muSSP: Efficient Min-cost Flow Algorithm for Multi-object Tracking

Congchao Wang[1], Yizhi Wang[1], Yinxue Wang[1], Chiung-Ting Wu[1] and Guoqiang Yu[1*]

[1] Virginia Polytechnic Institute and State University *Corresponding author (yug@vt.edu)

Introduction

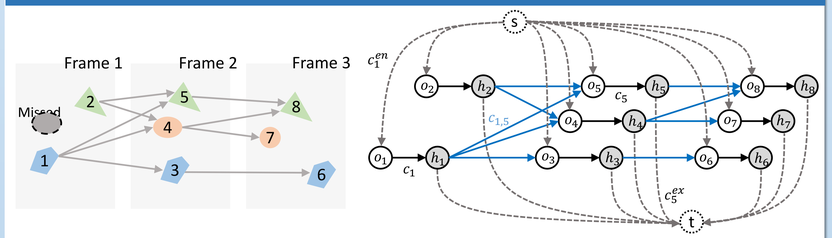
Motivation:

- Min-cost flow (MCF) has been widely used for solving data-association problem in multi-object tracking (MOT). Existing solvers for MCF problem in MOT either directly adoption or slight modifications of generic MCF algorithms, yielding sub-optimal efficiency. The low efficiency of existing MCF solvers holds back trackers to use sufficient long history frames for tracking
- The MCF problem in MOT owns **specialties** where dramatic computation should be saved

Minimum-update Successive Shortest Path(muSSP):

- An **exact** and **efficient** MCF solver for MOT based on successive shortest path (SSP), minimizing redundant computation by leveraging the problem's specialties
- Practically **hundreds** to **thousands** times faster than previous MCF solvers on five MOT benchmarks

Problem formulation



(a) Objects detected in three frames

(b) Typical flow network

Goal: Select a set of s-t paths with minimal total cost.

Specialties of the flow network in MOT:

1. Directed acyclic graph (DAG) with single source node ***s*** and single sink node ***t***
2. Unit-capacity graph
3. Each detection corresponds to a pair of nodes, a pre-node and a post-node, with a single arc linking them
4. Each pre-node is linked from ***s*** and each post-node is linked to ***t***

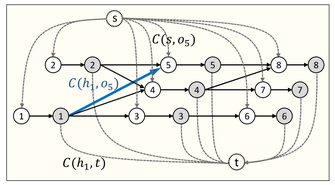
muSSP

SSP: Sequentially send flow along the shortest s-t path on residual graph.

Main idea: Dynamically maintain a shortest path tree of the residual graph while avoid computation that unrelated with finding the shortest s-t path

Directions of saving computation:

1. Increase the reuse of current shortest path tree
2. Decrease update frequency of shortest path tree
3. Accelerate the updating of shortest path tree



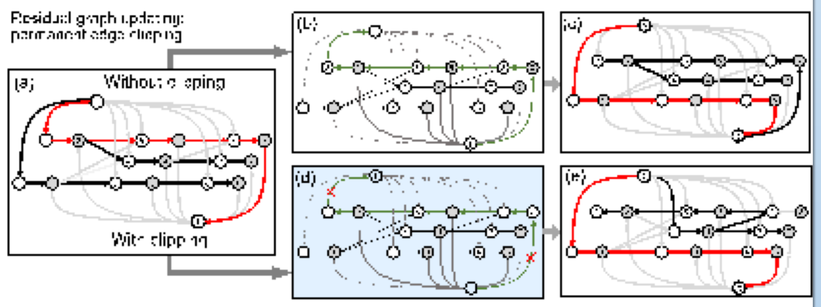
Dummy edge (blue)

Independent flipping lemma: Assume the shortest s-t path on current residual graph is $\pi = \{(s, v_0), (v_0, v_1), \dots, (v_x, t)\}$, sending flow along π only affects the decedent nodes of v_0. The shortest paths of other nodes do not change.

Strategy 1 and 2 are for minimizing the descendants number of v_0 and thus increase the reuse of current shortest path tree.

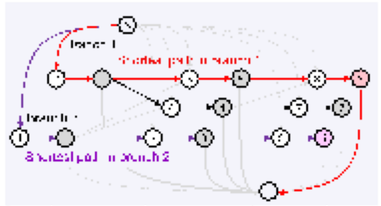
Strategy 1 (Dummy edge clipping): If the cost of linking two detections is too high, we can always have a better association where they are in different trajectories, so these arcs can be safely deleted (specialty 3 and 4)

Strategy 2 (Permanent edge clipping): Once an arc connected to either s or t becomes non-empty, the flow in it will always be 1 and these arcs can be pruned. (specialty 1 and 2)



Permanent edges (edges with crossings in (d))

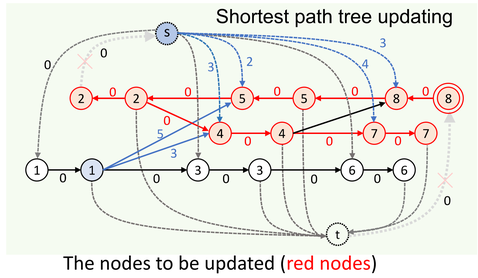
Strategy 3 (multi-path flipping): Specialty 1 and 4 enable us to simultaneously augment multiple shortest s-t paths and thus decrease the frequency of updating shortest path tree.



One shortest path tree contains two shortest path (red and purple)

The two s-t paths (s-2-5-8-t and s-1-3-6-t) are independent and shorter than all other s-t paths.

Strategy 4 (adapted Dijkstra's algorithm): The nodes to be reviewed for finding shortest s-t path form a zero-tree structure (specialty 1 and 2) and thus we can accelerate the updating of the shortest path tree.



The nodes to be updated (red nodes)

Zero-tree is a tree with zero cost for each of its arcs.
After sending flow along path s-2-5-8, the nodes to be updated form a zero-tree rooted at the double cycled 8 node.

Experiments

Solving the direct min-cost flow in MOT:

Table 1: Efficiency comparison on MOT datasets (in seconds)

Datasets	KITTI(DPM)				KITTI(reglets)			
	seq00	seq10	seq11	seq14	seq00	seq10	seq11	seq14
(a) graph design from [16]								
SSP	18.4(108)	142.8(269)	68.0(200)	120.8(448)	6.2(89)	9.1(129)	9.4(156)	31.8(245)
FollowMe	3.4(20)	12.7(24)	6.4(19)	12.0(44)	1.4(20)	1.8(26)	1.5(25)	6.4(49)
cs2	11.8(70)	88.4(167)	42.6(125)	41.1(152)	4.3(61)	7.2(103)	5.7(95)	10.3(79)
muSSP	**0.17(1)**	**0.53(1)**	**0.34(1)**	**0.27(1)**	**0.07(1)**	**0.07(1)**	**0.06(1)**	**0.13(1)**
(b) graph design from [14]								
SSP	20.9(116)	266.9(523)	77.9(223)	96.1(291)	9.4(117)	12.8(160)	18.1(227)	54.7(421)
FollowMe	3.2(18)	34.4(67)	9.6(28)	12.0(36)	3.8(48)	3.1(39)	5.9(74)	15.3(118)
cs2	13.6(76)	98.6(193)	45.1(129)	39.8(121)	5.0(63)	6.0(75)	6.3(78)	8.9(69)
muSSP	**0.18(1)**	**0.51(1)**	**0.35(1)**	**0.33(1)**	**0.08(1)**	**0.08(1)**	**0.08(1)**	**0.13(1)**
(c) graph design from [19]								
SSP	1.2h(24.5k)	10.0h(46.2k)	4.5h(26.0k)	5.2h(37.6k)	437.6(4.4k)	235.8(1.6k)	246.8(1.8k)	605.9(3.0k)
FollowMe	917.9(5.1k)	3.7h(16.9k)	1.0h(5.5k)	0.9h(6.7k)	198.4(2.0k)	224.5(1.5k)	242.2(1.7k)	649.1(3.2k)
cs2	24.6(137)	184.3(236)	78.7(127)	69.7(139)	5.9(59)	8.5(57)	7.7(55)	12.8(64)
muSSP	**0.18(1)**	**0.78(1)**	**0.62(1)**	**0.50(1)**	**0.10(1)**	**0.15(1)**	**0.14(1)**	**0.20(1)**
Datasets	CVPR19				ETHZ(DPM)		PTC	
	seq04	seq06	seq07	seq08	seq03	seq04	High	Mid
(a) graph design from [16]							(d) probability principled	
SSP	1.8h(4.9k)	370.9(976)	23.5(235)	136.2(619)	85.7(204)	173.0(455)	0.4h(106)	379.2(134)
FollowMe	2.4h(6.5k)	801.1(2.1k)	43.4(434)	220.6(1.0k)	21.2(50)	26.2(69)	1.2h(339)	0.3h(430)
cs2	441.3(337)	73.7(194)	16.2(162)	34.5(157)	39.4(94)	46.1(121)	71.5(5)	20.2(7)
muSSP	**1.31(1)**	**0.38(1)**	**0.10(1)**	**0.22(1)**	**0.42(1)**	**0.38(1)**	**13.10(1)**	**2.82(1)**
(b) graph design from [14]							Low	
SSP	1.6h(4.3k)	513.7(1.3k)	21.2(236)	38.5(167)	183.9(400)	137.3(490)	10.7(41)	
FollowMe	2.3h(6.2k)	803.3(2.0k)	36.2(402)	62.1(270)	53.1(115)	36.8(131)	40.4(155)	
cs2	137.2(103)	35.7(87)	6.7(74)	16.0(70)	63.1(137)	57.8(206)	3.0(12)	
muSSP	**1.33(1)**	**0.41(1)**	**0.09(1)**	**0.23(1)**	**0.46(1)**	**0.28(1)**	**0.25(1)**	

Solving the min-cost flow approximation to the high-order modeling in MOT:

Table 2: Efficiency comparison of solving quadratic programming problem

Method	SSP	FollowMe	cs2	**muSSP**
PETS S1.L1	391.9(19)	557.0(27)	613.1(30)	**20.3(1)**

Conclusion

We proposed an efficient yet exact min-cost flow solver muSSP for the data association in MOT. We expect this large degree of efficiency improvement will save computational time for existing applications, enable engineers to tackle larger scale of problems, and inspire researchers to build more accurate modeling for tracking, for instance, recruit more history frames in online tracking scenarios or refining iteratively the tracking results.