Semi-Markov/Graph Cuts

Alireza Shafaei

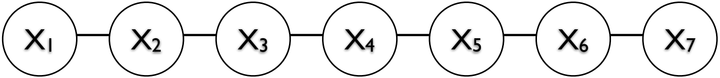
University of British Columbia

August, 2015

A Quick Review

- For a general chain-structured UGM we have:

$$p(x_1, x_2, \dots, x_n) \propto \prod_{i=1}^{n} \phi_i(x_i) \prod_{i=2}^{n} \phi_{i,i-1}(x_i, x_{i-1}),$$



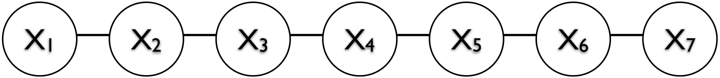
- In this case we only have local Markov property,

$$x_i \perp x_1, \dots, x_{i-2}, x_{i+2}, \dots, x_n | x_{i-1}, x_{i+1},$$

A Quick Review

- For a general chain-structured UGM we have:

$$p(x_1, x_2, \dots, x_n) \propto \prod_{i=1}^{n} \phi_i(x_i) \prod_{i=2}^{n} \phi_{i,i-1}(x_i, x_{i-1}),$$



- In this case we only have local Markov property,

$$x_i \perp x_1, \dots, x_{i-2}, x_{i+2}, \dots, x_n | x_{i-1}, x_{i+1},$$

- Local Markov property in general UGMs:
  - given neighbours, conditional independence of other nodes.

(Marginal independence corresponds to reachability.)

A Quick Review

- For chain-structured UGMs we learned the Viterbi decoding algorithm.
  - Forward phase:

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \max_{s'}\{\phi_i(s)\phi_{i,i-1}(s,s')V_{i-1,s'}\},$$

A Quick Review

- For chain-structured UGMs we learned the Viterbi decoding algorithm.
  - Forward phase:

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \max_{s'}\{\phi_i(s)\phi_{i,i-1}(s,s')V_{i-1,s'}\},$$

  - Backward phase: backtrack through argmax values.
  - Solves the decoding problem in $O(ns^2)$ instead of $O(s^n)$.

A Quick Review

- For inference in chain-structured UGMs we learned the forward-backward algorithm.

A Quick Review

- For inference in chain-structured UGMs we learned the forward-backward algorithm.
  1. Forward phase (sums up paths from the beginning):

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \sum_{s'} \phi_i(s)\phi_{i,i-1}(s,s')V_{i-1,s'}, \quad Z = \sum_s V_{n,s}.$$

A Quick Review

- For inference in chain-structured UGMs we learned the forward-backward algorithm.
  1. Forward phase (sums up paths from the beginning):

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \sum_{s'} \phi_i(s)\phi_{i,i-1}(s,s')V_{i-1,s'}, \quad Z = \sum_s V_{n,s}.$$

  2. Backward phase: (sums up paths to the end):

$$B_{n,s} = 1, \quad B_{i,s} = \sum_{s'} \phi_{i+1}(s')\phi_{i+1,i}(s',s)B_{i+1,s'}.$$

A Quick Review

- For inference in chain-structured UGMs we learned the forward-backward algorithm.
  1. Forward phase (sums up paths from the beginning):
     $$V_{1,s} = \phi_1(s), \quad V_{i,s} = \sum_{s'} \phi_i(s)\phi_{i,i-1}(s,s')V_{i-1,s'}, \quad Z = \sum_{s} V_{n,s}.$$
  2. Backward phase: (sums up paths to the end):
     $$B_{n,s} = 1, \quad B_{i,s} = \sum_{s'} \phi_{i+1}(s')\phi_{i+1,i}(s',s)B_{i+1,s'}.$$
  3. Marginals are given by $p(x_i = s) \propto V_{i,s}B_{i,s}$.

A Quick Review

- For chain-structured UGMs we learned the forward-backward and Viterbi algorithm.
- The same idea was generalized for tree-structured UGMs.

A Quick Review

- For chain-structured UGMs we learned the forward-backward and Viterbi algorithm.
- The same idea was generalized for tree-structured UGMs.
- For graphs with small cutset we learned the Cutset Conditioning method.

A Quick Review

- For chain-structured UGMs we learned the forward-backward and Viterbi algorithm.
- The same idea was generalized for tree-structured UGMs.
- For graphs with small cutset we learned the Cutset Conditioning method.
- For graphs with small tree-width we learned the Junction Tree method.

A Quick Review

- For chain-structured UGMs we learned the forward-backward and Viterbi algorithm.
- The same idea was generalized for tree-structured UGMs.
- For graphs with small cutset we learned the Cutset Conditioning method.
- For graphs with small tree-width we learned the Junction Tree method.
- Two more group of problems that we can deal with *exactly* in *polynomial* time.

A Quick Review

- For chain-structured UGMs we learned the forward-backward and Viterbi algorithm.
- The same idea was generalized for tree-structured UGMs.
- For graphs with small cutset we learned the Cutset Conditioning method.
- For graphs with small tree-width we learned the Junction Tree method.
- Two more group of problems that we can deal with *exactly* in *polynomial* time.
  - Semi-Markov chain-structured UGMs.

A Quick Review

- For chain-structured UGMs we learned the forward-backward and Viterbi algorithm.
- The same idea was generalized for tree-structured UGMs.
- For graphs with small cutset we learned the Cutset Conditioning method.
- For graphs with small tree-width we learned the Junction Tree method.
- Two more group of problems that we can deal with *exactly* in *polynomial* time.
  - Semi-Markov chain-structured UGMs.
  - Binary and attractive state UGMs.

Semi-Markov chain-structured UGMs

- Local Markov property in general chain-structured UGMs:
  - Given neighbours, we have conditional independence of other nodes.

Semi-Markov chain-structured UGMs

- Local Markov property in general chain-structured UGMs:
  - Given neighbours, we have conditional independence of other nodes.
- In Semi-Markov chain-structured models:
  - Given neighbours and their lengths, we have conditional independence of other nodes.

Semi-Markov chain-structured UGMs

- Local Markov property in general chain-structured UGMs:
  - Given neighbours, we have conditional independence of other nodes.
- In Semi-Markov chain-structured models:
  - Given neighbours and their lengths, we have conditional independence of other nodes.
- A subsequence of nodes can have the same state.
  - You can encourage smoothness.

Semi-Markov chain-structured UGMs

- Local Markov property in general chain-structured UGMs:
  - Given neighbours, we have conditional independence of other nodes.
- In Semi-Markov chain-structured models:
  - Given neighbours and their lengths, we have conditional independence of other nodes.
- A subsequence of nodes can have the same state.
  - You can encourage smoothness.
- Useful when you wish to keep track of how long you have been staying on the same state.

Semi-Markov chain-structured UGMs

- Previously, the potential of each edge was a function of neighboring vertices $\phi_{i,i-1}(s, s')$.

Semi-Markov chain-structured UGMs

- Previously, the potential of each edge was a function of neighboring vertices $\phi_{i,i-1}(s, s')$.
- In Semi-Markov chain-structured models we define the potential as $\phi_{i,i-1}(s, s', l)$

Semi-Markov chain-structured UGMs

- Previously, the potential of each edge was a function of neighboring vertices $\phi_{i,i-1}(s, s')$.
- In Semi-Markov chain-structured models we define the potential as $\phi_{i,i-1}(s, s', l)$
  - The potential of making a transition from s' to s after l steps.

Semi-Markov chain-structured UGMs

- Previously, the potential of each edge was a function of neighboring vertices $\phi_{i,i-1}(s, s')$.
- In Semi-Markov chain-structured models we define the potential as $\phi_{i,i-1}(s, s', l)$
  - The potential of making a transition from s' to s after l steps.
- You can encourage staying in certain states for a period of time.

Decoding the Semi-Markov chain-structured UGMs

- Let us look at the Viterbi decoding again:

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \phi_i(s) \cdot \max_{s'}\{\phi_{i,i-1}(s, s') \cdot V_{i-1,s'}\},$$

Decoding the Semi-Markov chain-structured UGMs

- Let us look at the Viterbi decoding again:

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \phi_i(s) \cdot \max_{s'}\{\phi_{i,i-1}(s,s') \cdot V_{i-1,s'}\},$$

- How can we update the formula to solve Semi-Markov chain structures?

Decoding the Semi-Markov chain-structured UGMs

- Let us look at the Viterbi decoding again:

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \phi_i(s) \cdot \max_{s'} \{\phi_{i,i-1}(s, s') \cdot V_{i-1,s'}\},$$

- How can we update the formula to solve Semi-Markov chain structures?

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \phi_i(s) \cdot \max_{s',l} \{\phi_{i,i-1}(s, s', l) \cdot V_{i-l,s'}\},$$

Decoding the Semi-Markov chain-structured UGMs

- Let us look at the Viterbi decoding again:

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \phi_i(s) \cdot \max_{s'} \{\phi_{i,i-1}(s, s') \cdot V_{i-1,s'}\},$$

- How can we update the formula to solve Semi-Markov chain structures?

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \phi_i(s) \cdot \max_{s', l} \{\phi_{i,i-1}(s, s', l) \cdot V_{i-l,s'}\},$$

- Depending on the application we can bound the maximum possible value of l to be L.

Decoding the Semi-Markov chain-structured UGMs

- Let us look at the Viterbi decoding again:

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \phi_i(s) \cdot \max_{s'} \{\phi_{i,i-1}(s, s') \cdot V_{i-1,s'}\},$$

- How can we update the formula to solve Semi-Markov chain structures?

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \phi_i(s) \cdot \max_{s',l} \{\phi_{i,i-1}(s, s', l) \cdot V_{i-l,s'}\},$$

- Depending on the application we can bound the maximum possible value of l to be L.
- For the unbounded case, L is simply n, the total length of chain.

Decoding the Semi-Markov chain-structured UGMs

- Let us look at the Viterbi decoding again:

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \phi_i(s) \cdot \max_{s'}\{\phi_{i,i-1}(s,s') \cdot V_{i-1,s'}\},$$

- How can we update the formula to solve Semi-Markov chain structures?

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \phi_i(s) \cdot \max_{s',l}\{\phi_{i,i-1}(s,s',l) \cdot V_{i-l,s'}\},$$

- Depending on the application we can bound the maximum possible value of l to be L.
- For the unbounded case, L is simply n, the total length of chain.
- **Note** that it is *different* from having an order-L Markov chain (why?).

Inference in the Semi-Markov chain-structured UGMs

- Forward-backward algorithm for the Semi-Markov models:
  - Forward phase (sums up paths from the beginning):

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \phi_i(s) \cdot \sum_{s',l} \phi_{i,i-1}(s, s', l) V_{i-l,s'}, \quad Z = \sum_s V_{n,s}.$$

Inference in the Semi-Markov chain-structured UGMs

- Forward-backward algorithm for the Semi-Markov models:
  - Forward phase (sums up paths from the beginning):

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \phi_i(s) \cdot \sum_{s',l} \phi_{i,i-1}(s, s', l) V_{i-l,s'}, \quad Z = \sum_s V_{n,s}.$$

  - Backward phase: (sums up paths to the end):

$$B_{n,s} = 1, \quad B_{i,s} = \sum_{s',l} \phi_{i+1}(s') \phi_{i+1,i}(s', s, l) B_{i+l,s'}.$$

Inference in the Semi-Markov chain-structured UGMs

- Forward-backward algorithm for the Semi-Markov models:
  - Forward phase (sums up paths from the beginning):

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \phi_i(s) \cdot \sum_{s',l} \phi_{i,i-1}(s,s',l) V_{i-l,s'}, \quad Z = \sum_s V_{n,s}.$$

  - Backward phase: (sums up paths to the end):

$$B_{n,s} = 1, \quad B_{i,s} = \sum_{s',l} \phi_{i+1}(s') \phi_{i+1,i}(s',s,l) B_{i+l,s'}.$$

  - Marginals are given by $p(x_i = s) \propto V_{i,s} B_{i,s}$.

Inference in the Semi-Markov chain-structured UGMs

- Forward-backward algorithm for the Semi-Markov models:
  - Forward phase (sums up paths from the beginning):

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \phi_i(s) \cdot \sum_{s',l} \phi_{i,i-1}(s, s', l) V_{i-l,s'}, \quad Z = \sum_s V_{n,s}.$$

  - Backward phase: (sums up paths to the end):

$$B_{n,s} = 1, \quad B_{i,s} = \sum_{s',l} \phi_{i+1}(s') \phi_{i+1,i}(s', s, l) B_{i+l,s'}.$$

  - Marginals are given by $p(x_i = s) \propto V_{i,s} B_{i,s}$.
- Questions?

Solving with graph cuts

- Restricting the structure of graph is just one way to simplify our tasks.

Solving with graph cuts

- Restricting the structure of graph is just one way to simplify our tasks.
- We can also restrict the potentials.

Solving with graph cuts

- Restricting the structure of graph is just one way to simplify our tasks.
- We can also restrict the potentials.
- Here we look at a group of pairwise UGMs with the following restrictions:

Solving with graph cuts

- Restricting the structure of graph is just one way to simplify our tasks.
- We can also restrict the potentials.
- Here we look at a group of pairwise UGMs with the following restrictions:
  1. Binary variables.

Solving with graph cuts

- Restricting the structure of graph is just one way to simplify our tasks.
- We can also restrict the potentials.
- Here we look at a group of pairwise UGMs with the following restrictions:
  1. Binary variables.
  2. Pairwise potential makes a submodular problem.

Solving with graph cuts

- Restricting the structure of graph is just one way to simplify our tasks.
- We can also restrict the potentials.
- Here we look at a group of pairwise UGMs with the following restrictions:
  1. Binary variables.
  2. Pairwise potential makes a submodular problem.
- Can be decoded by reformulation as a Max-Flow problem.

Solving with graph cuts

- Restricting the structure of graph is just one way to simplify our tasks.
- We can also restrict the potentials.
- Here we look at a group of pairwise UGMs with the following restrictions:
  1. Binary variables.
  2. Pairwise potential makes a submodular problem.
- Can be decoded by reformulation as a Max-Flow problem.
- Can be generalized to non-binary cases.

Solving with graph cuts

- Restricting the structure of graph is just one way to simplify our tasks.
- We can also restrict the potentials.
- Here we look at a group of pairwise UGMs with the following restrictions:
  1. Binary variables.
  2. Pairwise potential makes a submodular problem.
- Can be decoded by reformulation as a Max-Flow problem.
- Can be generalized to non-binary cases.
- Can be 2-approximated when not submodular (under a different constraint).

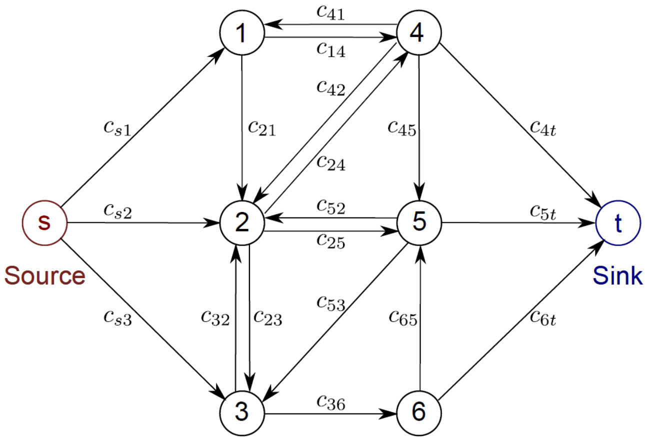
Solving with graph cuts

- Restricting the structure of graph is just one way to simplify our tasks.
- We can also restrict the potentials.
- Here we look at a group of pairwise UGMs with the following restrictions:
  1. Binary variables.
  2. Pairwise potential makes a submodular problem.
- Can be decoded by reformulation as a Max-Flow problem.
- Can be generalized to non-binary cases.
- Can be 2-approximated when not submodular (under a different constraint).
- In the general case it is known to be NP-Hard.

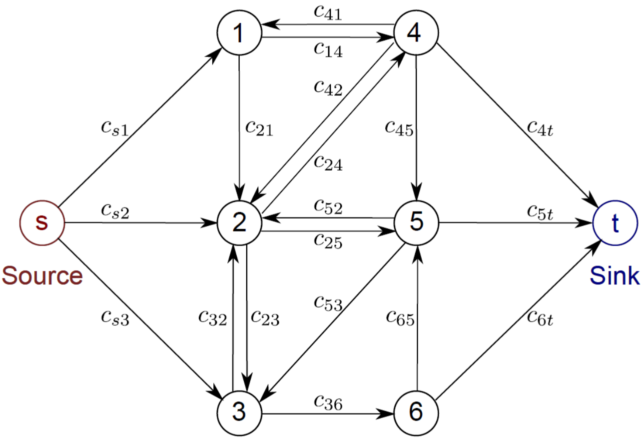
Solving with graph cuts

- Restricting the structure of graph is just one way to simplify our tasks.
- We can also restrict the potentials.
- Here we look at a group of pairwise UGMs with the following restrictions:
  1. Binary variables.
  2. Pairwise potential makes a submodular problem.
- Can be decoded by reformulation as a Max-Flow problem.
- Can be generalized to non-binary cases.
- Can be 2-approximated when not submodular (under a different constraint).
- In the general case it is known to be NP-Hard.
- The following material is borrowed from Simon Prince's (@UCL) slides. Available at computervisionmodels.com

The Max-Flow problem

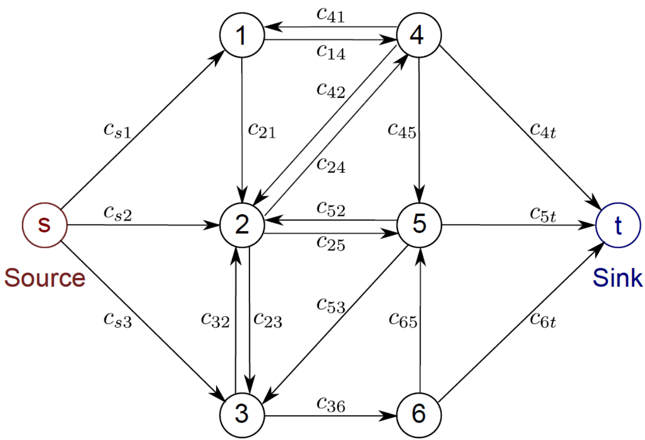


The Max-Flow problem



- The goal is to push as much ’flow’ as possible through the directed graph from the source to the sink.

The Max-Flow problem



- The goal is to push as much 'flow' as possible through the directed graph from the source to the sink.
- Cannot exceed the (non-negative) capacities C_{ij} associated with each edge.

Saturation

- When we push the maximum flow from source to sink:
  - There must be at least one saturated edge on any path from source to sink, otherwise you can push more flow.

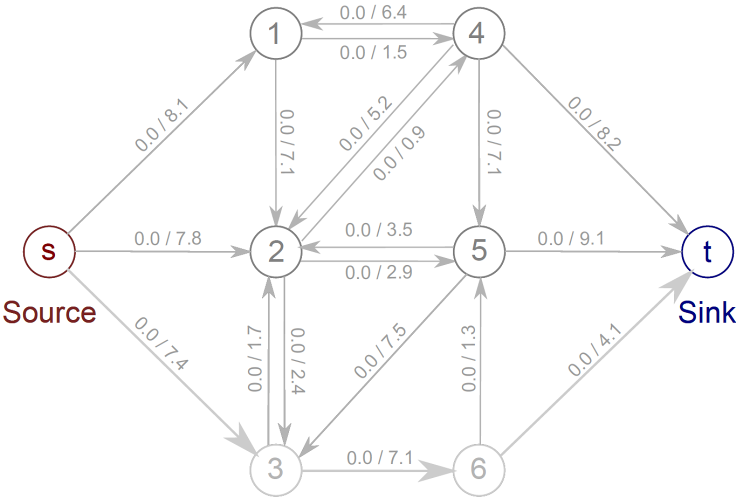
Saturation

- When we push the maximum flow from source to sink:
  - There must be at least one saturated edge on any path from source to sink, otherwise you can push more flow.
  - The set of saturated edges hence separate the source and sink.

Saturation

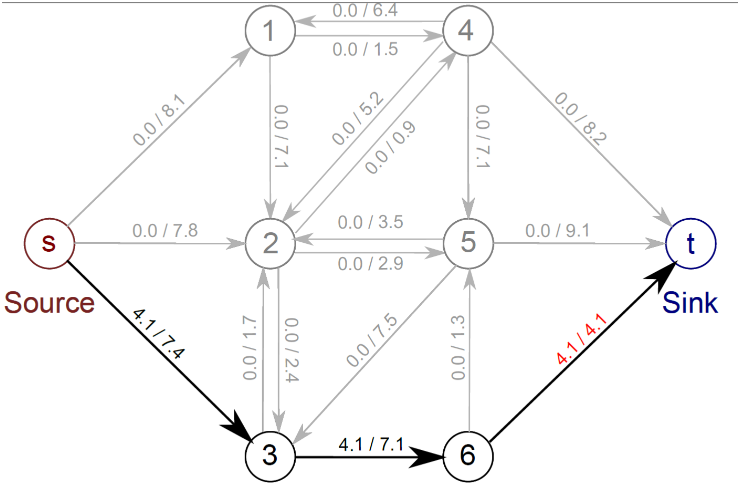
- When we push the maximum flow from source to sink:
  - There must be at least one saturated edge on any path from source to sink, otherwise you can push more flow.
  - The set of saturated edges hence separate the source and sink.
  - This set is simultaneously the min-cut and the max-flow.

An example



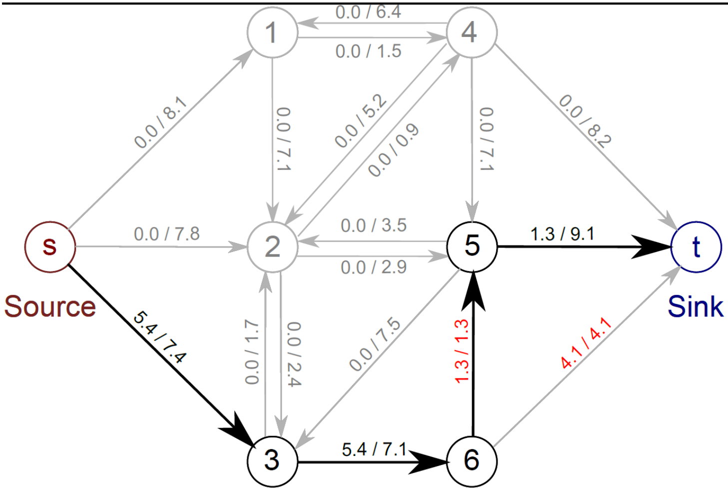
- Two numbers are: current flow/ total capacity

An example



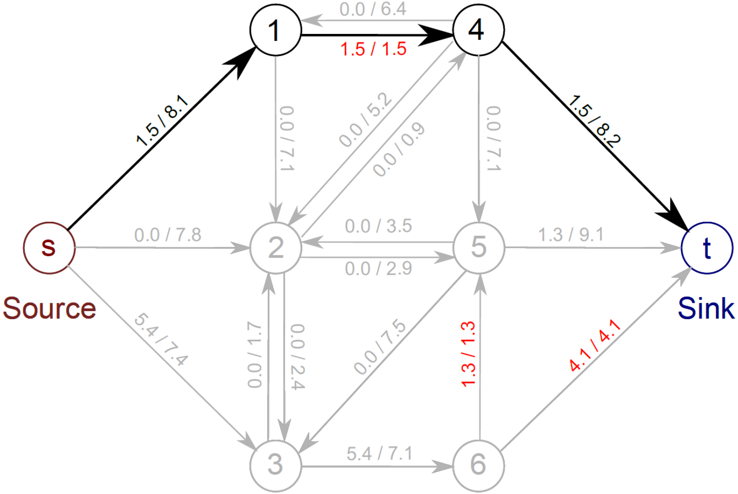
- Chose any path from source to sink with spare capacity and push as much flow as possible.

An example

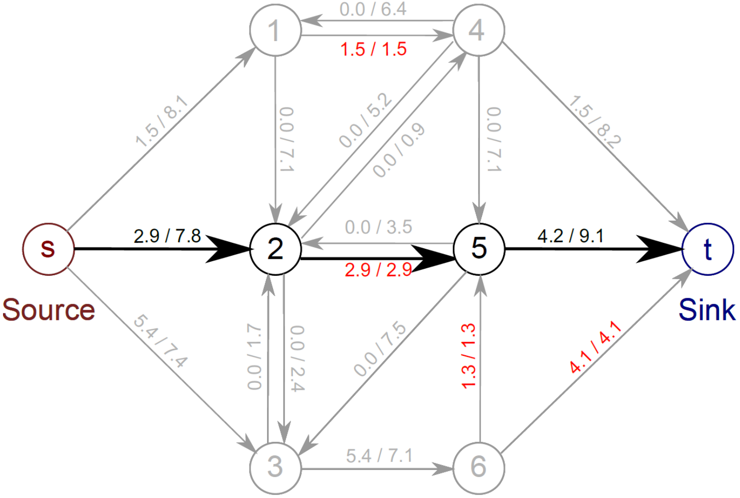


1
4
0.0 / 6.4
0.0 / 1.5
0.0 / 8.1
0.0 / 7.1
0.0 / 5.2
0.0 / 0.9
0.0 / 7.1
0.0 / 8.2
s
2
5
t
0.0 / 7.8
0.0 / 3.5
0.0 / 2.9
1.3 / 9.1
Source
Sink
5.4 / 7.4
0.0 / 1.7
0.0 / 2.4
0.0 / 7.5
1.3 / 1.3
4.1 / 4.1
3
6
5.4 / 7.1

An example

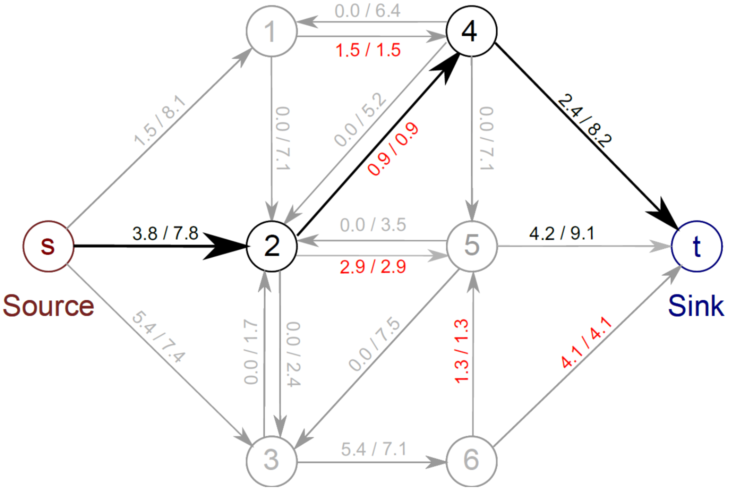

1
4
s
2
5
t
3
6
Source
Sink
0.0 / 6.4
1.5 / 1.5
1.5 / 8.1
1.5 / 8.2
0.0 / 7.1
0.0 / 5.2
0.0 / 0.9
0.0 / 7.1
0.0 / 7.8
0.0 / 3.5
0.0 / 2.9
1.3 / 9.1
5.4 / 7.4
0.0 / 1.7
0.0 / 2.4
0.0 / 7.5
1.3 / 1.3
4.1 / 4.1
5.4 / 7.1

An example



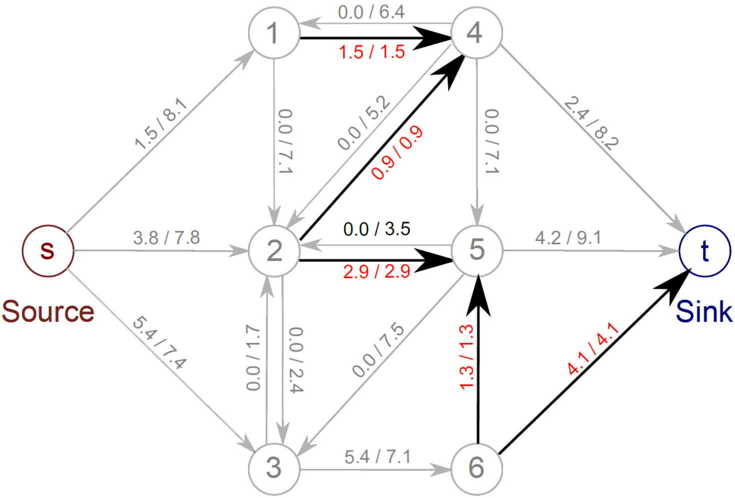
0.0 / 6.4
1.5 / 1.5
1.5 / 8.1
0.0 / 7.1
0.0 / 5.2
0.0 / 0.9
0.0 / 7.1
1.5 / 8.2
s
2.9 / 7.8
2
0.0 / 3.5
2.9 / 2.9
5
4.2 / 9.1
t
1
4
Source
Sink
5.4 / 7.4
0.0 / 1.7
0.0 / 2.4
0.0 / 7.5
1.3 / 1.3
4.1 / 4.1
3
5.4 / 7.1
6

An example



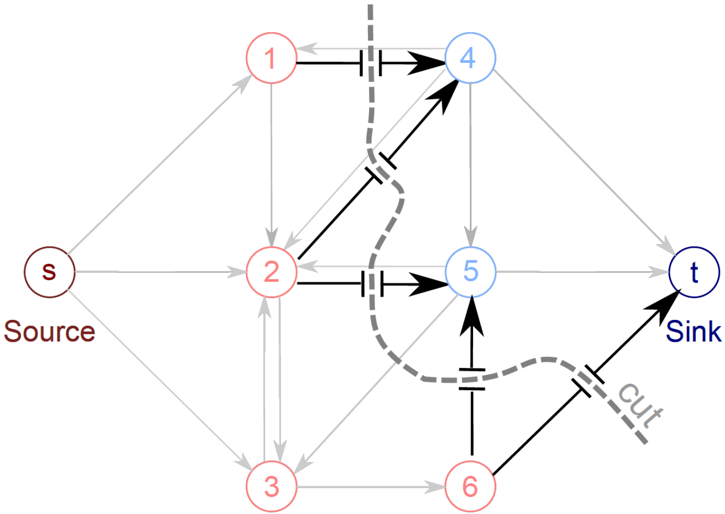
0.0 / 6.4
1
4
1.5 / 1.5
1.5 / 8.1
0.0 / 7.1
0.0 / 5.2
0.9 / 0.9
0.0 / 7.1
2.4 / 8.2
s
3.8 / 7.8
2
0.0 / 3.5
5
4.2 / 9.1
t
2.9 / 2.9
Source
Sink
5.4 / 7.4
0.0 / 1.7
0.0 / 2.4
0.0 / 7.5
1.3 / 1.3
4.1 / 4.1
3
5.4 / 7.1
6

An example



- No further ‘augmenting path’ exists.

An example



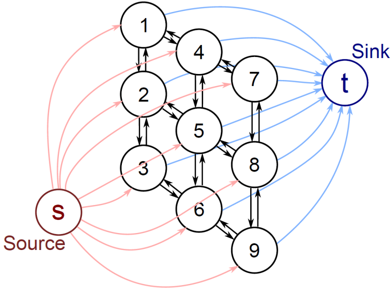
- The saturated edges partition the graph into two subgraphs.

The binary MRFs

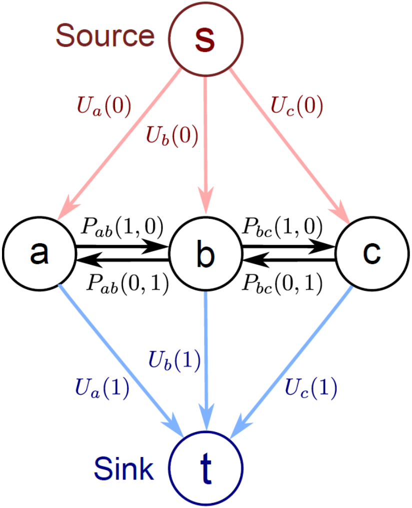
- In the simplest form, let us constrain the pairwise potentials for adjacent nodes m, n to be:
  - $\phi_{m,n}(0,0) = \phi_{m,n}(1,1) = 0$.
  - $\phi_{m,n}(1,0) = \theta_{10}$.
  - $\phi_{m,n}(0,1) = \theta_{01}$.

The binary MRFs

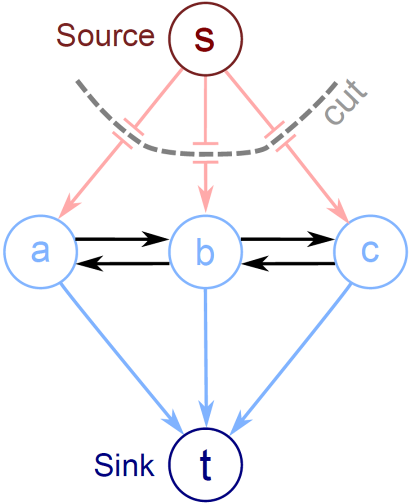
- In the simplest form, let us constrain the pairwise potentials for adjacent nodes m, n to be:
  - $\phi_{m,n}(0,0) = \phi_{m,n}(1,1) = 0.$
  - $\phi_{m,n}(1,0) = \theta_{10}.$
  - $\phi_{m,n}(0,1) = \theta_{01}.$
- Will make a graph such that each cut corresponds to a configuration.



The binary MRFs



The binary MRFs



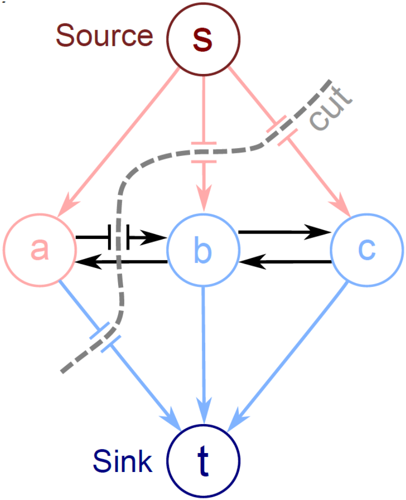
Solution

0	0	0

Cost

$$U_a(0) + U_b(0) + U_c(0)$$

The binary MRFs



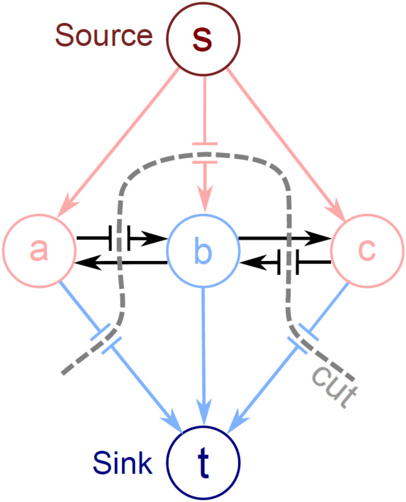
Solution

1	0	0

Cost

$$U_a(1) + U_b(0) + U_c(0)$$
$$+P_{bc}(1,0)$$

The binary MRFs



Solution

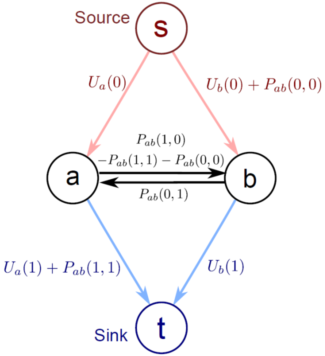
1	0	1

Cost

$$U_a(0) + U_b(0) + U_c(0)$$
$$+P_{ab}(1,0) + P_{bc}(0,1)$$

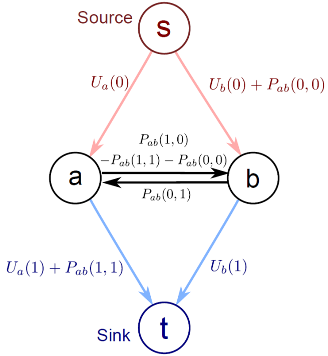
The binary MRFs

- In the general case:



The binary MRFs

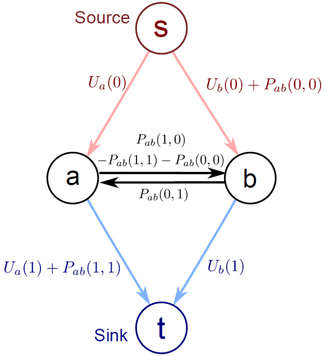
- In the general case:



- Constraint $\theta_{10} + \theta_{01} > \theta_{11} + \theta_{00}$ (attraction).

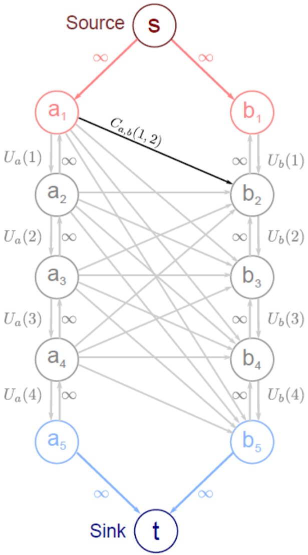
The binary MRFs

- In the general case:



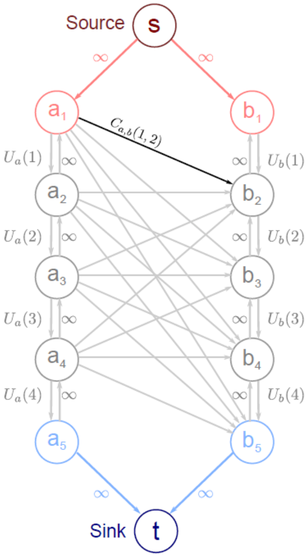
- Constraint $\theta_{10} + \theta_{01} > \theta_{11} + \theta_{00}$ (attraction).
- If met, the problem is called "submodular" and we can solve it in polynomial time.

Other cases



Source
s
∞
∞
a_1
b_1
$C_{a,b}(1,2)$
$U_a(1)$
∞
∞
$U_b(1)$
a_2
b_2
$U_a(2)$
∞
∞
$U_b(2)$
a_3
b_3
$U_a(3)$
∞
∞
$U_b(3)$
a_4
b_4
$U_a(4)$
∞
∞
$U_b(4)$
a_5
b_5
∞
∞
Sink
t

Other cases



$$P_{ab}(\beta,\gamma) + P_{ab}(\alpha,\delta) - P_{a,b}(\beta,\delta) - P_{ab}(\alpha,\gamma) \geq 0,$$

Other cases

- Another type of constraint allows approximate solutions.

Other cases

- Another type of constraint allows approximate solutions.
  - if the pairwise potential is a metric

$$
\begin{aligned}
P(\alpha,\beta) = & \quad 0 \quad \Leftrightarrow \alpha = \beta \\
P(\alpha,\beta) & = P(\beta,\alpha) > 0 \\
P(\alpha,\beta) & \leq P(\alpha,\gamma) + P(\gamma,\beta)
\end{aligned}
$$

Other cases

- Another type of constraint allows approximate solutions.
  - if the pairwise potential is a metric

$$\begin{aligned} P(\alpha,\beta) = \quad 0 &\Leftrightarrow \alpha = \beta \\ P(\alpha,\beta) &= P(\beta,\alpha) > 0 \\ P(\alpha,\beta) &\leq P(\alpha,\gamma) + P(\gamma,\beta) \end{aligned}$$

- Alpha Expansion Algorithm (next week) uses the max-flow idea as a subroutine to do coordinate descent in the label space.

Conclusion

- Decoding and inference is still efficient with Semi-Markov models.

Conclusion

- Decoding and inference is still efficient with Semi-Markov models.
  - Useful if need to control the length of each state over a sequence.

Conclusion

- Decoding and inference is still efficient with Semi-Markov models.
  - Useful if need to control the length of each state over a sequence.
- Graph cuts help with decoding on models with pairwise potentials.

Conclusion

- Decoding and inference is still efficient with Semi-Markov models.
  - Useful if need to control the length of each state over a sequence.
- Graph cuts help with decoding on models with pairwise potentials.
  - Exact solution in binary case if submodular.

Conclusion

- Decoding and inference is still efficient with Semi-Markov models.
    - Useful if need to control the length of each state over a sequence.
- Graph cuts help with decoding on models with pairwise potentials.
    - Exact solution in binary case if submodular.
    - Exact solution in multi-label case if submodular.

Conclusion

- Decoding and inference is still efficient with Semi-Markov models.
  - Useful if need to control the length of each state over a sequence.
- Graph cuts help with decoding on models with pairwise potentials.
  - Exact solution in binary case if submodular.
  - Exact solution in multi-label case if submodular.
  - Approximate solution in multi-label case if a metric.

Thank you!

Questions?