Faster MAP inference?

- We’ve now seen two algorithms for MAP inference:
  - Variable elimination: Exact, but potentially very slow
  - Loopy Max-product BP: Fast, but approximate

- It turns out that in some cases, MAP problems are easier than Marginal inference problems
  - One interesting case: With binary random variables, and potential functions that satisfy (relatively weak) restrictions, exact inference on a pairwise Markov network is efficient

Consider a simpler problem...

- Suppose we want to find a bright object against a dark background
  - But some of the pixel values are slightly wrong
  - So want to estimate a 0 or 1 label for each pixel

A slightly more interesting problem...

- Foreground vs background segmentation
  - We want to label every pixel of an image with a 0 or a 1, indicating whether it’s a background or foreground pixel

A slightly more interesting problem...

- One approach: Use some human input
  - Use sketches out a few strokes to indicate foreground and background, then we try to classify rest of pixels

Solving with an MRF
**Data cost**

\[ D(f_y, Y) \]

Adapted from N. Snavely’s slide

**Solving with an MRF**

- So, we want to solve a problem of the form:

\[ X^* = \arg \min_X \sum_i D_i(X_i, Y_i) + \sum_{(i,j) \in E} V(X_i, X_j) \]

- \( Y \) variables are given
- \( X \) variables are binary-valued
- \( D \) cost functions have any form
- \( V \) cost functions have the form:

\[ V(X_i, X_j) = \begin{cases} 0 & \text{if } X_i = X_j \\ k & \text{otherwise} \end{cases} \]

Observed pixel data
Unobservable binary labels

**Network flow can help**

- The minimization problem with 2 labels can be solved exactly using network flow
  - Construction probably due to [Hammer et al. 65]
  - First applied to images by [Greig et al. 86]
- Classical Computer Science problem reduction
  - Turn a new problem into a problem we can solve!

**Maximum flow problem**

- Max flow problem:
  - Each edge is a “pipe”
  - Find the largest flow \( F \) of “water” that can be sent from the “source” to the “sink” along the pipes
  - Source output + sink input = flow value
  - Edge weights give the pipe’s capacity

**Minimum cut problem**

- Min cut problem:
  - Find the cheapest way to cut the edges so that the “source” is separated from the “sink”
  - Cut edges going from source side to sink side
  - Edge weights now represent cutting “costs”

**Max flow/Min cut theorem**

- Max Flow = Min Cut:
  - Proof sketch: value of a flow is value over any cut
  - Maximum flow saturates the edges along the minimum cut
  - Ford and Fulkerson, 1962
  - Problem reduction!

Ford and Fulkerson gave first polynomial time algorithm for globally optimal solution
“Augmenting Path” algorithms

- Find a path from S to T along non-saturated edges
- Increase flow along this path until some edge saturates

A graph with two terminals

Adapted from R. Zabih’s slide

Min flow algorithms

- Ford-Fulkerson (1962) is the classic algorithm
  - Takes time $O(|E| f)$, where $f$ is the maximum flow
  - May not converge in some cases
- Edmonds-Karp (1972) gave an improved version
  - Same as F-F, but the augmented path is always the shortest with available capacity. Can be found using breadth-first search.
  - Takes time $O(|V| |E|^2)$

Back to binary MRFs...

- We want to solve a problem of the form:

$$X^* = \arg \min_X \sum_i D_i(X_i, Y_i) + \sum_{(i,j) \in E} V(X_i, X_j)$$

  - Y variables are given
  - X variables are binary-valued
  - D cost functions have any form
  - V cost functions have the form:
    $$V(X_i, X_j) = \begin{cases} 0 & \text{if } X_i = X_j \\ 1 & \text{otherwise} \end{cases}$$

Observed pixel data

Unobservable binary labels

Basic graph cut construction

- One non-terminal vertex per pixel
  - Each pixel connects directly to s, t, and to its neighbors
  - Edge to t has weight $D_p(0)$, edge to s has weight $D_p(1)$
  - Edge $(p,q)$ has weight $V_{pq}(0,1)$
- Cost of cut is the cost of the entire labeling
  - Pixel p labeled 1 if connected to t, labeled 0 if connected to s

$$E(x_1, \ldots, x_n) = \sum_P D_p(x_p) + \sum_{P \ni p} V_{pq}(x_p, x_q)$$

A cut

Adapted from R. Zabih’s slide