Multiple Object Tracking

April 6, 2022

1 Introduction

Multiple object tracking can be cast as minimum-cost flow problem. The idea behind this approach is that, given a set of detections over multiple adjacent frames, a flow graph is constructed whose solution (a set of trajectories) is the globally optimal one. As for the more simple Kalman and particle filter tracking approaches, there are ways to incorporate a motion model and to use robust appearance features. This approach has two disadvantage that can however be greatly reduced. The first is that it requires solving a flow graph which can be computationally expensive, however, recent solvers based on shortest paths can now solve complex graphs in milliseconds. The second disadvantage is that obtaining the optimal solution implies using the detections of multiple frames which creates a delay (1-4 seconds in our case). The advantage on the other hand is that this method simultaneously recover the number of objects and their trajectories without the need of heuristics.

2 Background

2.1 Flow problem

Flow problems are a class of combinatorial optimization methods in which the input is a flow graph whose edges have a capacities and they receive a flow. The goal is to find the maximum or minimum flow that respect the capacity constraints on each edge and that have equal incoming and outgoing flow at all vertices except for certain designated terminals. Typical examples of network flow problem are optimization of work schedule, optimization of production and delivery of goods and variants allowing for multiple object tracking.

Let $G = (V, E)$ be a directed graph with node set V and edge set E. We define our example graph to have two designated terminals $s, t \in V$ named source and target nodes as shown in Figure 1. The capacity of an edge c_{uv} is fixed and it is equivalent to the maximum amount of flow that can pass through that edge. The flow in an edge f_{uv} instead is the amount that we are looking for and obey to the following two constraints:

1. Capacity constraint: $f_{uv} \leq c_{uv} \; \forall \; (u, v) \in E$
2. Flow conservation: $\sum_{u:(u,v)\in E} f_{uv} = \sum_{u:(v,u)\in E} f_{vu}$

The first constraint defines that the flow cannot exceed the capacity of an edge. The second that the sum of the flow that enter a node is equal to the sum of the flow that exit a node. If we imagine a flow graph as a road network where the edges are roads and nodes are intersection, these constraints describe the behavior of the traffic. The maximum amount of cars that can travel on a road is the capacity, while the flow is the amount of cars at a given time. Moreover, we can easily understand that the amount of cars that enter an intersection is equal to the amount that exit.

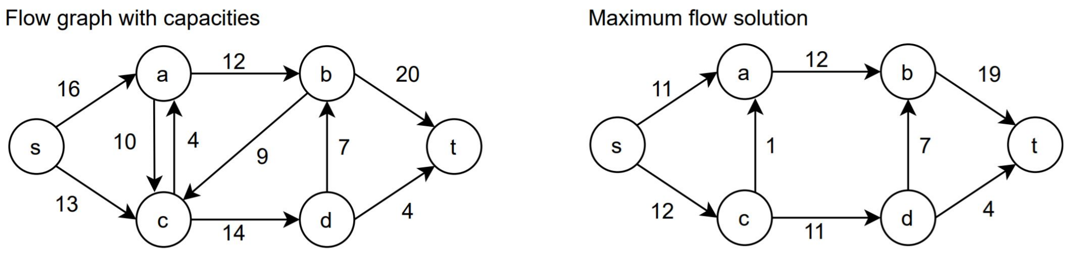


Figure 1: (Left) Example flow graph whose edges have an associated capacity. (Right) The maximum flow solution.

Fig 1 shows an example flow graph on the left and the maximum flow solution on the right. The value of flow is the sum of the flow that exist the source or that enter the target node $|f| = \sum_{v:(s,v)\in E} f_{s,v} = 23$

2.2 Minimum-cost maximum flow problem

The minimum-cost maximum flow problem is one specific type of network flow problem in which the input is again a flow graph but the edges have also a associated cost. Recalling the example of the road network, the cost associated to a road could be function of its length. Going from point A to point B should not only consider the traffic in a road but also the cost/time required to travel it.

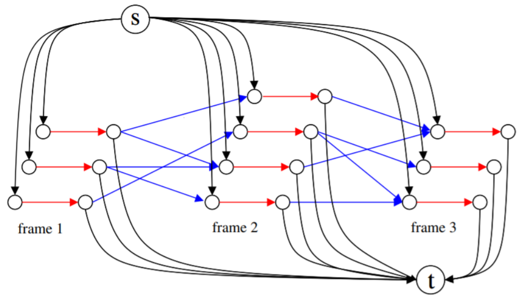
Formally, the graph can be defined as $G = (V, E, C)$ with real-valued edge cost function C. The constraints we saw befor hold here as well. The solution to this problem is to find the maximum flow among the set of maximum flow solution that has the lowest cost. The cost can be computed as $\sum_{(u,v)\in E} f_{uv}C(u, v)$.

The minimum cost circulation problem is one way to solve the minimum cost maximum flow problems but we do not give more details here as it goes out of the scope of this document. Instead, there are variants of the minimum cost maximum flow problem that can be solved with Integer Linear Programming (ILP) and other more efficient methods that exploit shortest paths. These will be discussed in the following section.

2.3 Minimum-cost maximum flow problem for multiple object tracking

The work of Zhang et al. [8] is the first, to the best of our knowledge, that shows how data association for multiple object tracking can be cast as a network flow problem. The optimal solution is a set of paths connecting source and target nodes and it is the one with minimum cost and maximum flow. This solution can be found by simultaneously solving for the number of objects and their trajectories according to a cost function that takes into account the likelihood of a detection (confidence) as well as pairwise relationships between detections for all consecutive frames. Different extensions [7, 2, 3, 4, 6, 5] demonstrate speedups by using more sophisticated solvers, propose slight variations of the formulation, and the use of motion and appearance cues to increase the robustness in challenging situations.

Figure 2 shows an example of such a flow graph used to solve multiple object tracking problems. The graph is a directed acyclic graph constructed from detections from multiple consecutive frames. Each detection is represented by two nodes (a pre-node and a-post node) where the edges represent possible associations. By definition, all edges have unitary capacity $c_{uv} = 1$ and the flow is binary $f_{uv} \in \{0, 1\}$. The reason behind this is two folds: 1) in conjunction with the constraints it prevents that trajectories share the same edges/nodes and 2) the problem can be solved efficiently using binary programming. The cost $C(u, v)$ of the edges connecting nodes in different frames (blue edges) defines how likely the two detections are to be the same object. The cost for the edges connecting the pre-node and the post-nodes (red edges) instead encodes the confidence of the detection. Both of these costs are negative values. The remaining edges connecting the source and sink node to the rest of the nodes encode a weight that is positive and that can be understood as the entry cost for creating a trajectory. As the "internal edges" have negative costs and the "external" ones positive, the sum of the costs along a path connecting source to sink may or may not be negative. Since the problem is minimization, it is clear that if a path has positive cost it will not be part of the solution.



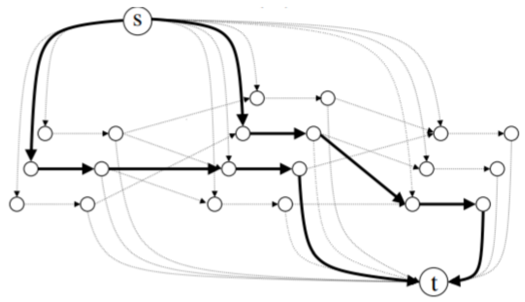


Figure 2: (Left) Typical flow graph over detections for multiple object tracking. (Right) Example solution.

A straightforward way to solve this flow graph is to use Integer Linear Programming (ILP) or more specifically Binary Integer Programming (BIP). Eq. 2.3 describe the binary integer programming formulation. It is a minimization over the costs, that also leads to the maximum flow solution due to the negative costs, with the constrains discussed in the previous section. In fact, as f_{uv} is a binary variable, the second constraint is unnecessary.

$$\begin{aligned}
&\underset{f}{\text{minimize}} && \sum_{(u,v)\in E} f_{uv} C(u,v) \\
&\text{subject to} && \sum_{u:(u,v)\in E} f_{uv} = \sum_{u:(v,u)\in E} f_{vu} \\
&\text{and} && f_{uv} \le c_{uv} \ \forall\ (u,v) \in E
\end{aligned}$$

ILP is an efficient solver for relatively small networks but it does not scale well for larger ones. Luckily, there are other ways to obtain the same optimal solution but at a fraction of the time required by ILP. These are based on shortest paths and are named Successive Shortest Paths (SSP) or k-shortest paths (KSP). [5] is the reference paper of the solver called **muSSP** that we use in this project. To keep this document compact, we do not explain how it works but we simply consider it as a black box that solves flow graphs.

3 Method

3.1 Defining the flow graph

The flow graph we use has the same structure seen in Fig. 2 with also skip connections. A skip connection is an edge that connects nodes/detections that belong to non adjacent frames. The reason is to compensate for possible missing detections. To reduce the complexity of the graph, edges are created only if the two nodes composing them are within a predefined maximum distance and time. An example is given in Fig. 3. In addition, when computing the cost $C(u,v)$ which can be function of time, distance, appearance and other, if any of these quantities or a function of them signal that the edge is highly likely to not be a good connection, it will be removed from the graph. These conditions can be adjusted or defined as needed.

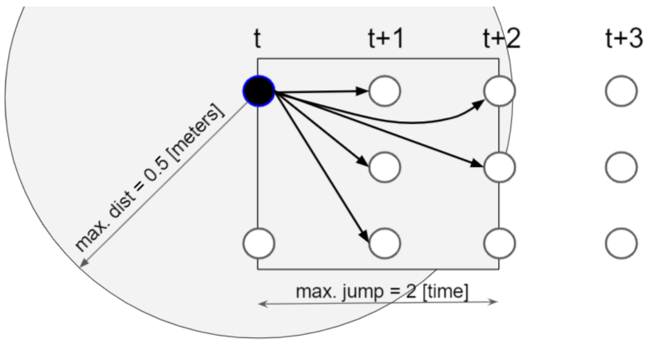


Figure 3: Example showing how connections from a node are created only ahead in time and limited by time and distance. In practice, other parameters such as the appearance (color and bounding box similarity) can be used to remove/create edges.

3.1.1 Computing the costs

The cost of an edge connecting two detections is a function of the distance between the two, the time and also the appearance. The appearance similarity can be computed from color histograms, the size of the bounding boxes or from features computed using deep neural networks. We will see later on that a nodes in the flow graph can also be a tracklets (small trajectories which is a chain of detections). The cost of the edges connecting them in this case can also take into account a motion model.

Let's consider only the distance between two detections for now. We define the cost of an edge as

$$C(u,v) = -\exp\left(\frac{-d^2}{\sigma^2}\right) \tag{1}$$

where d is the distance between the two detections and σ is a parameter to chose. As the problem is a minimization and that the cost of the edges are negative, the solver will try to activate/use as many edges as possible.

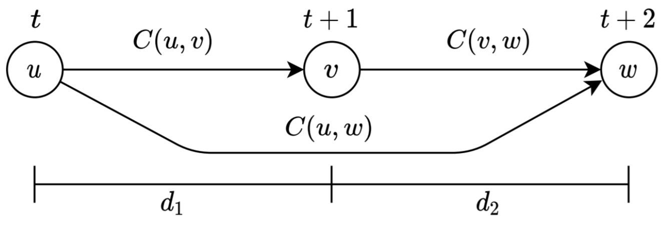


Figure 4: There are two possible solution to this graph; i) the edges connecting u and w is selected or ii) the two edges connecting all three nodes. If the node u,v and w are collinear in space, the cost functions we designed will be

The first question that arises from this is what happens with the skip connections in Fig. 4?. Since the number of inward and outward edges to a node can only be one because of the flow constraint, and, if there are no missing detections, does the algorithm prefer to pass from the skip connection? The answer is no. The algorithm will try to create the longest trajectory possible which is computed as the sum of the costs along the path. The sum of the costs of the two edges connecting u, v and v, w is always larger in magnitude to the cost of u, w if this triplet of nodes is collinear:

$$C(u,v) + C(v,w) < C(u,w) \tag{2}$$

$$-\exp\left(\frac{-d_1^2}{\sigma^2}\right) - \exp\left(\frac{-d_2^2}{\sigma^2}\right) < -\exp\left(\frac{-(d_1+d_2)^2}{\sigma^2}\right) \tag{3}$$

As we mentioned before, the flow graph can also be constructed on tracklets. A tracklet is a chain of detections forming a small trajectory. Fig. 5 shows example tracklets with the notion of tail and head nodes. One advantage of using tracklets is that we dispose of more information which leads to more robust costs; in addition the tracklets allow us to use a motion model that generate an additional cost useful for tracking.

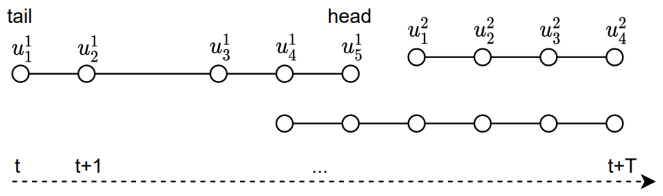


Figure 5: Tracklets. The head of a tracklet is the most recent node while the tail is the least recent one.

Cost based on distance. The cost function of the distance is the same we explained beforehand. We report it here for completeness. This can be used for detections and for tracklets. In the case of the tracklets, the distance is measured from the head to the tail if the tail of the second tracklet is ahead in time.

$$C_d(u,v) = -\exp\left(\frac{-d^2}{\sigma_d^2}\right) \tag{4}$$

Cost based on time. For the cost function of the time we use an exponential curve that encourage links between close detections

$$C_t(u,v) = -\exp(-(\Delta t - 1)\sigma_t) \tag{5}$$

Cost based on color appearance. To measure the color similarity between two image patches we first transform the RGB patch to LAB color space and compute an histogram of each channel as

$$s(H_1,H_2) = \frac{\sum_i (H_1(i) - \bar{H}_1)(H_2(i) - \bar{H}_2)}{\sqrt{\sum_i (H_1(i) - \bar{H}_1)^2 \sum_i (H_2(i) - \bar{H}_2)^2}} \tag{6}$$

$$\bar{H}_k = \frac{1}{n_k} \sum_i H_k(i) \tag{7}$$

then we take the cost as

$$C_s(u,v) = -\exp\left(\frac{-(1-s(H_1,H_2))^2}{\sigma_s^2}\right) \tag{8}$$

Cost based on bounding box similarity. To measure the bounding box similarity we use

$$b(w_1,w_2,h_1,h_2) = 1 - \frac{1}{2}\left[\frac{|w_1-w_2|}{\max(w_1,w_2)} + \frac{|h_1-h_2|}{\max(h_1,h_2)}\right] \tag{9}$$

$$C_b(u,v) = -\exp\left(\frac{-(1-b(w_1,w_2,h_1,h_2))^2}{\sigma_b^2}\right) \tag{10}$$

Cost based on linear motion model. To connect tracklets we measure how likely two tracklets are to follow the same motion. To do so, for both tracklets we project/predict future positions (or in the past) and measure how close these predicted locations are to the other trajectory. We use a linear model. If the sum of this two way distance is low, the tracklets are likely to correspond to the same object.

We take the final cost for of an edge connecting two detections or two tracklets as the product of the various costs i.e.

$$C(u,v) = C_d(u,v)C_t(u,v)C_s(u,v)C_b(u,v) \tag{11}$$

3.1.2 Effect of the costs on the solution

As we mentioned before, the cost of a trajectory in the flow graph is the sum of the costs associate to the edges composing it. The edge that connect the source to a pre-node and the edge that connect post-node to the target node are also part of the trajectory and have a positive weight instead. These positive costs are manually defined and can be understood as the entry cost for creating a trajectory. If the input data has many false positive detections, setting a low entry cost will promote the creation of spurious trajectories. If it is too high, small trajectories will be excluded from the solution. Fig. 6 shows the effect of these costs on the solution.

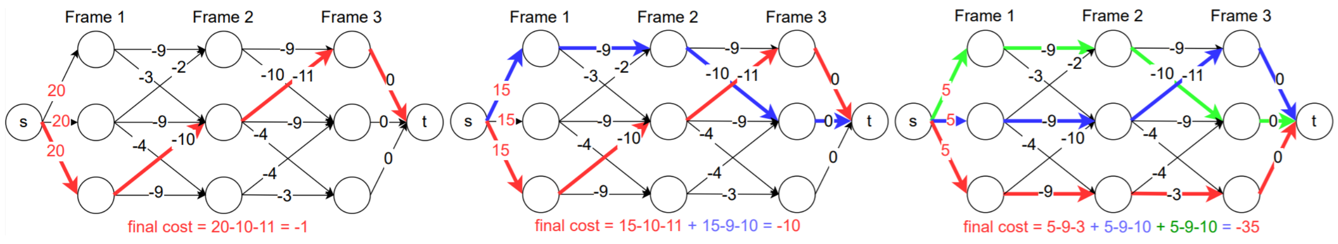

Frame 1
Frame 2
Frame 3
s
t
final cost = 20-10-11 = -1
final cost = 15-10-11 + 15-9-10 = -10
final cost = 5-9-3 + 5-9-10 + 5-9-10 = -35

Figure 6: The flow graph in these three example are the same except for the costs of the edges originating from the source node. In the first case the entry cost so high to the point where the solution is composed of only one trajectory. In the last case instead the entry cost is low and the solution comprises more trajectories. It is important to note here that the high cost trajectory of the first example is not part of the third example. All three solution are optimal.

3.2 Solving the flow graph

To solve the flow graph we rely on a solver based on shortest paths called Successive Shortest Paths (SSP). A fast and memory efficient version of SSP has been recently published in [5]. The C++ code is publicly available [1] under Apache License 2.0 which allows for commercial use. This solver has been integrated in our codebase through a wrapper.

3.3 The two steps approach

The trajectories describing the motion of airplanes and cars are smooth and predictable. The use of a motion model is therefore highly advantageous in this project. Since the minimum-cost flow formulation of MOT finds the globally optimal solution by performing local decision (decisions between pair of detections only), implementing a motion model in the form of an additional cost it is not straightforward. Adding smooth constraints on the trajectories would require computing pairwise costs (costs over three detections) at least which can likely only be solved with quadratic programming.

The solution we propose here is to perform tracking twice. In the first step we solve a flow graph whose nodes are detections, in the second a flow graph whose nodes are tracklets computed from the solution of the first. The cost of linking two tracklets can now take into consideration a motion model which would tells us how likely two tracklets follow the same motion. Note here that the time required to solve the second graph is much lower as the number of nodes composing it is one order of magnitude smaller than the first. Fig. 7 shows an illustration of this two steps approach. The solution of the first flow graph which is constructed from detections is a set of trajectories that closely follow the motion of the objects. In this step the parameters should be chosen so that the trajectoires may be disjointed but that correctly describe the motion of the obejcts. We should avoid here having ”swaps” of IDs. This trajectories are then split into small tracklets of roughly equal length and possibly processed in order to reduce noise or remove the one that are too short (i.e. of length 2 or less). With this new set of tracklets we now construct a second flow graph. The costs of the edges in this case are dependend on the motion model and the other features such as appearance, distance and time.

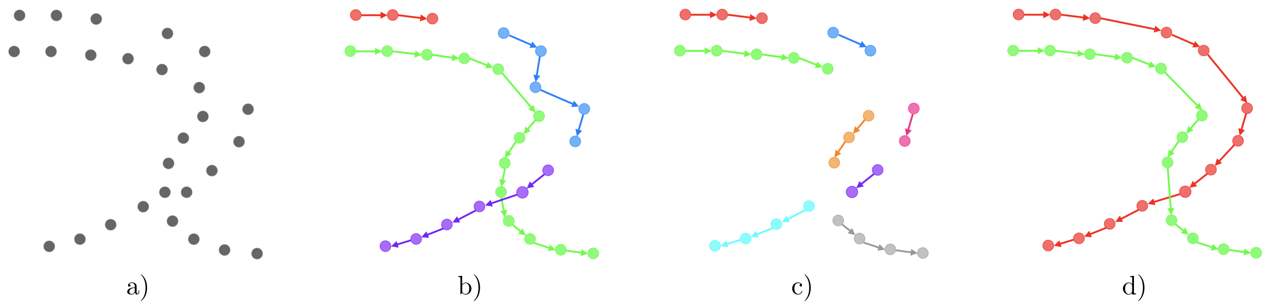


Figure 7: Illustration of the two steps approach. a) Set of initial detection for a batch of consecutive frames. b) Solution of the flow graph whose nodes are the detections. The parameters here should be set so that the resulting trajectories do not have ”swaps” of IDs which may also promote discontinuities. c) Splitting of the trajectories in roughly similar length and cleaning. d) Solution of the second flow graph whose nodes are tracklets. In this second pass we use a motion model.

References

[1] mussp solver. `https://github.com/yu-lab-vt/muSSP`.

[2] Berclaz J. B., Fleuret F., and Fua P. Multiple object tracking using flow linear programming. In *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 1–8, 2009.

[3] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *TPAMI 2011*, 33(9):1806–1819, 2011.

[4] Asad A. Butt and Robert T. Collins. Multi-target tracking by lagrangian relaxation to min-cost network flow. In *CVPR 2013*, pages 1846–1853, 2013.

[5] Wang C., Wang Y., Wang Y., Wu C., and Yu C. mussp: Efficient min-cost flow algorithm for multi-object tracking. In *NIPS*, 2019.

[6] Lenz P., Geiger A, and Urtasun R. Followme: Efficient online min-cost flow tracking with bounded memory and computation. In *CVPR 2015*, 2015.

[7] Hamed Pirsiavash, Deva Ramanan, and Charless C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR 2011*, pages 1201–1208, 2011.

[8] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.