is obtained using suitable algorithms depending on the nature of the problem. Thus, data representation and inference are decoupled in GBT. The number of vertices grows linearly with the number of measurements, making GBT an attractive choice for handling large data sets.

More importantly, there are many applications where certain Markov condition holds, such that the likelihood of an incremental track association does not depend on the path the track took to reach "here." Then a log likelihood ratio can be assigned as a weight on an edge in the graph, and finding the best global solution in the form of a set of tracks becomes finding the maximum weight vertex-disjoint path cover of the weighted, directed acyclic graph, which can be solved very efficiently. In the following "vertex-disjoint" is always assumed for a path cover.

For applications where the Markov condition (or the path independence condition) does not hold, for example when long term attribute information is present, GBT offers savings in data representation but not necessarily in inference as compared to MHT. This paper considers a special type of hard constraints posed by attributes: A pair[1] of measurements are required to be in the same track (because for example they have the same identification code). The associated graph theoretic problem is that of finding the maximum weight vertex-disjoint path cover under the constraint that a given pair of vertices have to be on the same path. Solving this problem will be our focus in this paper, which is organized as follows.

In Section II we set up the GBT problem, describe the case of path independence, and show how it can be solved efficiently. In Section III we introduce the same-track constraint, present a greedy algorithm and show that it is not optimal. Inspired by the NP-completeness results in [2], [3], and to attract more attention and research, we make a conjecture that the problem is NP-hard. This motivates us to seek efficient algorithms for special classes of graphs that can be used as subroutines in a general algorithm, so in Section IV we present an efficient algorithm for trellis graphs. In Section V we indicate how this algorithm can be used as a building block in a search algorithm for the general case. In Section VI we draw conclusions and discuss future work.

This material is based upon work supported by the Washington Headquarters Services under Contract No. HQ0034-12-C-0050. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Washington Headquarters Services.

Approved for public release; unlimited distribution.

[1] In general, there can be a set of such pairs, and the problem becomes even more difficult.

II. Path independence and maximum weight path cover

The GBT approach employs a track graph $G(V, E)$ where a vertex represents a measurement or a tracklet[2] and an edge represents a possible association (after spatial and temporal gating). A vertex-disjoint path cover [4], [5] (or path partition [6]) is a set of paths, including path of length zero (i.e., single vertex) such that each vertex belongs to one and only one path. The tracking problem then is to determine the best path cover of the track graph according to certain criterion. Figure 1 illustrates a valid path cover with three paths: ade, bc and f. For the problems we consider in this paper, we can typically use the (start or stop) time stamps of the measurements/tracklets to impose a direction of association from an earlier time to a later time, and therefore the track graphs we consider in the following will be directed acyclic graphs, unless otherwise stated.

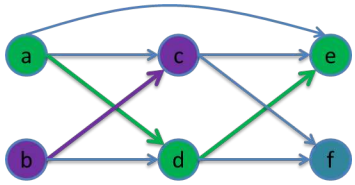

Fig. 1. A path cover with three paths: ade, bc and f.

If we have a weighting function $w(\cdot)$ on a path p, for example as a function of its likelihood, then we can define the best path cover as

$$\mathcal{P}^* = \underset{\mathcal{P}}{\operatorname{argmax}} \sum_{p \in \mathcal{P}} w(p). \tag{1}$$

This general form does not permit efficient algorithms: We may not be able to reuse the result of $w(\{de\})$ in calculating both $w(\{ade\})$ and $w(\{bde\})$; the weighting function may be path dependent.

However, there are many applications where the likelihood of a track satisfies a Markov, or path independence, condition; see [7], [8], [1], [9] for details, and [10], [11], [12] for related work. In such a case, we can assign a log likelihood ratio [13] as a weight on an edge, and define the best path cover as

$$\mathcal{P}^* = \underset{\mathcal{P}}{\operatorname{argmax}} \sum_{p \in \mathcal{P}} \sum_{e \in p} w(e). \tag{2}$$

A weighted track graph is shown in Figure 2. Then Equation (2) defines a maximum weight path cover problem on the graph, which can be solved by maximum weight bipartite matching [14] or minimum cost network flow [5]. In Figure 3 we illustrate the maximum weight bipartite matching approach, where we "split" each vertex v into two (connected) vertices v' and v'', such that each in-arc of v goes into v' and each out-arc of v goes out of v''. The resultant companion graph is a bipartite graph and its maximum weight bipartite matching provides edges that can be assembled into paths to give the solution to (2).

[2]A tracklet is a set of measurements determined by the sensor to have originated from the same target.

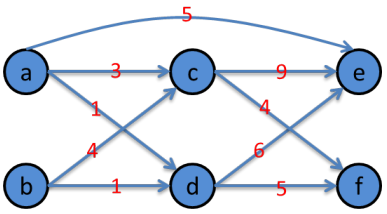

Fig. 2. With path independence, the tracking problem becomes one of finding the maximum weight path cover on a weighted track graph.

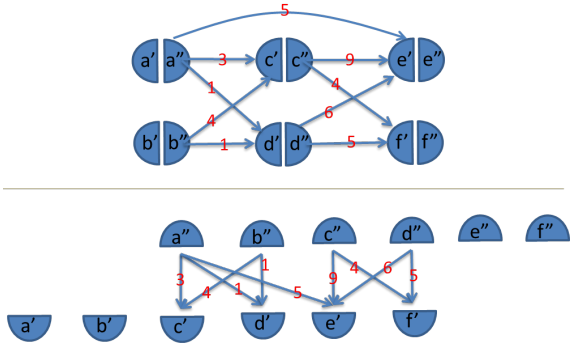

Fig. 3. Maximum weight path cover of a directed acyclic graph can be solved using maximum weight bipartite matching.

The complexity of solving for the maximum weight bipartite matching by the Hungarian algorithm [15] is $O(|V|^3)$ where $|\cdot|$ denotes cardinality.

III. The same-track problem and a conjecture on its complexity

Attribute information often leads to track dependence. For example, if we have color information of the vehicles being tracked, then whether a red car should be "stitched" with a car in an earlier frame, whose color is not observed, depends not only on the kinematics, but also on whether this no-color car has already been stitched with an orange car even earlier.

In such cases, a path weight cannot be written as a sum of edge weights, and we have to solve the problem in (1), not (2). However, we consider a simplified problem where attributes provide hard constraints and nothing else. More specifically, we do not have propagation and likelihood models for the attribute, and the only information we extract and use is whether a path is valid or not. In such a case, if we first construct a weighted track graph using only path independent information such as kinematics, then a valid path for the constrained problem has the same weight as in the

Approved for public release; unlimited distribution.

unconstrained case, while an invalid path has a weight of minus infinity.

Thus, we can attempt to solve (2), i.e., obtaining the maximum weight path cover, but subject to some hard constraints on the validity of paths. As a starting point, in this paper we consider only a single *same-track* constraint that a given pair of vertices have to be on the same path.

For this problem, there is an apparent greedy algorithm.

The greedy algorithm (approximate solution):

1) Find the longest path, p_l, between the pair of vertices that form the same-track constraint.
2) Delete all vertices on p_l and their incident edges.
3) Solve the maximum weight bipartite matching problem, now unconstrained, for the remaining graph. Let the set of paths be $\bar{\mathcal{P}}$.
4) Return $\mathcal{P} = \{p_l, \bar{\mathcal{P}}\}$ as an approximate solution.

It can be readily verified, by using the track graph in Figure 2, that the greedy algorithm does not always provide the optimal solution: If the same-track pair is a-f, then the greedy solution is (acf, bde) with a total weight of 14, while the optimal solution is (adf, bce) with a total weight of 19.

For a small graph, we can solve the problem by a brute force algorithm that enumerates all connecting paths between the same-track pair, and for each path follows the same steps as in the greedy algorithm. The number of connecting paths can grow exponentially as the graph becomes larger and the pair of vertices become farther away. Thus there is reason to suspect that the constrained maximum weight path cover problem could be NP-hard. In [2], [3] some other types of constrained path cover problems were discussed in the context of computer program testing, and many problems were proved to be NP-hard. In view of these, and with an intention to spur more research in this area, we venture the following:

Conjecture. *The same-track problem of finding the maximum weight vertex-disjoint path cover of a weighted directed acyclic graph such that a given pair of vertices are on the same path is NP-hard.*

A proof of this conjecture would be valuable. An efficient algorithm to disapprove this conjecture would be satisfying.

IV. An Efficient Optimal Algorithm for Trellis Graphs

The possibility that the same-track problem is NP-hard motivates us to seek efficient algorithms for special classes of graphs and use them as subroutines in a general search. We consider a graph $G(V, E)$ that is a *trellis* [16]: The vertex set V can be divided into disjoint subsets, called stages, of vertices $\{V_i\}$, $i = 1, 2, \ldots, N$, such that any edge $e = v_i v_j$ is between two adjacent stages, i.e., there is an index t such that $v_i \in V_t$ and $v_j \in V_{t+1}$. Figure 4 illustrates one such graph, or more precisely, its companion bipartite graph where we split each vertex v into v' and v'' as discussed in Section II.

Approved for public release; unlimited distribution.

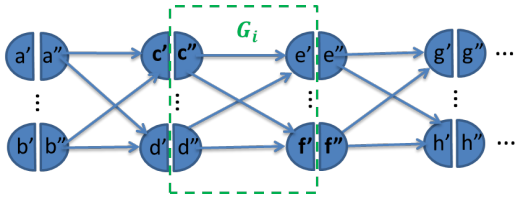

Fig. 4. In a trellis, a local matching does not interfere with other local matchings, and the union is always a valid global matching.

Consider the subgraph G_i formed by edges between two adjacent stages and their associated vertices, and a matching M_i of G_i. It is apparent that the (companion) graph G is the union of such subgraphs, and the union of the "local" matchings M_i always forms a valid matching M for the whole graph G. It is this property that will be exploited in constructing an efficient algorithm for the same-track problem.

Figure 5, together with Figure 4, illustrate the idea. Let a and h be the same-track pair. Suppose that the edge cf is on the optimal global solution. It follows that, in the subgraph G_i, the local matching of the "remaining" graph, after the vertices c and f and their incident edges are deleted, has to be optimal. We can thus obtain a weight for the edge cf in an auxiliary graph shown in Figure 5 and find the connecting path in the global optimal solution by finding the longest path in the auxiliary graph.

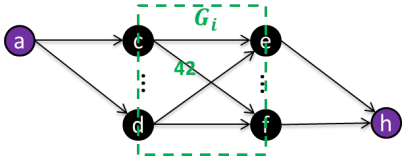

Fig. 5. Constructing an auxiliary weighted graph and finding its longest path, which will be in the best path cover of the same-track problem.

The trellis algorithm (exact solution):

1) Let v_a and v_b be the same-track pair that are constrained to be on the same path, and v_a be earlier than v_b (in the topological sort). Perform a forward sweep to mark "gray" all vertices reachable from v_a. Perform a backward sweep to mark "black" all gray vertices that can reach v_b. Construct an auxiliary graph with only black vertices (including v_a and v_b).
2) For an edge e in the auxiliary graph with weight w_e, consider the corresponding edge $v''_x v'_y$ in the companion graph and the subgraph G_i that contains it, as described before. Obtain the maximum weight bipartite matching for the remaining graph of G_i with v''_x and v'_y deleted together with their incident edges, and let the optimal weight be W_e. Assign the weight $w_e + W_e$ to edge e in the auxiliary graph.

3) Find the longest path p in the auxiliary graph. This path is in the best global solution.
4) The rest of the steps follow those in the greedy algorithm.

The worst case running time of the trellis algorithm is $O(n^5)$ where $n = |V|$, since there are at most $O(n^2)$ edges, each costing $O(n^3)$ to solve for the optimal weight in the subgraph. In practice, the main concern is for a trellis having $N \gg 1$ stages with $m = O(1)$ vertices in each stage. There are m^N possible connecting paths, but the trellis algorithm can be run with $m^5 N$ time.

Numerical examples: We conduct Monte Carlo runs to evaluate the trellis algorithm and compare with the greedy algorithm and the brute force algorithm. More specifically, we choose $m = 6$ for the number of vertices in a stage, and varies the number of stages N from 5 to 15. For each (m, N) configuration, we conduct 100 Monte Carlo runs and record the worst running time. In each run, 50% of all possible edges are randomly chosen to be present with a weight uniformly distributed between -1 and 1.

Both the trellis algorithm and the greedy algorithm are run for all values of N. The brute force algorithm is run for only $N = 5, 6$ and 7. In these 3 settings, the solutions obtained by the trellis algorithm and the greedy algorithm are judged against the optimal solution obtained by the brute force algorithm, and the fraction of the optimal solutions out of all runs is recorded for the two algorithms. This fraction is always 1 for the trellis algorithm, as we reasoned before, but less than 0.5 for the greedy algorithm.

The worst running times and the fractions of optimal solutions are plotted in Figure 6, where results of all N values are combined for better visualization even though the brute force algorithm is run for only three values of N.

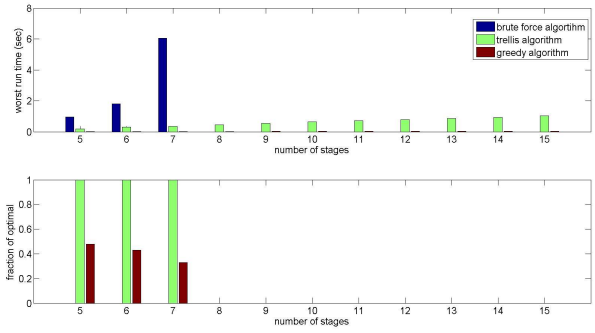

Fig. 6. Top: Worst running times; the brute force algorithm was run only for the first three cases. Bottom: Fraction of solutions that are optimal as determined by the brute force algorithm for the first three cases.

V. Search for the General Case

In this section we indicate how the trellis algorithm can be used in a search algorithm for the general case. If, as illustrated in Figure 7, an edge "jumps over" a stage, then we can construct an auxiliary graph by adding an auxiliary vertex in the stage that is jumped over, and the resultant graph will be a trellis[3]. Any solution for the original graph can be expressed as a solution for the auxiliary graph: If the edge ab is in the solution on the left, then replacing ab with azb is a valid solution on the right. If the edge ab is not in the solution on the left, then the solution is valid on the right which does not include either az or zb.

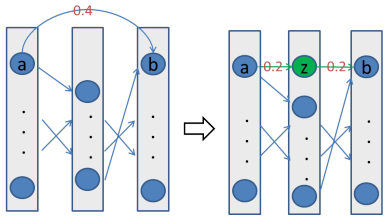

Fig. 7. Adding an auxiliary vertex to turn the graph into a trellis.

Therefore, if we solve for the optimal solution on the right, and it is a valid solution on the left (i.e., it contains either both az and zb, or neither az or zb), then it is the optimal solution on the left. If however, only one of az and zb is present in the solution, then we need to search further. Suppose az is present and zb is absent. Instead of splitting the weight of ab equally between az and zb, we can decrease the weight on az and increase the weight on zb while maintaining the same sum. If we solve again the problem for the trellis on the right, it *may* happen that az disappears from the optimal solution, and the optimal solution is valid for the graph on the left. The situation where az is absent but zb is present can be similarly handled.

The above procedure has the potential to terminate with an optimal solution, but this is not guaranteed. A procedure guaranteed to terminate with the optimal solution is as follows, which is practical only if the number of "violating" edges that jump over stages is small enough.

The search algorithm (exact solution):

1) Identify all "violating" edges that jump over stages, and enumerate their presence and absence in the optimal solution. (If there are L such edges, then the number of configurations is 2^L.)
2) For each enumerated configuration, construct an auxiliary trellis graph by adding auxiliary vertices in the "jumped over" stages, and assigning weights appropriately.
3) Solve the problem for each trellis, and record the total weight.
4) Declare the solution with the largest weight to be the optimal solution.

[3]This procedure can be applied recursively for jumps over more than one stage.

Approved for public release; unlimited distribution.

Other heuristics with a combination of weight re-assignment and enumeration (of a smaller set) can also be applied, and are not elaborated here.

VI. Conclusions

Graph Base Tracking offers the advantage of decoupling data representation from data inference, and is better suited for handling large data sets as compared to MHT. Under certain path independence condition, which usually holds for kinematics obtained with modern sensors, the inference problem can be transformed into one of finding the maximum weight path cover, which can be solved efficiently. However, with long term attribute information, path independence no longer holds, and research is being carried out to find efficient algorithms to solve such problems. This paper considers a special case of attribute-induced hard constraint, i.e., that a given pair of measurements/tracklets are known to be from the same target and therefore should be in the same track in the tracking solution. We have described the graph theoretic problem of constrained maximum weight path cover, made a conjecture about its computational complexity, and presented an efficient optimal algorithm for a subclass of graphs with a trellis structure. We also indicate how such an algorithm can be used in a search procedure for the general case.

The same-track problem can involve more than one pair of vertices, and finding efficient algorithms is part of the ongoing research. The constraint can also take the form that a given pair of vertices cannot be on the same path. An efficient trellis algorithm for one such constraint will be reported in a future paper, and more research is needed to deal with more than one constraints.

We have assumed in this paper that the only information we extract and use from the attribute information is the hard constraint that defines the validity of a path. In more general cases, the constraint can be "soft" in the sense that attribute can evolve and its measurement is probabilistic, e.g., red color can look brownish under a different lighting. Finding algorithms that are more efficient than a full scale MHT is another direction of future research.

Acknowledgment

This work was supported in part by Contract No. HQ0034-12-C-0050.

The authors would like to thank Dr. Ravi Prasanth for the trellis terminology. They would also like to thank Drs. Joao Cabrera and Chris Moss for their valuable comments and feedback.

References

[1] C.-Y. Chong, "Graph approaches for data association," in *15th International Conference on Information Fusion*. IEEE, Jul. 2012, pp. 1578–1585.

[2] S. C. Ntafos and S. L. Hakimi, "On Path Cover Problems in Digraphs and Applications to Program Testing," *IEEE Transactions on Software Engineering*, vol. SE-5, no. 5, pp. 520–529, 1979.

[3] S. Ntafos and T. Gonzalez, "On the computational complexity of path cover problems," *Journal of Computer and System Sciences*, vol. 29, no. 2, pp. 225–242, Oct. 1984.

[4] Wikipedia, "Path cover," http://en.wikipedia.org/wiki/Path_cover, 2015.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*, 2nd ed. The MIT Press, Sep. 2001.

[6] A. Bondy and U. S. R. Murty, *Graph Theory (Graduate Texts in Mathematics)*, 1st ed. Springer, Sep. 2011.

[7] C.-Y. Chong, G. Castanon, N. Cooprider, S. Mori, R. Ravichandran, and R. Macior, "Efficient multiple hypothesis tracking by track segment graph," in *12th International Conference on Information Fusion*. IEEE, Jul. 2009, pp. 2177–2184.

[8] G. Castanon and L. Finn, "Multi-target tracklet stitching through network flows," in *2011 IEEE Aerospace Conference*. IEEE, Mar. 2011, pp. 1–7.

[9] S. Mori and C.-Y. Chong, "Performance analysis of graph-based track stitching," in *16th International Conference on Information Fusion*. IEEE, Jul. 2013, pp. 196–203.

[10] N. Vyahhi, S. Bakiras, P. Kalnis, and G. Ghinita, "Tracking Moving Objects in Anonymized Trajectories," in *Database and Expert Systems Applications*, ser. Lecture Notes in Computer Science, S. Bhowmick, J. Küng, and R. Wagner, Eds. Springer Berlin Heidelberg, 2008, vol. 5181, pp. 158–171.

[11] S. Zhang and Y. Bar-Shalom, "Track Segment Association for GMTI Tracks of Evasive Move-Stop-Move Maneuvering Targets," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 3, pp. 1899–1914, Jul. 2011.

[12] L. J. van der Merwe and J. P. de Villiers, "Track-stitching using graphical models and message passing," in *16th International Conference on Information Fusion*. IEEE, Jul. 2013, pp. 758–765.

[13] Y. Bar-Shalom, S. S. Blackman, and R. J. Fitzgerald, "Dimensionless score function for multiple hypothesis tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 1, pp. 392–400, Jan. 2007.

[14] F. T. Boesch and J. F. Gimpel, "Covering Points of a Digraph with Point-Disjoint Paths and Its Application to Code Optimization," *J. ACM*, vol. 24, no. 2, pp. 192–198, Apr. 1977.

[15] Wikipedia, "Hungarian algorithm," http://en.wikipedia.org/wiki/Hungarian_algorithm, 2015.

[16] D. A. Castanon, "Efficient algorithms for finding the K best paths through a trellis," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 26, no. 2, pp. 405–410, Mar. 1990.

Approved for public release; unlimited distribution.