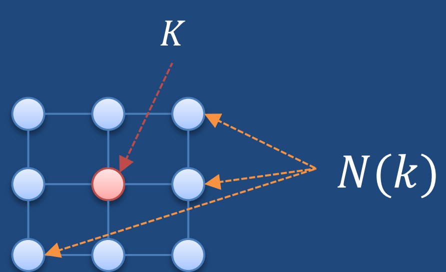
Modeling Multiple-object Tracking as Constrained Flow Optimization Problem

Introduction

- Multiple-object tracking
  1. Detection Step: time-Independent
  2. Linking Step: connect detections into most likely trajectories: NP-Complete
- Problem: Linking detections into trajectories for multiple *visually-similar* objects
- Solutions:
  - Filtering, Greedy dynamic programming etc.: don’t ensure global optimum
  - Integer Linear Programming (ILP): Ensures global optimum but NP-Complete

Multi-Object Tracking as Constrained Flow optimization

- Divide scene into k discrete locations.

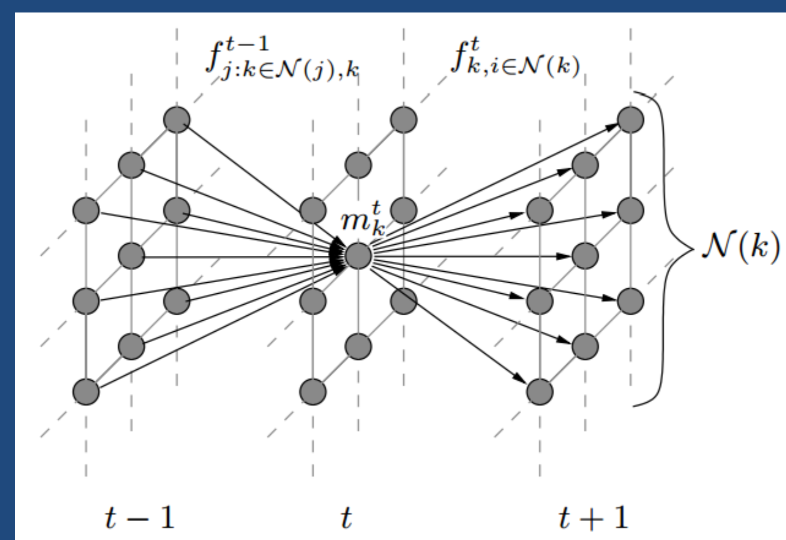


$$\forall j,t \sum_{i:j\in N(i)} f_{i,j}^{t-1} = m_j^t = \sum_{k\in N(j)} f_{j,k}^t$$

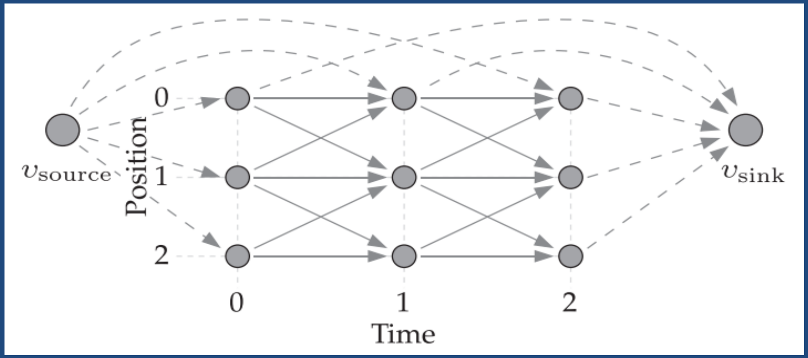
$$\forall k,t \sum_{j\in N(k)} f_{k,j}^t \leq 1$$

$$\forall k,j,t \quad f_{k,j}^t \geq 0$$

- Model occupancy map over time using directed graph.



Multi-Object Tracking as Constrained Flow optimization



- Posterior probability of an object:
  - m: occupancy map (holding variables m_j^t)
  - Ғ: set of feasible maps m.
  - M_i^t :random variable presenting the true value of i in time t.
  - I^t :signal

$$\sum_{j \in N(v_{source})} f_{v_{source},j}^t = \sum_{k: v_{sink} \in N(k)} f_{k,v_{sink}}^t$$

$$\rho_i^t = \hat{P}\left(M_i^t = 1 \middle| I^t\right)$$

$$m^* = \arg max_{m \in \text{Ғ}} \hat{P}(M = m | I^t)$$

- M_i^t as conditional independence variable

$$m^* = \arg max_{m \in \text{Ғ}} log \prod_{t,i} \hat{P}\left(M_i^t = m \middle| I^t\right) = \arg max_{m \in \text{Ғ}} \sum_{t,i} log \hat{P}\left(M_i^t = m \middle| I^t\right)$$

Integer Linear Programming (ILP) Formulation

- M_i^t as conditional independence variable

$$m^* = \arg max_{m \in \text{₣}} log \prod_{t,i} \hat{P}\left(M_i^t = m \middle| I^t\right) = \arg max_{m \in \text{₣}} \sum_{t,i} \left(log \frac{\rho_i^t}{1-\rho_i^t}\right) m_i^t$$

- **ILP System:**
  - *Maximize* $\sum_{t,i} \left(log \frac{\rho_i^t}{1-\rho_i^t}\right) \sum_{j \in N(i)} f_{i,j}^t$
  - *Subject to* $\forall i, j, t \quad f_{i,j}^t \geq 0$

$$\forall i, t \sum_{j \in N(i)} f_{i,j}^t \leq 1$$

$$\forall i, t \sum_{j \in N(i)} f_{i,j}^t - \sum_{k : i \in N(k)} f_{k,i}^{t-1} \leq 0$$

$$\sum_{k \in N(v_{source})} f_{j,k}^t - \sum_{i : j \in N(i)} f_{i,j}^{t-1} \leq 0$$

From Integer to Continuous Linear Program

- Integer LP solution: NP-complete problem
- Continuous LP: Polynomial time average complexity
- Relax the ‘integer’ condition to reduce complexity
- *Problem:* continuous LP does not usually converge to optimal solution of original ILP!!
- *Solution:* Total-unimodularity of constraint matrix

Total Unimodularity

- Total-Unimodular Matrix:
  - all square sub-matrices has determinants 1,0 or -1

or

  - For every subset of rows $R \subseteq \{1,2\ldots,m\}$, there is a partition of rows such that $R = R_1 \cup R_2$, $R_1 \cap R_2 = \emptyset$

$$\forall_{j = 1,2,\ldots n} \left(\Sigma_{i\in R1}\, a_{ij} - \Sigma_{i\in R2}\, a_{ij}\right) \in \{-1,0,1\}$$

- Ensures integer solution even for continuous LP
- Constraint matrix is Total-Unimodular

Total-Unimodularity of Constraint Matrix

- Constraint matrix:
  - arrange columns in ascending order of time; each column represent one location at one time instant
- Rows:
  - divided into 2 parts based on conditions

$$
\begin{array}{c}
\overbrace{\qquad}^{1} \;\cdots\; \underbrace{\overbrace{\qquad\qquad}^{t-1}}_{i} \;\; \underbrace{\overbrace{\qquad\qquad}^{t}}_{j} \;\cdots\; \overbrace{\qquad}^{T}
\\
\left[\begin{array}{cccccccccccc}
\vdots & \cdots & & \vdots & & & \vdots & & & & \vdots \\
 & & & \vdots & & & \vdots & & & & \\
0 & \cdots & \cdots & \cdots & 0 & \boxed{1 \cdots 1 \cdots 1} & & & 0 & \cdots & 0 \\
 & & & \vdots & & & \vdots & & & & \\
\vdots & \cdots & & \vdots & & & \vdots & & & & \vdots \\
\hline
\vdots & \cdots & & \vdots & & & \vdots & & & & \\
0 & \cdots & 0 & \boxed{-1 \cdots -1 \cdots -1} & & \boxed{1 \cdots 1 \cdots 1} & & & 0 & \cdots & 0 \\
\vdots & & & \vdots & & \vdots & \vdots & \vdots & & & \\
\vdots & & & & & \vdots & -1 & \vdots & & & \\
\vdots & & & & & -1 & \vdots & \vdots & & & \\
\vdots & & & & & \vdots & & -1 & & & \\
\vdots & \cdots & & & & & & \vdots & & & \vdots
\end{array}\right]
\begin{array}{l}
\left.\vphantom{\begin{array}{c}1\\1\\1\\1\\1\end{array}}\right\} U_1 \\
\left.\vphantom{\begin{array}{c}1\\1\\1\\1\\1\\1\\1\end{array}}\right\} U_2
\end{array}
\end{array}
$$

$$U_1 : \quad \{\textstyle\sum_{j \in \mathcal{N}(i)} f_{i,j}^{t} \leq 1\}, \forall t, i$$

$$U_2 : \quad \{\textstyle\sum_{j \in \mathcal{N}(i)} f_{i,j}^{t} - \sum_{k : i \in \mathcal{N}(k)} f_{k,i}^{t-1} \leq 0\}, \forall t, i$$

$$\textstyle\sum_{j \in \mathcal{N}(v_{\text{source}})} f_{v_{\text{source}},j} - \sum_{k : v_{\text{sink}} \in \mathcal{N}(k)} f_{k,v_{\text{sink}}} \leq 0$$

Total-Unimodularity of Constraint Matrix

- Only 3 rows can be non-zero ($\in \{-1,0,1\}$) for one column
- Eight cases

  for partitions

$\{a_{ij} \mid i \in R_1\}$	$\{a_{ij} \mid i \in R_2\}$	$\sum_{i\in R_1} a_{ij} - \sum_{i \in R_2} a_{ij}$
$\{0,...,0,1\}$	$\{0,...,0\}$	1
$\{0,...,0,1\}$	$\{0,...,0,1\}$	0
$\{0,...,0,1\}$	$\{0,...,0,-1\}$	2
$\{0,...,0,1\}$	$\{0,...,0,1,-1\}$	1
$\{0,...,0\}$	$\{0,...,0\}$	0
$\{0,...,0\}$	$\{0,...,0,1\}$	-1
$\{0,...,0\}$	$\{0,...,0,-1\}$	1
$\{0,...,0\}$	$\{0,...,0,1,-1\}$	0

  $R_1 = R \cap U_1, R_2 = R \cap U_2$ for any $R \subseteq \{1,2...,m\}$

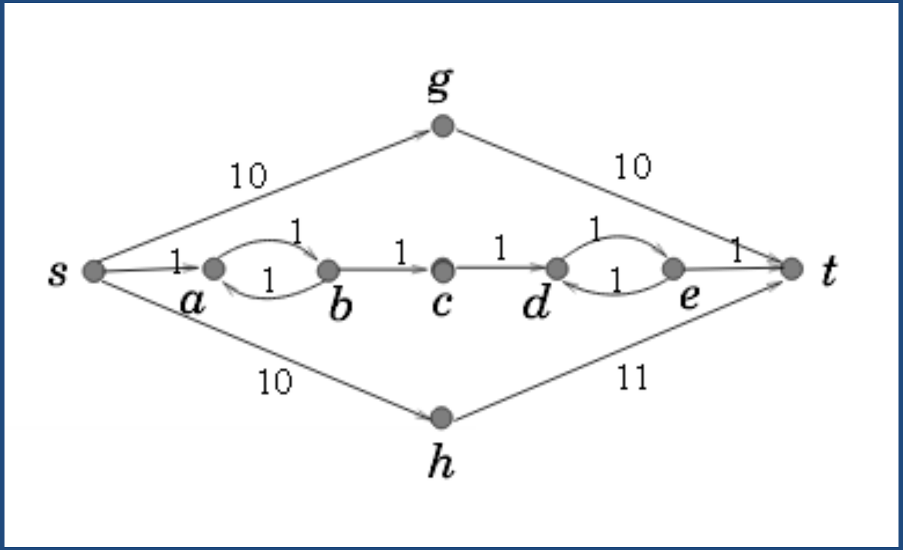
- Every possibility satisfies total-unimodualrity but 3^{rd} –*Problem*
- *Solution:* Move non-zero row of R_1 to R_2 for 3^{rd} case

K-shortest paths (KSP) formulation

- Why?
  - Relaxed ILP solution polynomial but practically not efficient!
  - Need real time efficiency for practical problems
- KSP:
  - Given a graph G(V,E), compute a set of k shortest paths {p1,p2,…..,pk} such that the total cost is minimum

KSP formulation

– Problem constraints:
  - Node disjoint
  - Node simple



Three node-simple shortest paths: 6,20,21

Three non-simple shortest paths (allowing loops ‘aba’ & ‘ded’):6,8,10

KSP formulation

- ILP to KSP
  - Maximizing flow in ILP is equivalent to minimizing path cost function in KSP (just negate the ILP objective function)

$$c(e_{i,j}^t) = -\log\left(\frac{\rho_i^t}{1-\rho_i^t}\right)$$

$$\mathbf{f}^* = \arg\min_{\mathbf{f}\in\mathfrak{H}} \sum_{t,i} c(e_{i,j}^t) \sum_{j\in\mathcal{N}(i)} f_{i,j}^t$$

  - Any path between source and sink nodes with arbitrary 'k' is in the set of feasible solutions of ILP
  - The value of 'k' that achieves min. cost is the one that maximizes the flow in ILP solution

KSP formulation

- Optimality of ‘k’ is guaranteed due to convexity of the path cost function

$$\text{cost}(P_l) = \sum_{i=1}^{l} \text{cost}(p_i^*)$$

- The shortest paths at each iteration are computed by Dijkstra’s algorithm
  - Complexity?
    - O(k(m+nlogn))

Applications

- 2D segmentation to 3D segmentation:
  - 2D: shortest path problem; Dijkstra's algorithm
  - 3D: minimal weight surface problem
    - can be presented as instance of ILP with totally unimodular constraint matrix
- Optimization by LP:
  - LP provides an upper/ lower bound for original ILP
  - Branch-and-bound: optimization of ILP through recursive LP solution.
- Tracking multiple humans in surveillance videos
- Min-Cost flow problems: e.g. efficient network routing

Question??

- How does Total-Unimodulairty ensure Integer solution even for continuous linear program??

Answer: Total-Unimodularity -> Integer Solution

- Quadratic system: $Cx = y$
- Cramer's rule: $x_j = |C_y^j| / |C|$
- C_y^j = C with j^{th} column replaced with y, $(C_1, C_2, ..C_{j-1}, y, C_{j+1}, ..C_m)$
- $|C_y^j| = \Sigma_i (-1)^{i+j} y_i |C_{ij}|$
- C_{ij} = C with i^{th} row, j^{th} column deleted
- If $|C| \in \{-1,1\}$ and $C_{ij} \in \{-1,0,1\}$ for all I and j, then x is integer => *Total Unimodularity, Integer solution*

Question??

- Why is the path cost function of KSP convex?

Answer: Edge costs can be negative and total path cost function is summation for successive shortest path costs that are monotonically increasing!

- At each iteration for 'k', we have:

$$\mathrm{cost}(p^*_{i+1}) \geq \mathrm{cost}(p^*_i) \qquad \forall i$$

$$\mathrm{cost}(P_l) = \sum_{i=1}^{l} \mathrm{cost}(p^*_i)$$

- Therefore, the path cost function over 'k' is convex

$$\mathrm{cost}(P_{k^*-1}) \geq \mathrm{cost}(P_{k^*}) \leq \mathrm{cost}(P_{k^*+1})$$