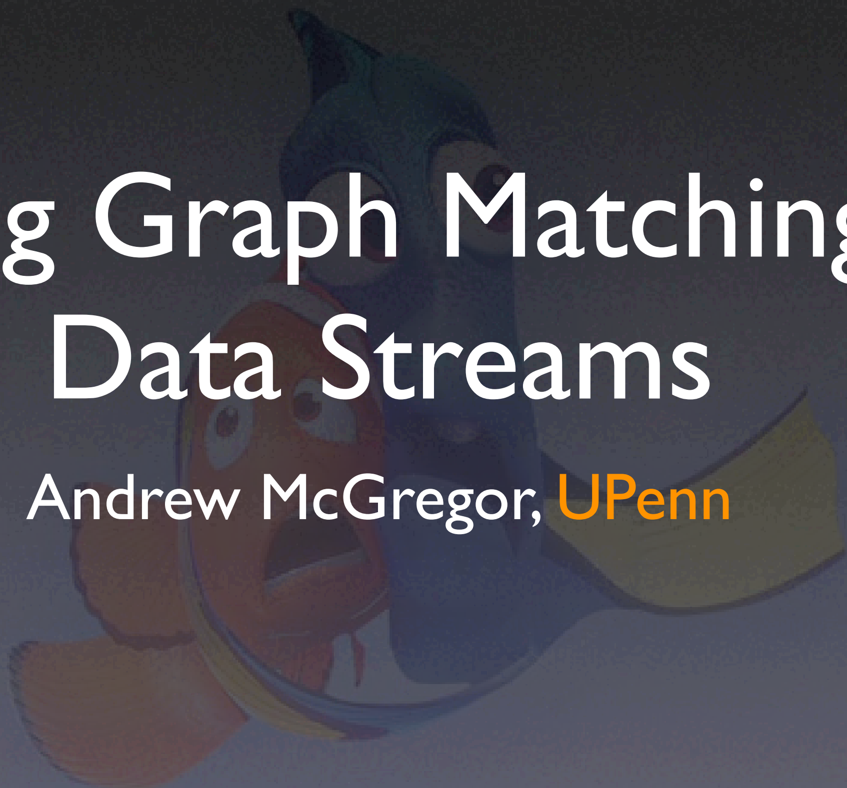


Finding Graph Matchings in Data Streams

Andrew McGregor, UPenn

The Streaming Model

The Streaming Model

- Classic Problem: Median Finding [Munro & Paterson]

The Streaming Model

- Classic Problem: Median Finding [Munro & Paterson]
- Parameters of the Model:
  - How much memory?
  - How many passes?
  - How much computation time between data elements?

The Streaming Model

- Classic Problem: Median Finding [Munro & Paterson]
- Parameters of the Model:
  - How much memory?
  - How many passes?
  - How much computation time between data elements?
- Statistics, Norms and Histograms…

The Streaming Model

- Classic Problem: Median Finding [Munro & Paterson]
- Parameters of the Model:
  - How much memory?
  - How many passes?
  - How much computation time between data elements?
- Statistics, Norms and Histograms…
- What about graph problems?

Graph Streaming

- Instance of graph problem $G = (V, E)$
- Edges arrive in arbitrary order: $e_1, e_2, e_3, \ldots, e_m$
- Memory limit $O(n \text{ polylog } n)$ where $n = |V|$

- Spanner Construction, Bipartite Matching, Lower Bounds [Feigenbaum, Kannan, M. , Suri, Zhang '04 &'05]
- "Annotation" Stream Model [Aggarwal, Datar, Rajagopalan, Ruhl '04, Demetrescu, Finocchi, Ribichini '05]

Matching

- A matching - set of edges with no two edges sharing an end point.
- Problems:
  - Find the matching of maximum cardinality (MCM)
  - Find the matching of maximum weight (MWM)
- (Non-streamable) Algorithms:
  - Exact polytime algorithm for both [Gabow '90]
  - Linear-time $1+\epsilon$ approx for MCM [Kalantari & Shokoufandeh '95]
  - Linear-time $3/2+\epsilon$ approx for MWM [Drake & Hougardy '03]

Results

- Unweighted Matchings:

  $1+\epsilon$ approximation in constant passes.

- Weighted Matchings:

  $3+2\sqrt{2}$ approximation in single pass.

  $2+\epsilon$ approximation in constant passes.

Unweighted Matchings.

An Easy 2 Approximation

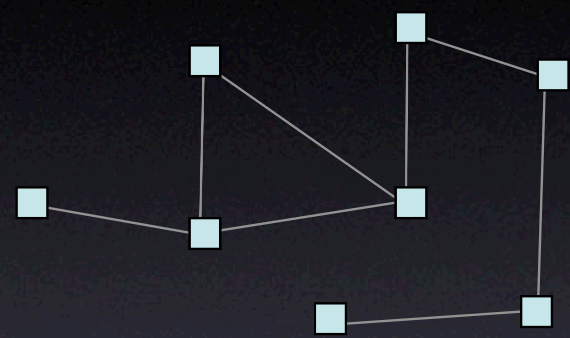
- Greedy Algorithm:

  Store an edge if it is not adjacent to stored edge

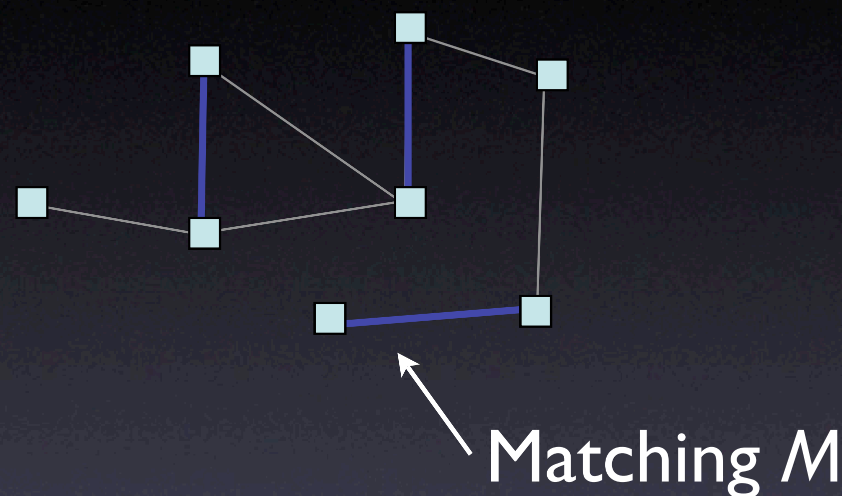
- Construct a maximal matching - 2 Approximation

Augmenting Paths

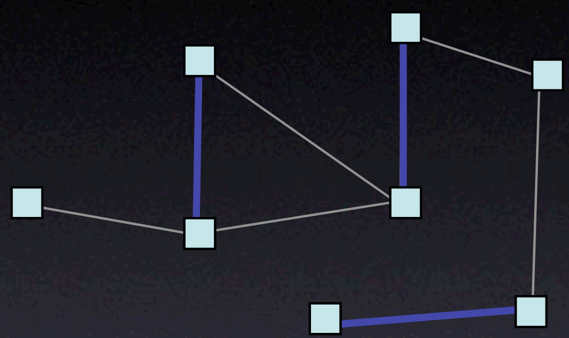
Augmenting Paths



Augmenting Paths

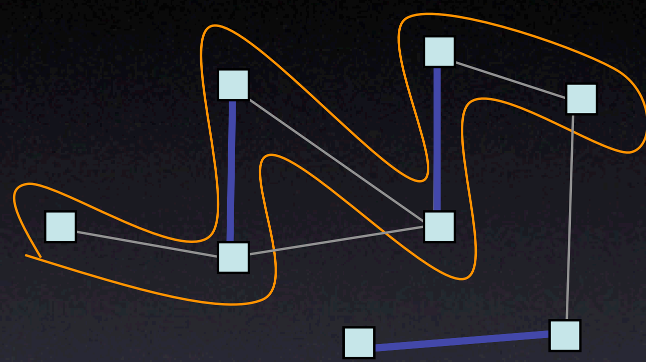


Augmenting Paths



- Augmenting Path: simple path starting and ending at unmatched nodes such that edges alternate between M and $E \setminus M$.

Augmenting Paths



- Augmenting Path: simple path starting and ending at unmatched nodes such that edges alternate between M and $E \setminus M$.

Augmenting Paths

- Augmenting Path: simple path starting and ending at unmatched nodes such that edges alternate between *M* and *E**M*.

Augmenting Paths

- Consider augmenting paths defined by taking the symmetric difference between current (maximal) matching and optimum matching.
- Let P_i be the number of length *i* augmenting paths

$$|M| + \sum_{1 \le i \le k} P_i \ge OPT(1 - 1/k)$$

Algorithm Outline

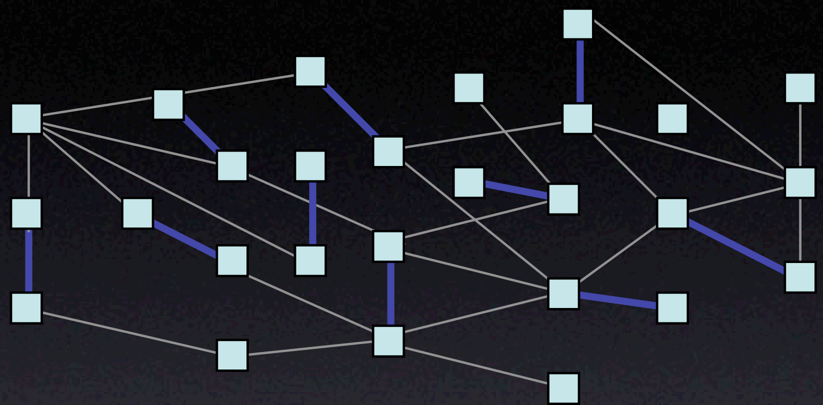
1. Find a maximal matching
2. For $1 \le i \le k$:

   Find a set, S_i, of length i augmenting paths

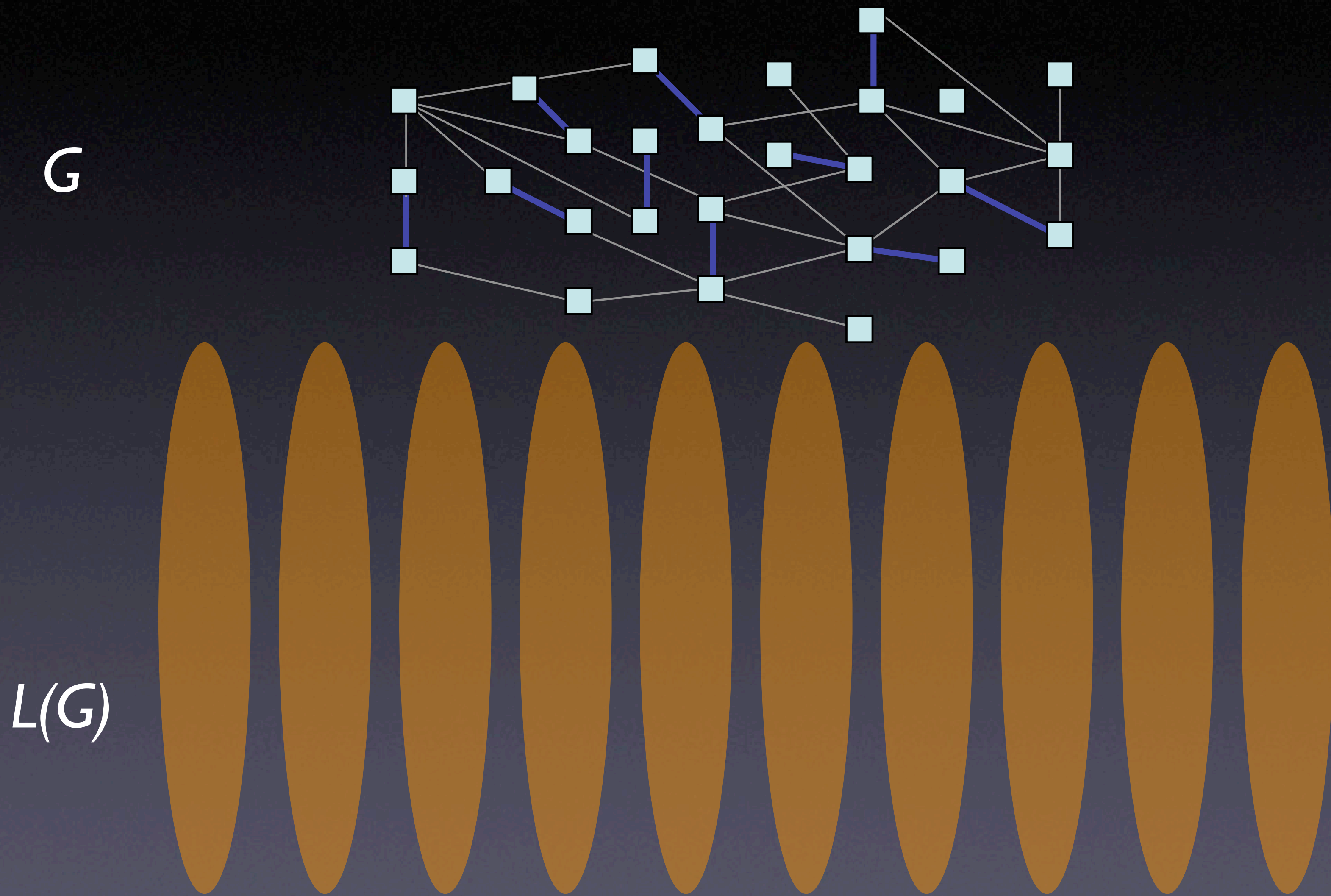
3. Augment current matching with S_j where j = argmax S_i
4. Repeat from 2 unless S_j is small

Projecting to Layered Graphs

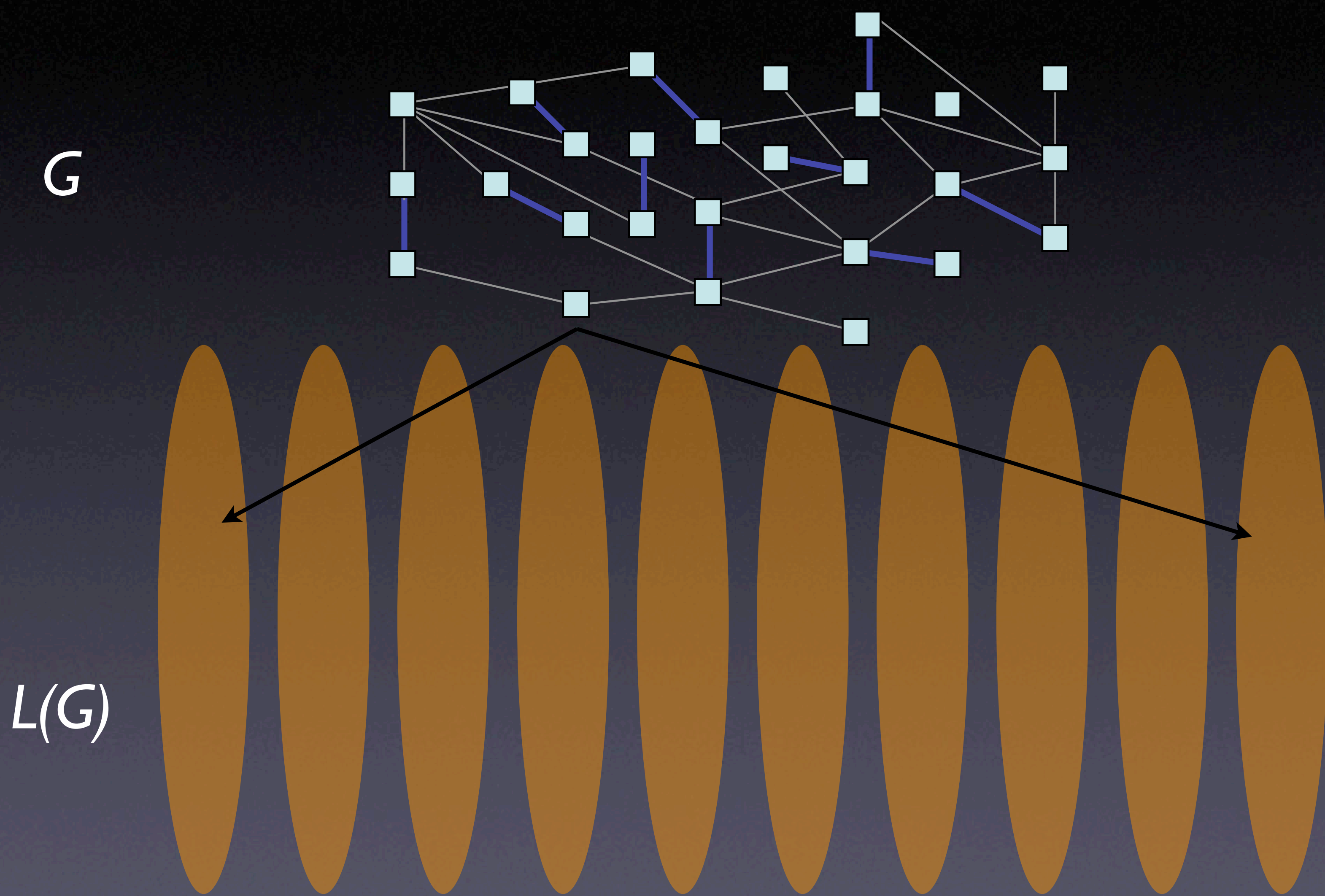
G



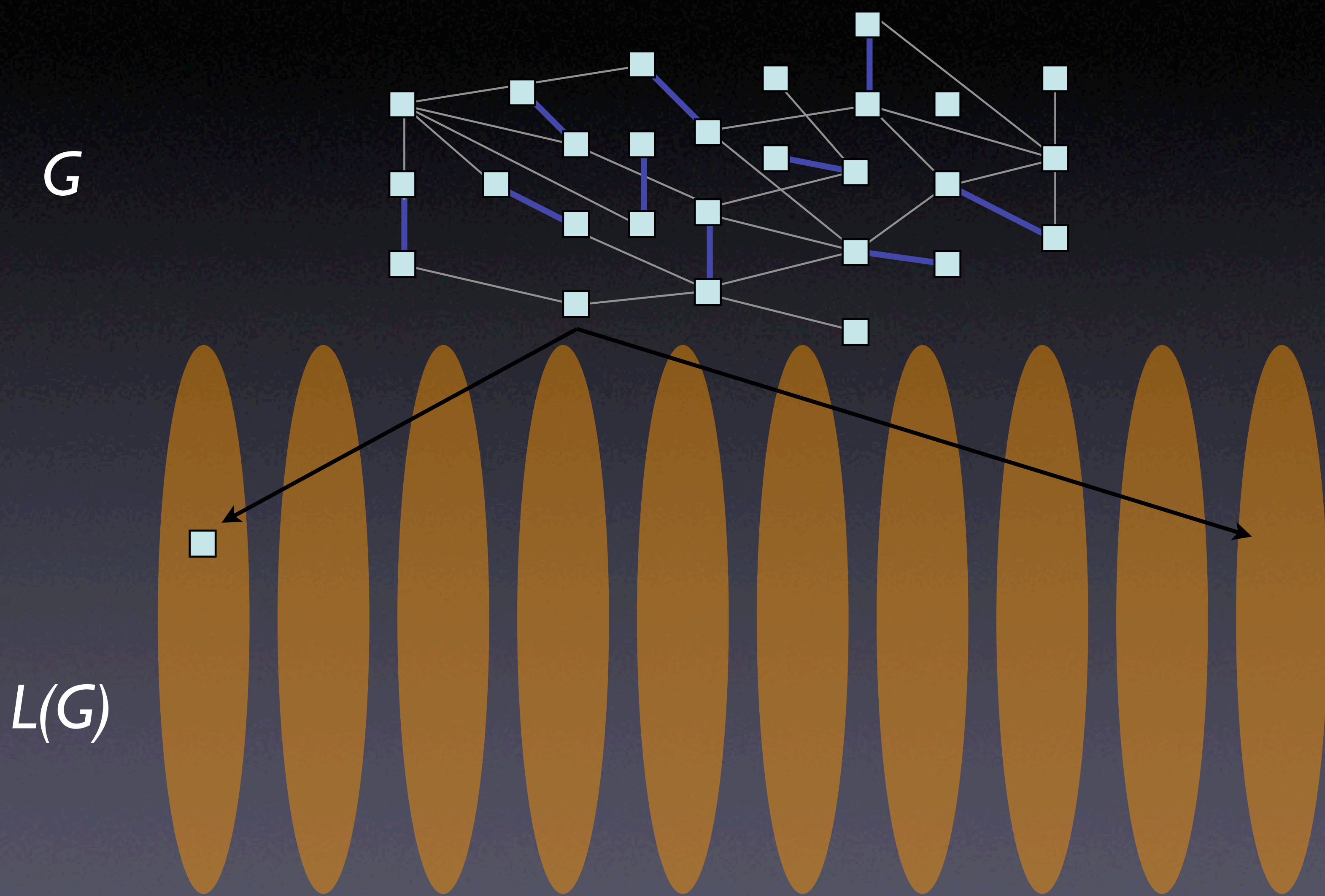
Projecting to Layered Graphs



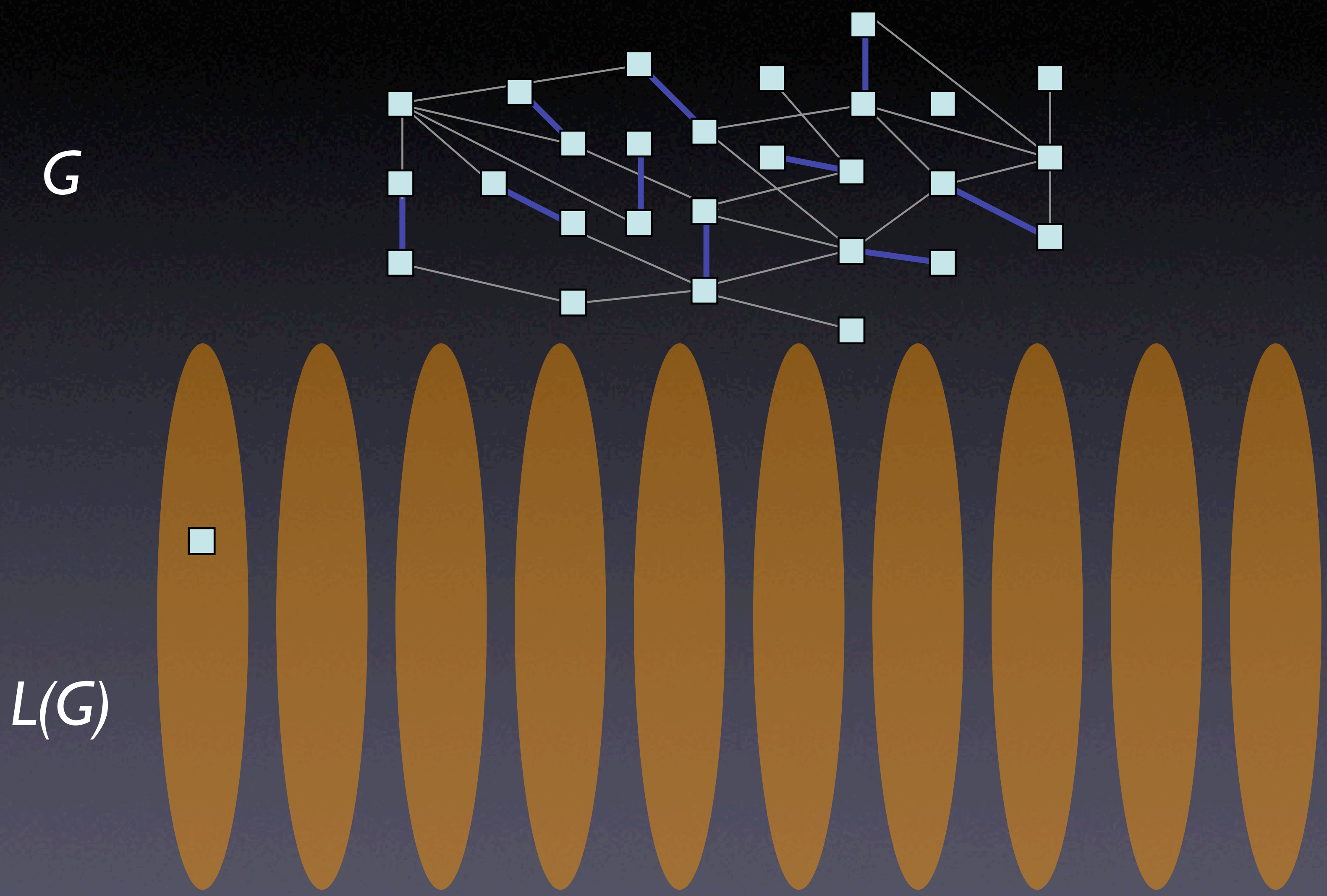
Projecting to Layered Graphs



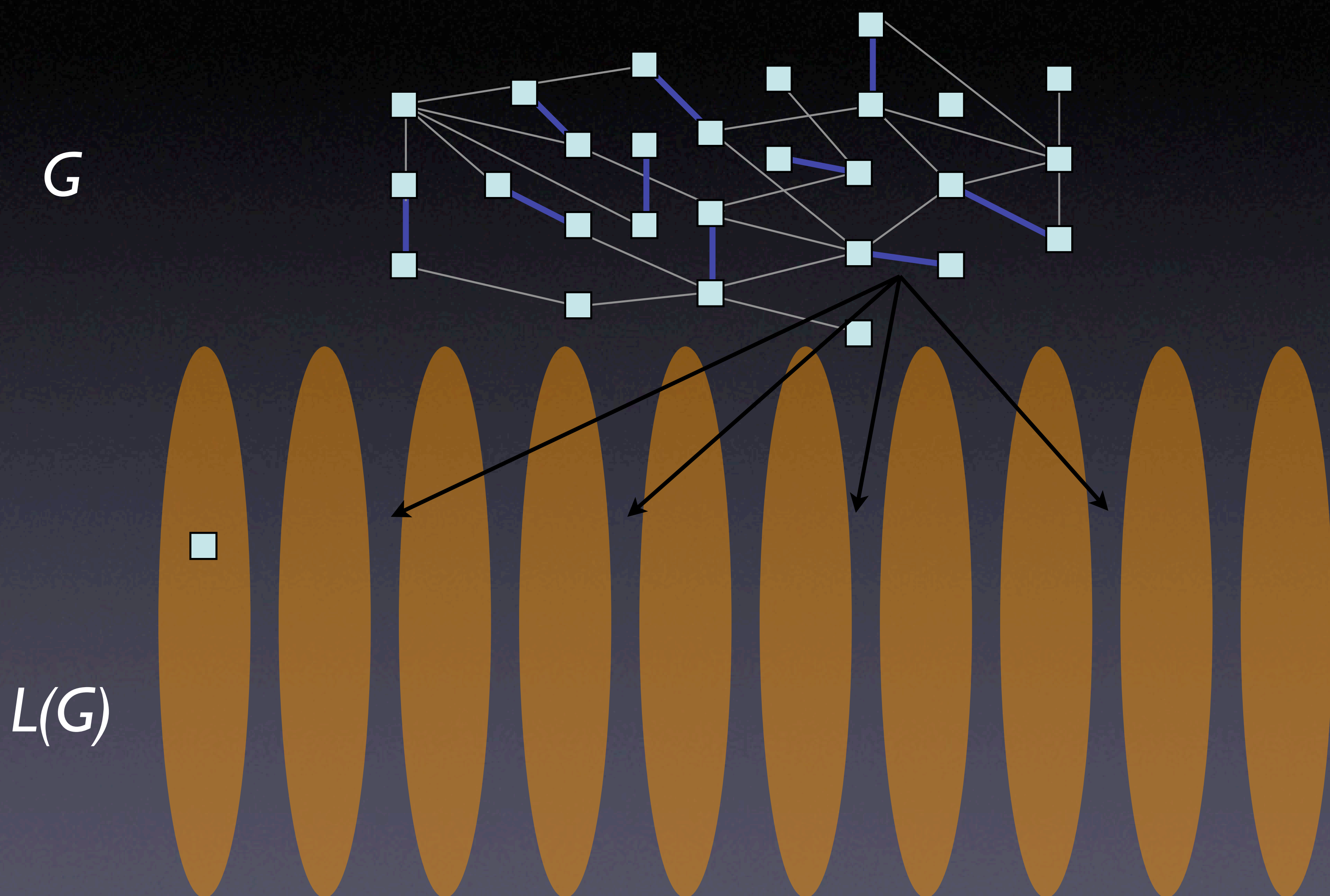
Projecting to Layered Graphs



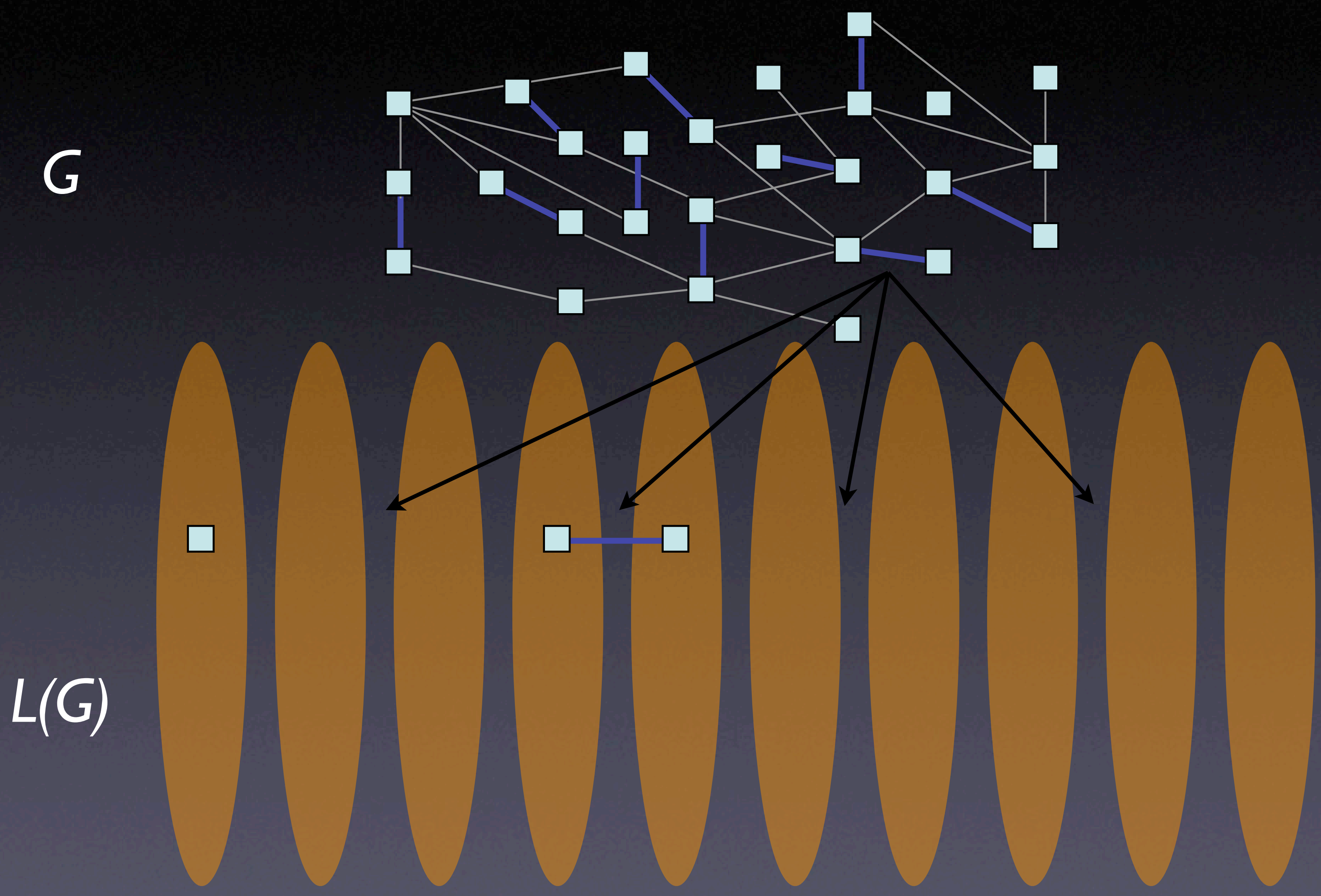
Projecting to Layered Graphs



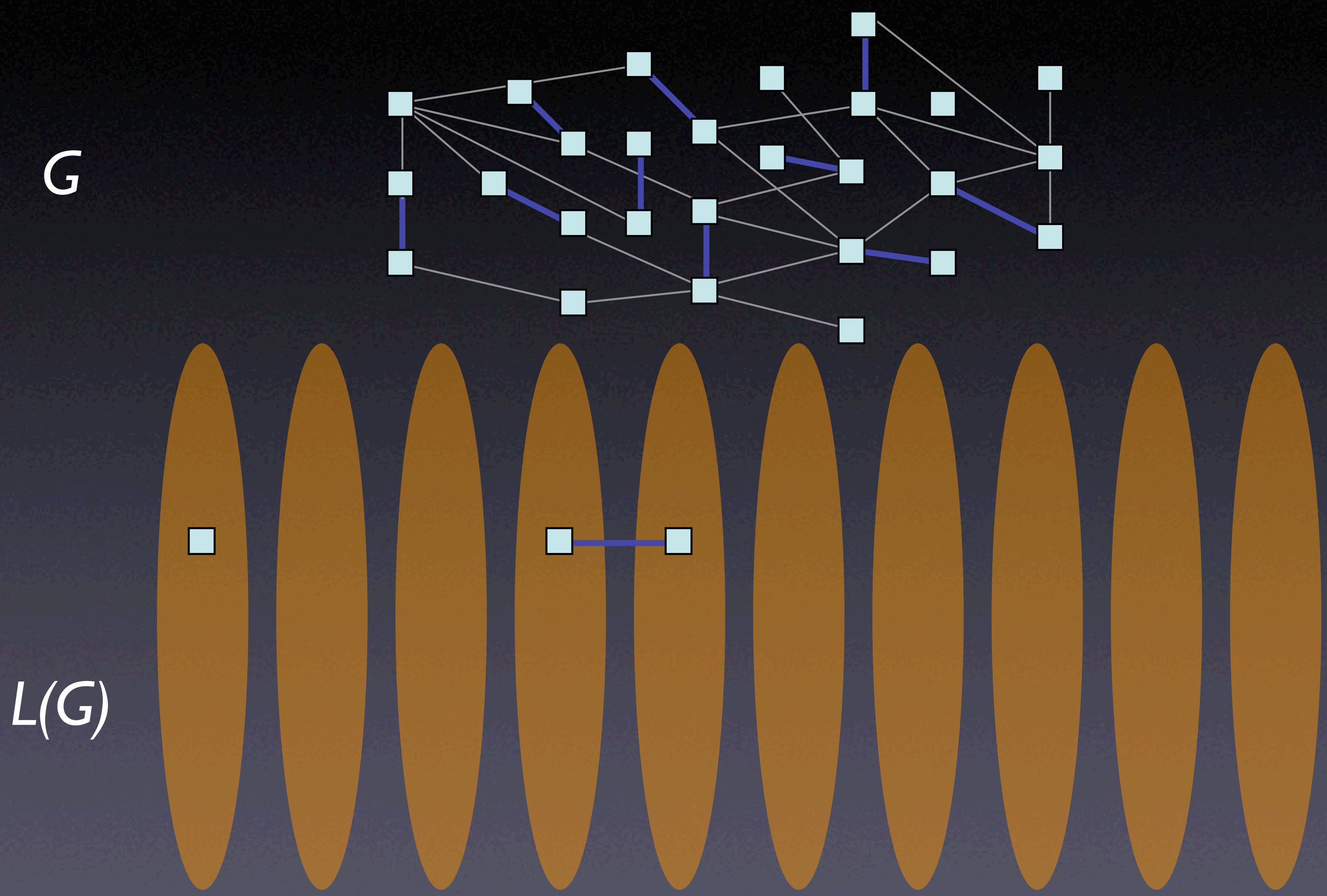
Projecting to Layered Graphs



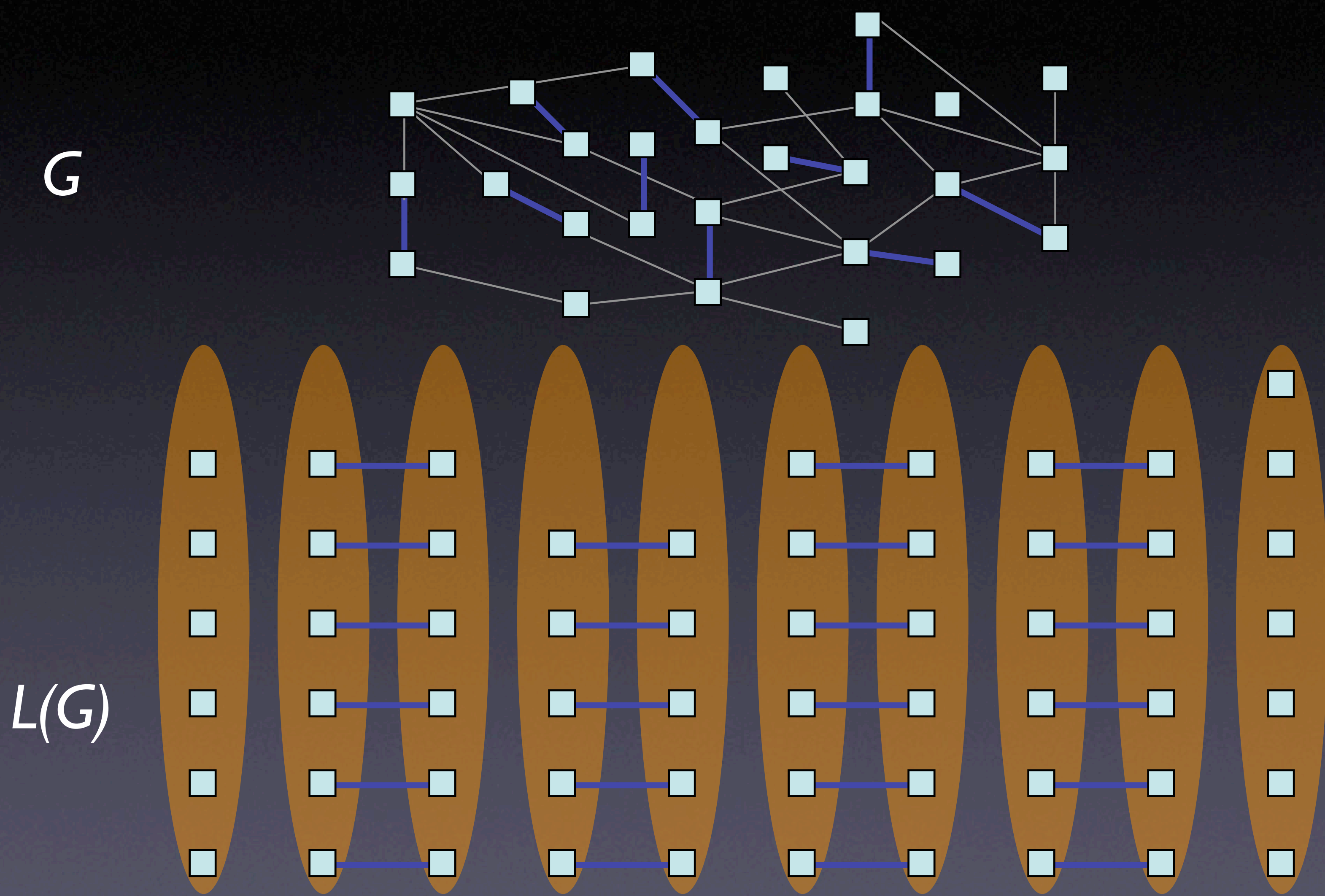
Projecting to Layered Graphs



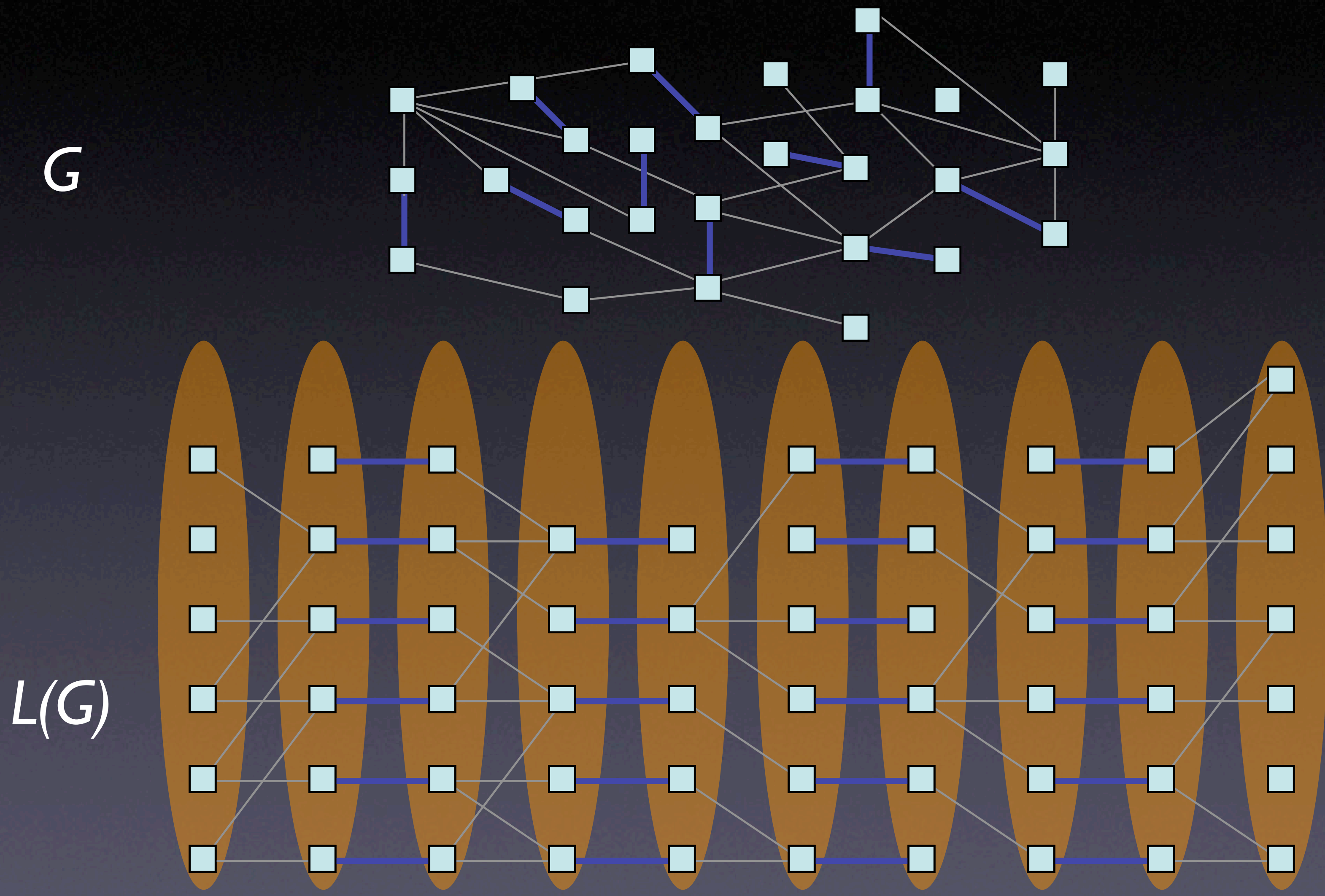
Projecting to Layered Graphs



Projecting to Layered Graphs

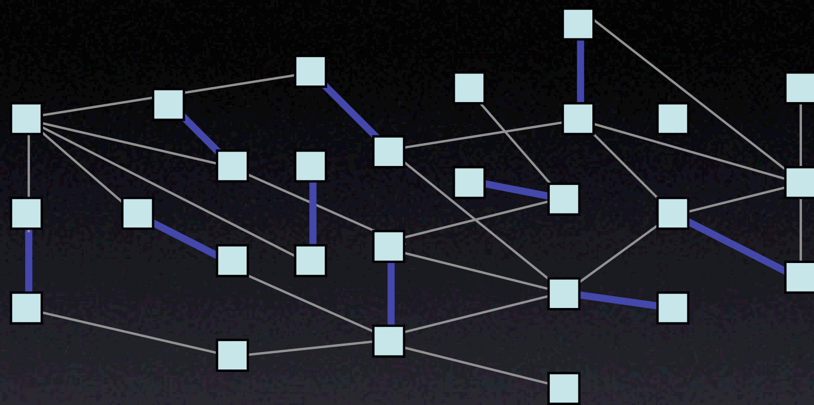


Projecting to Layered Graphs

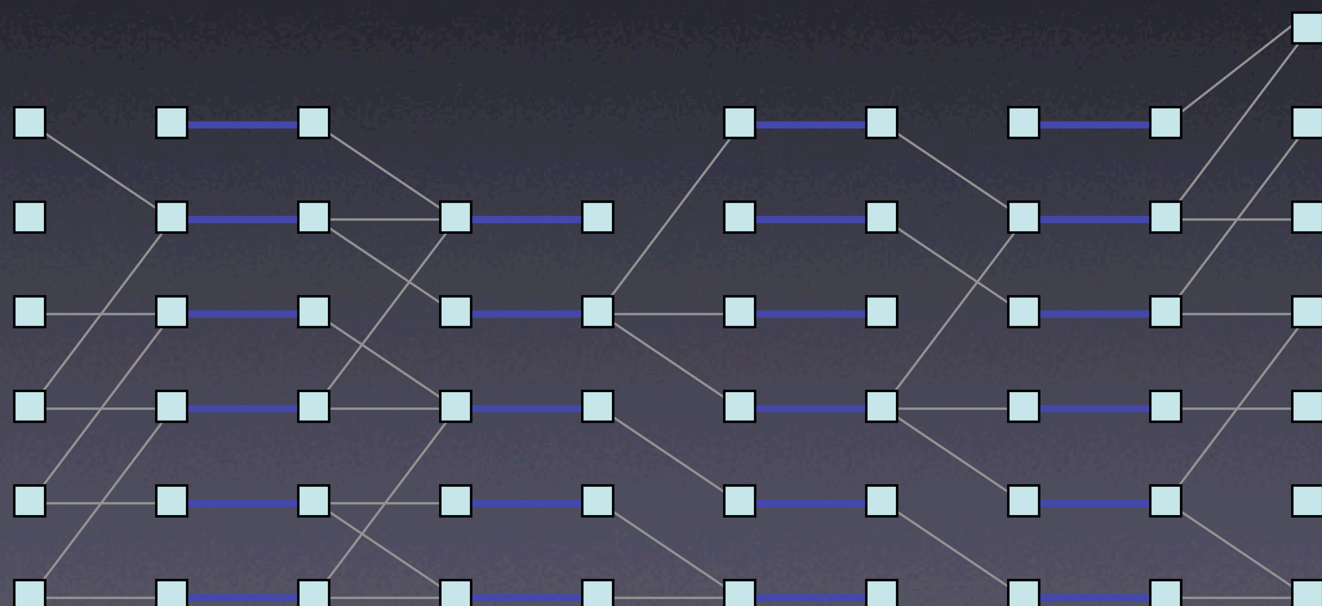


Projecting to Layered Graphs

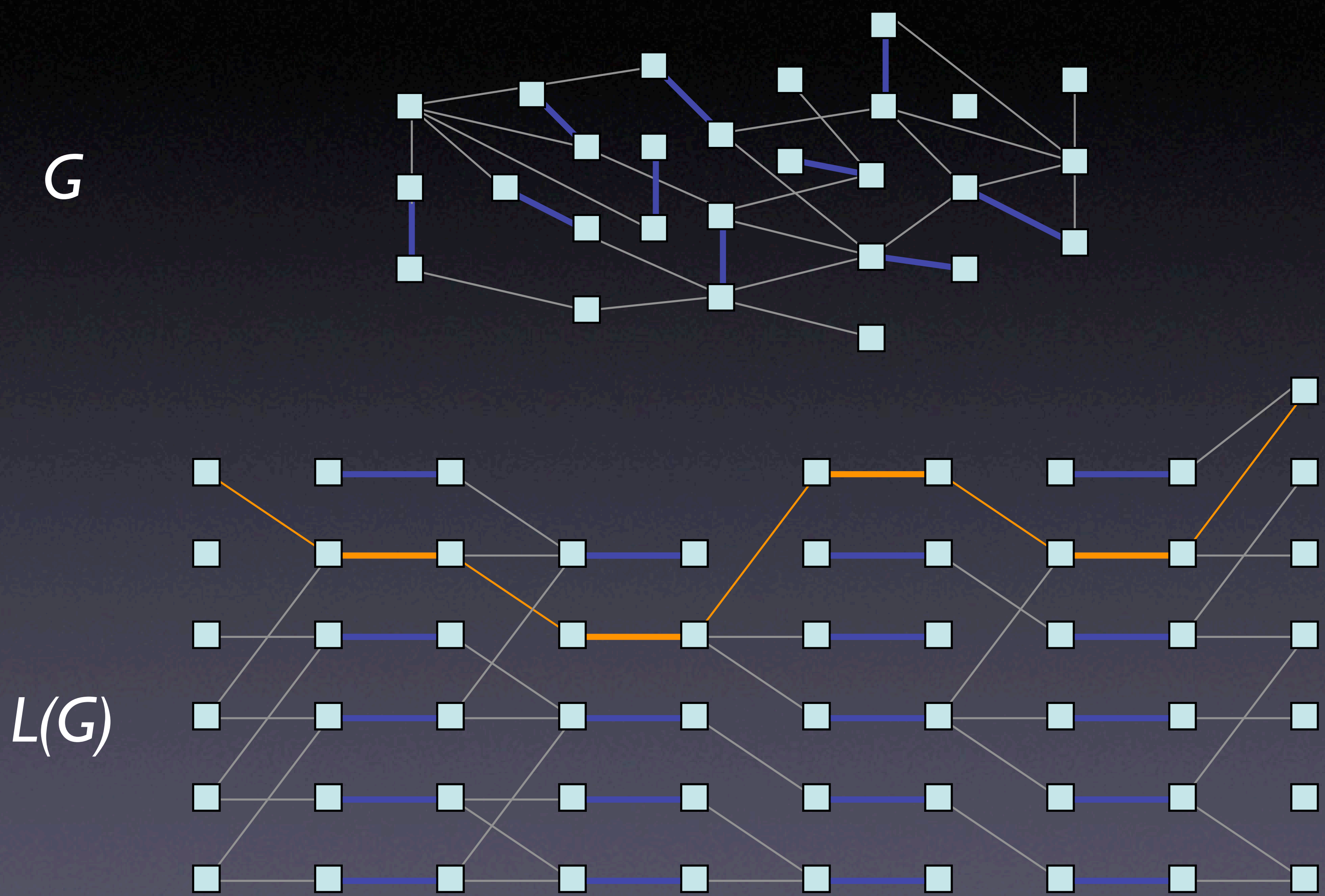
G



L(G)



Projecting to Layered Graphs



Lemma: If there are P_i length i augmenting paths in G then we expect $P_i \, / \, 2(2i)^i$ node disjoint paths in $L(G)$.

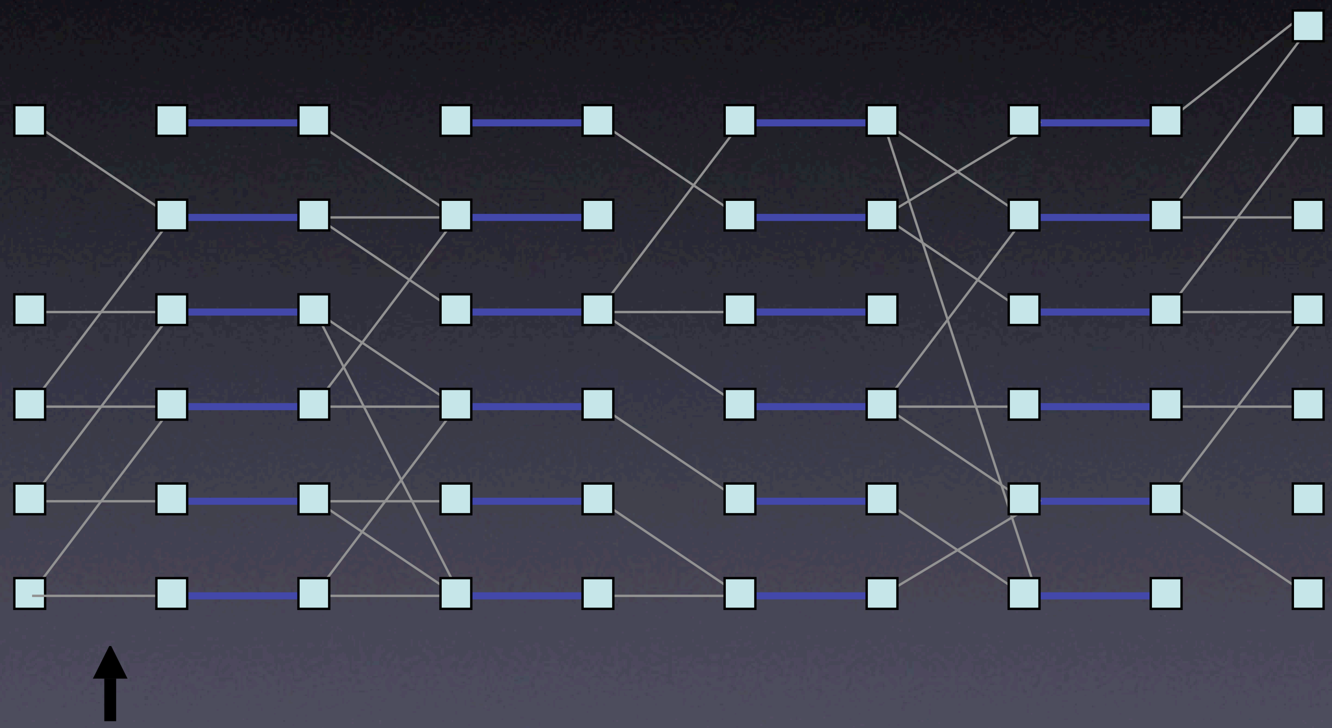
Lemma: If there are P_i length i augmenting paths in G then we expect $P_i / 2(2i)^i$ node disjoint paths in $L(G)$.

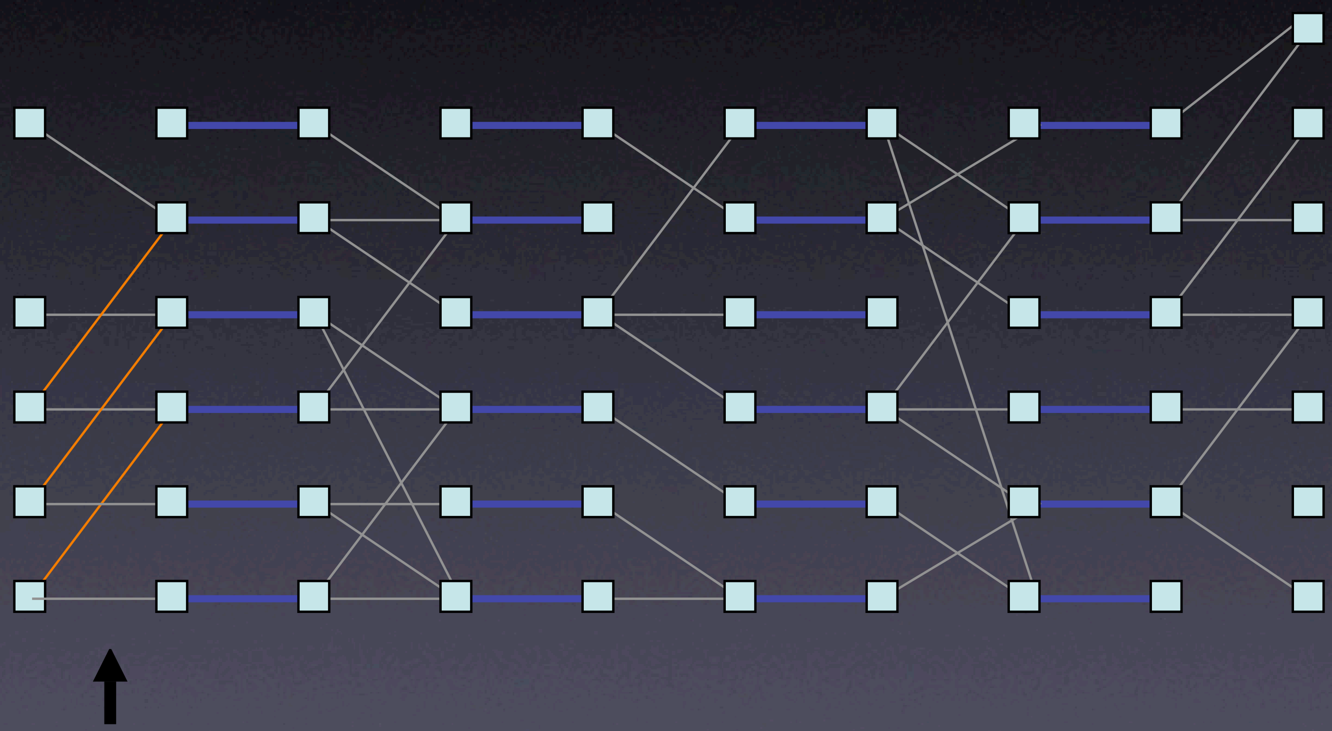
Lemma: A maximal set of node disjoint paths in $L(G)$, is an $i+2$ approximation to the maximum set of node disjoint paths in $L(G)$.

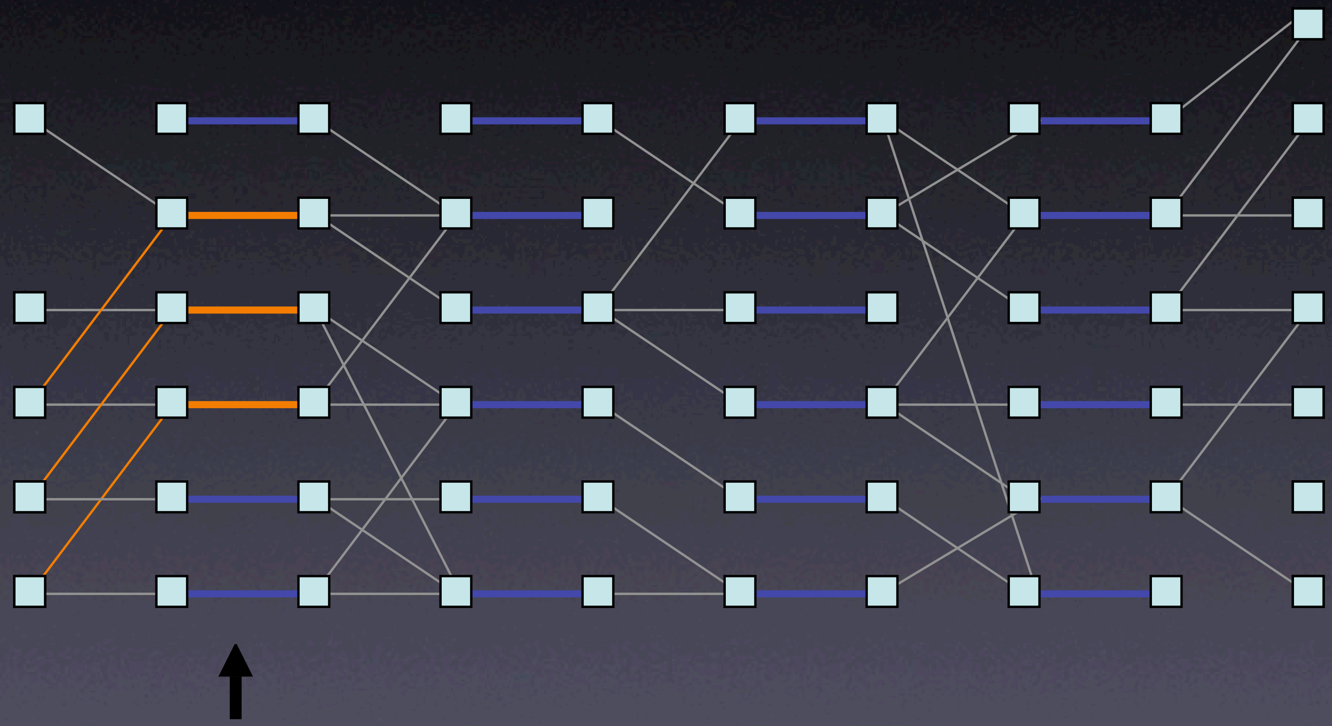
Lemma: If there are P_i length i augmenting paths in G then we expect $P_i / 2(2i)^i$ node disjoint paths in $L(G)$.

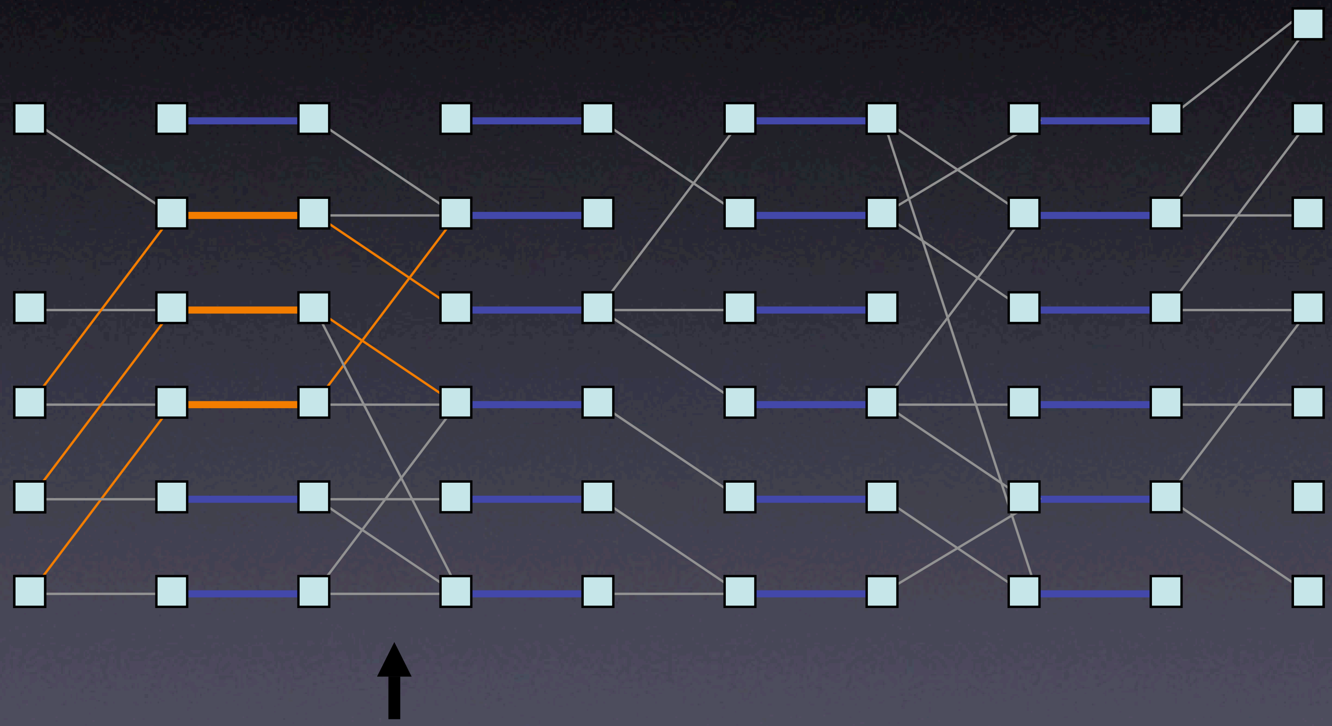
Lemma: A maximal set of node disjoint paths in $L(G)$, is an $i+2$ approximation to the maximum set of node disjoint paths in $L(G)$.

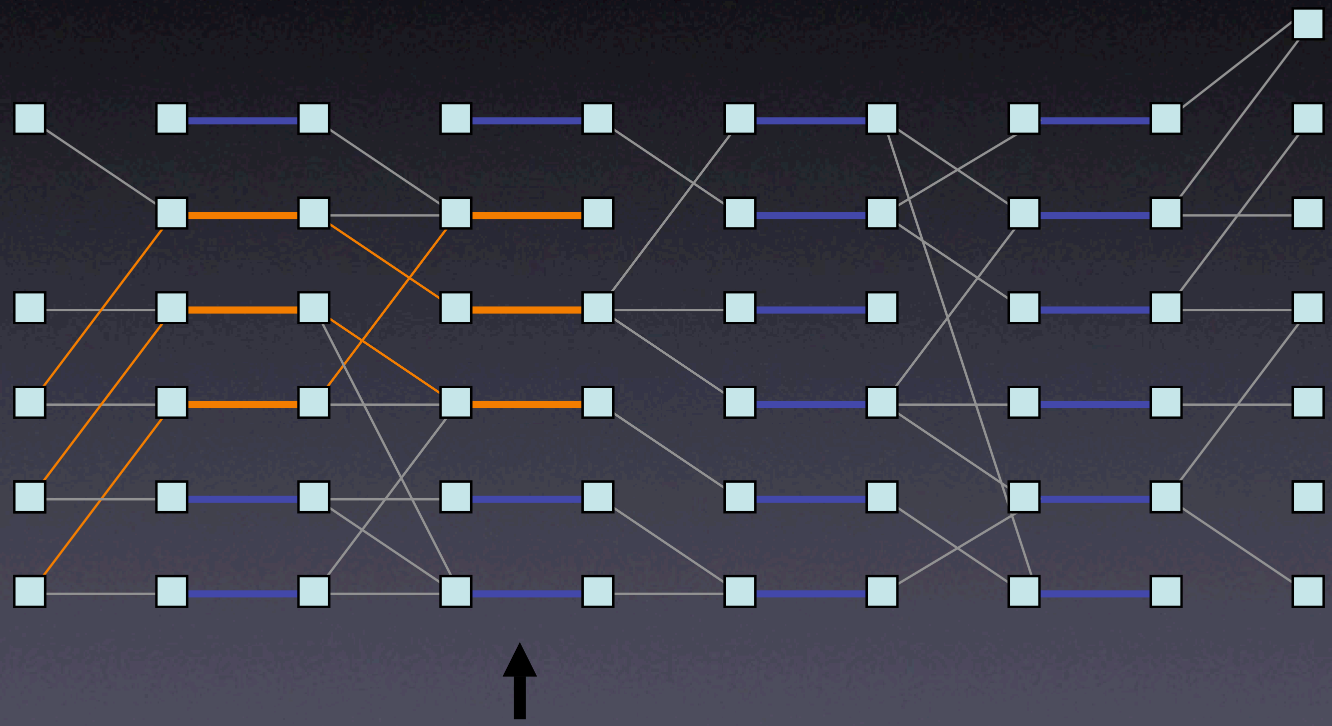
To find a constant fraction of length i augmenting paths P_i, create layered graph and greedily find node disjoint paths.

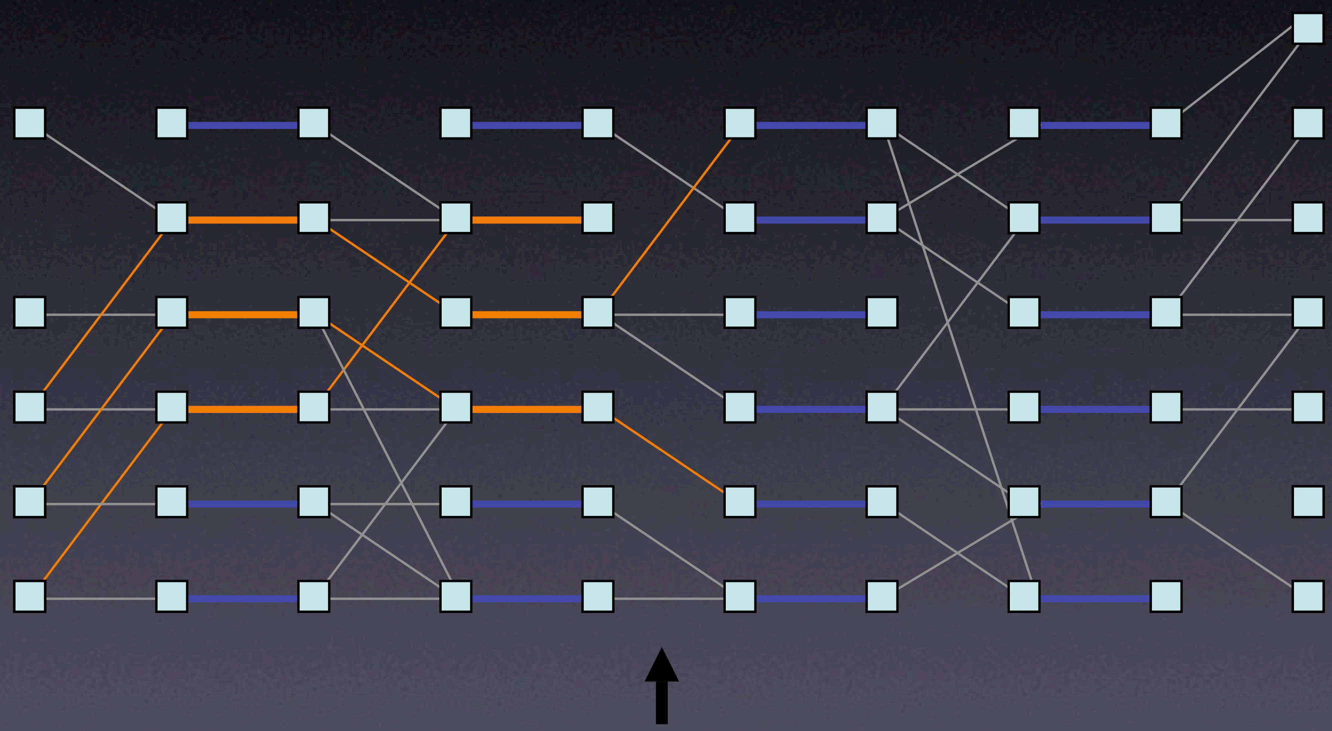


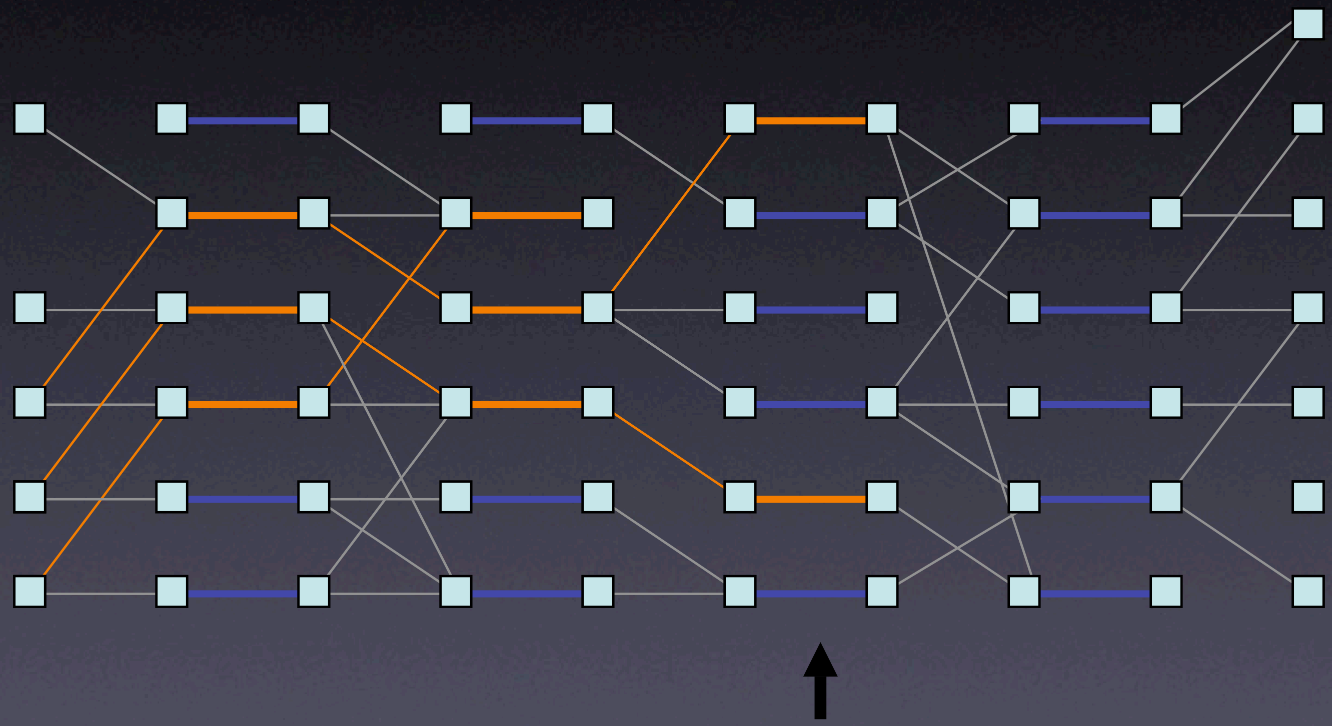


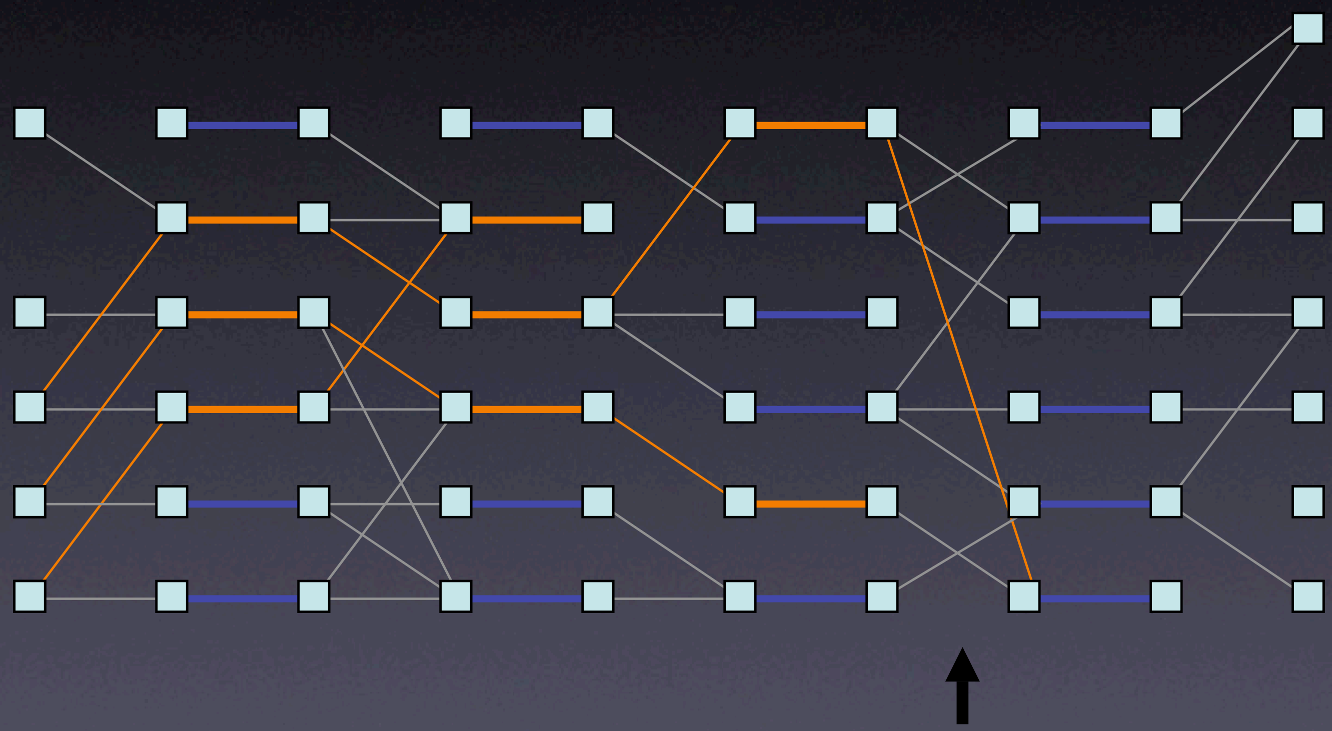


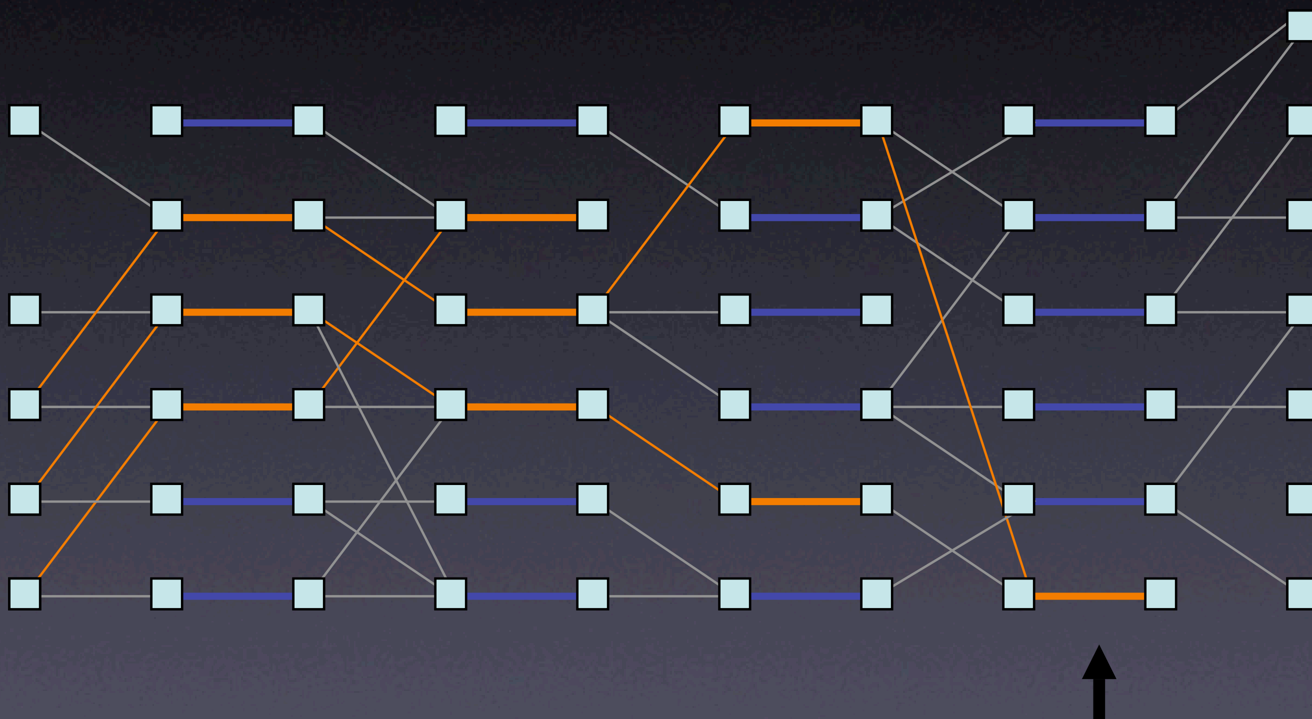


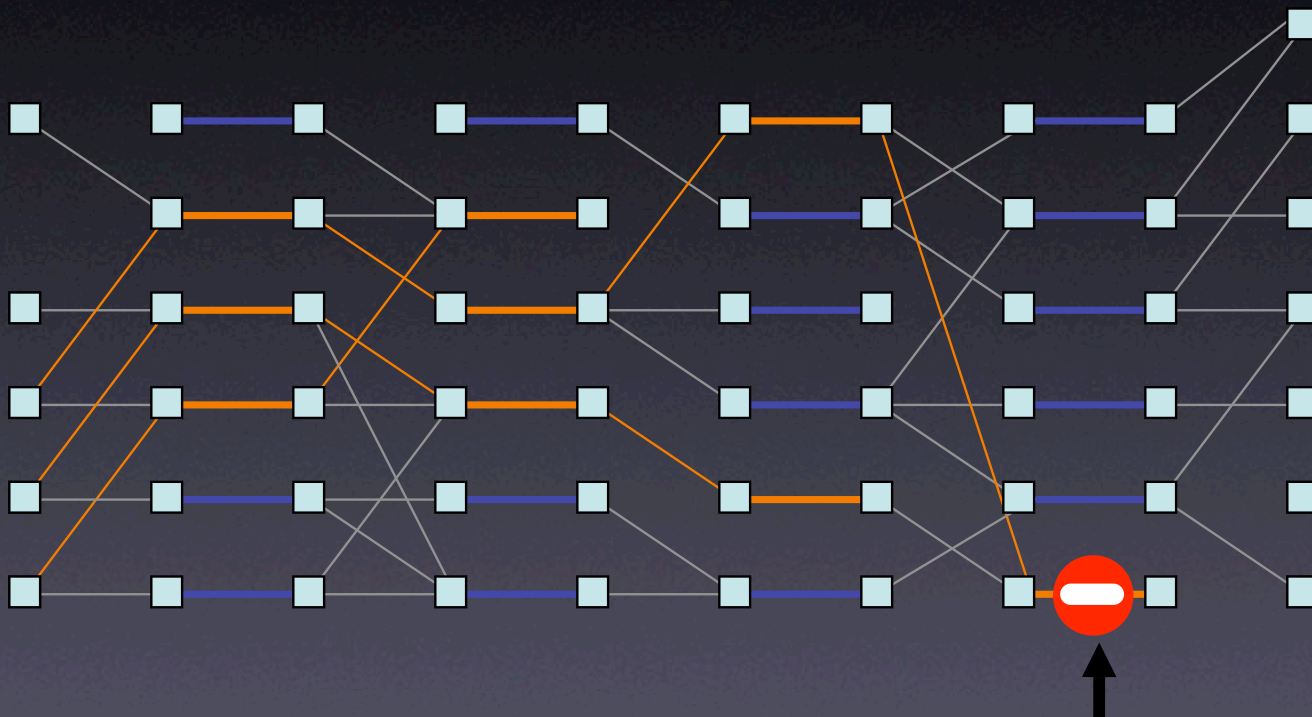


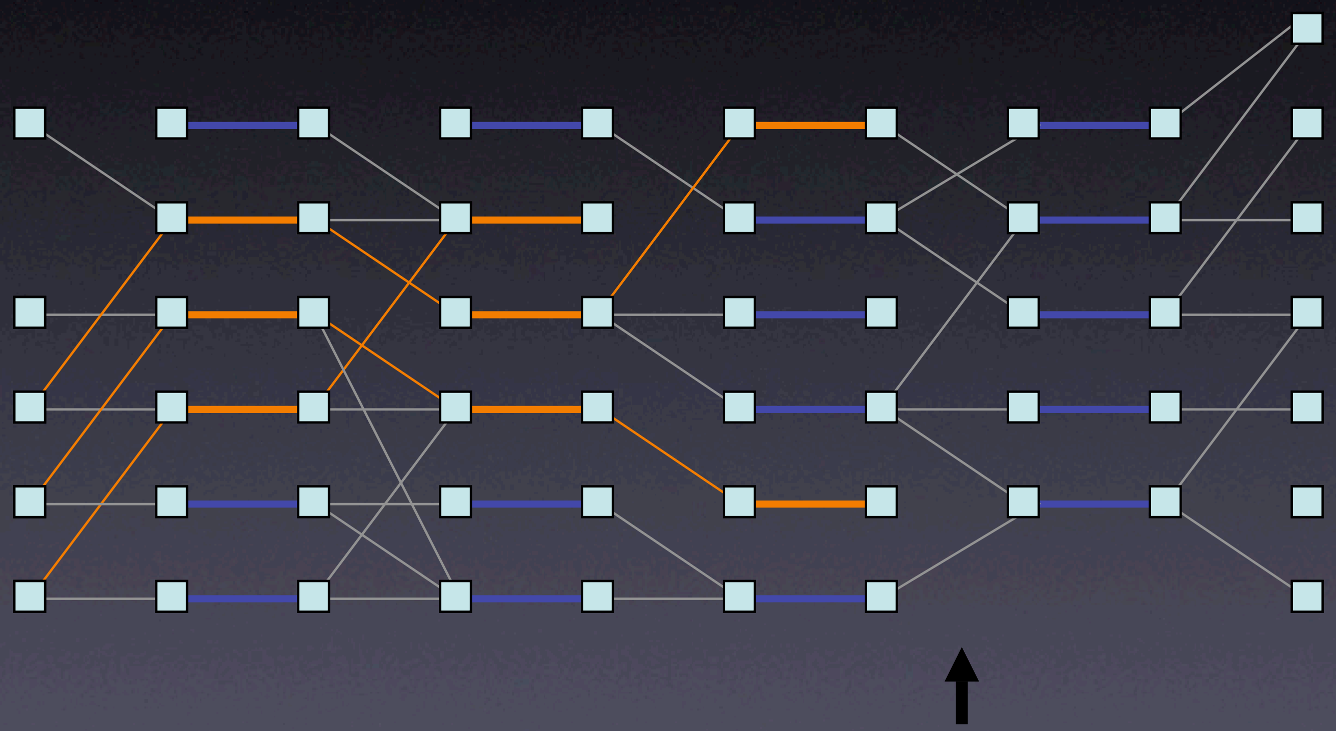


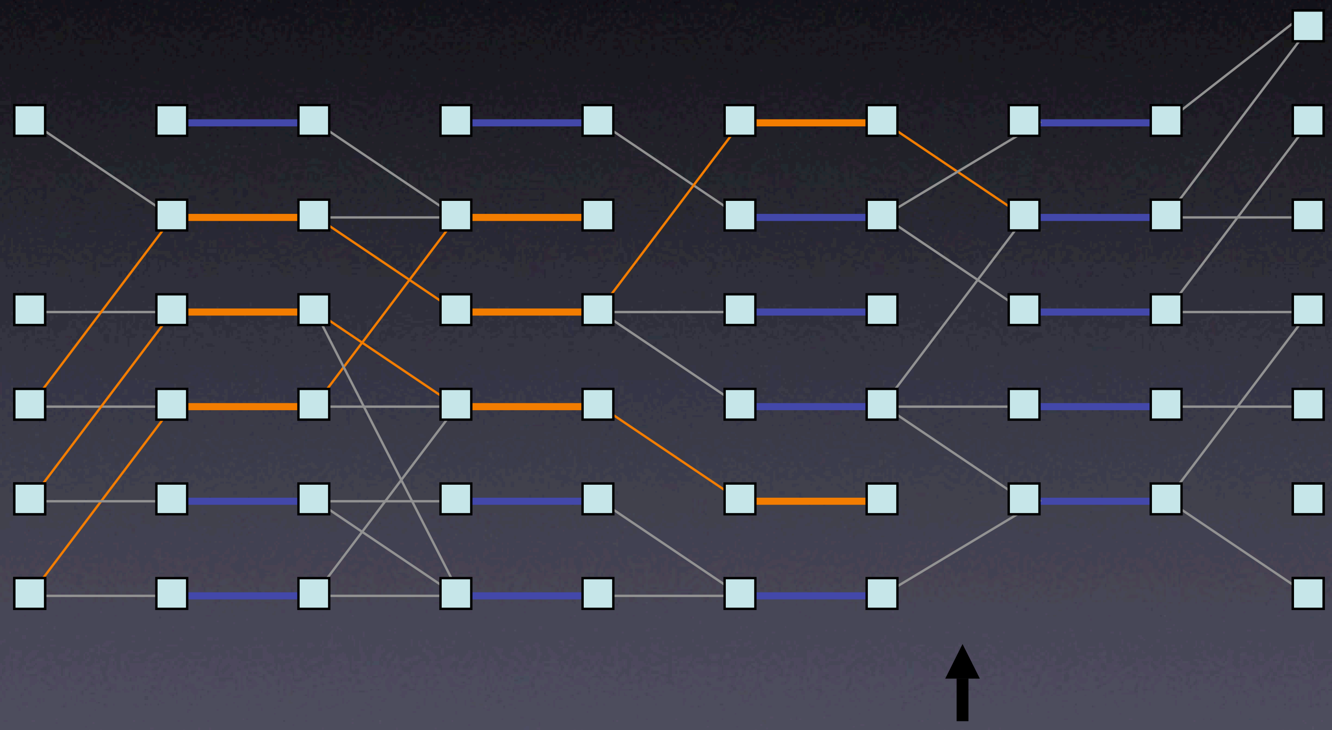


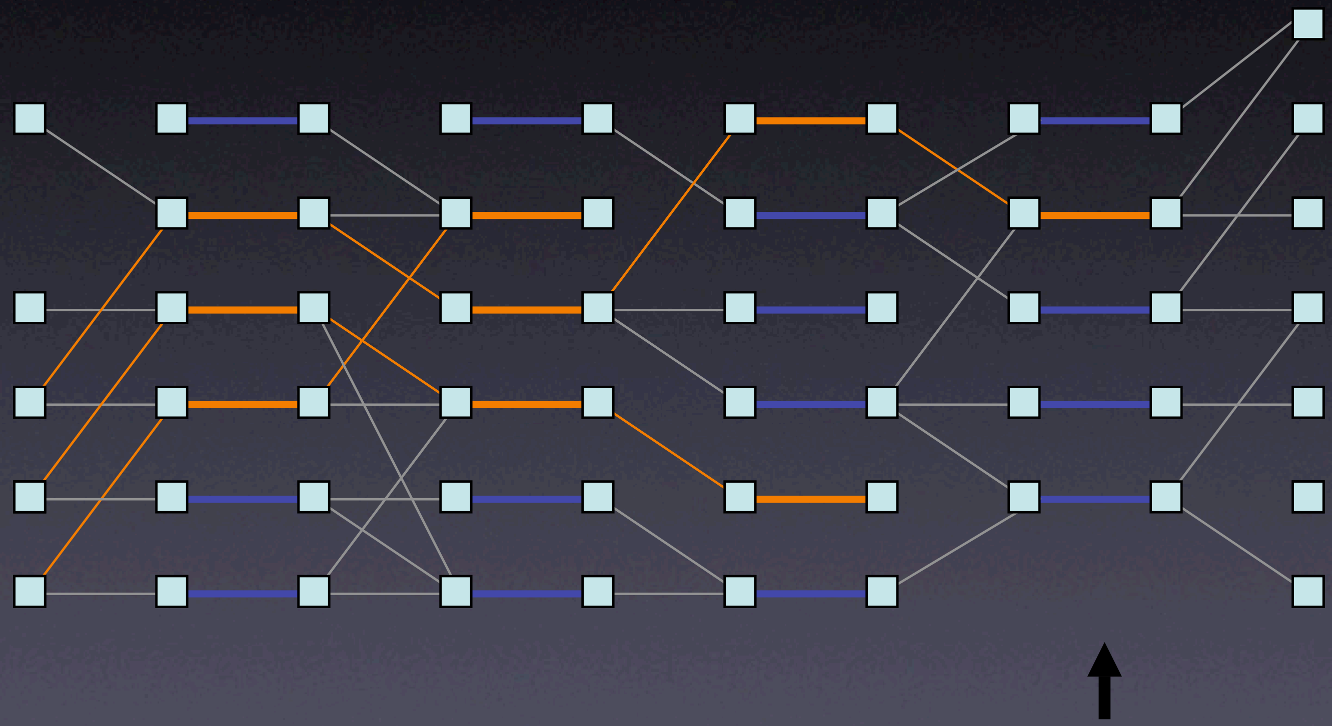


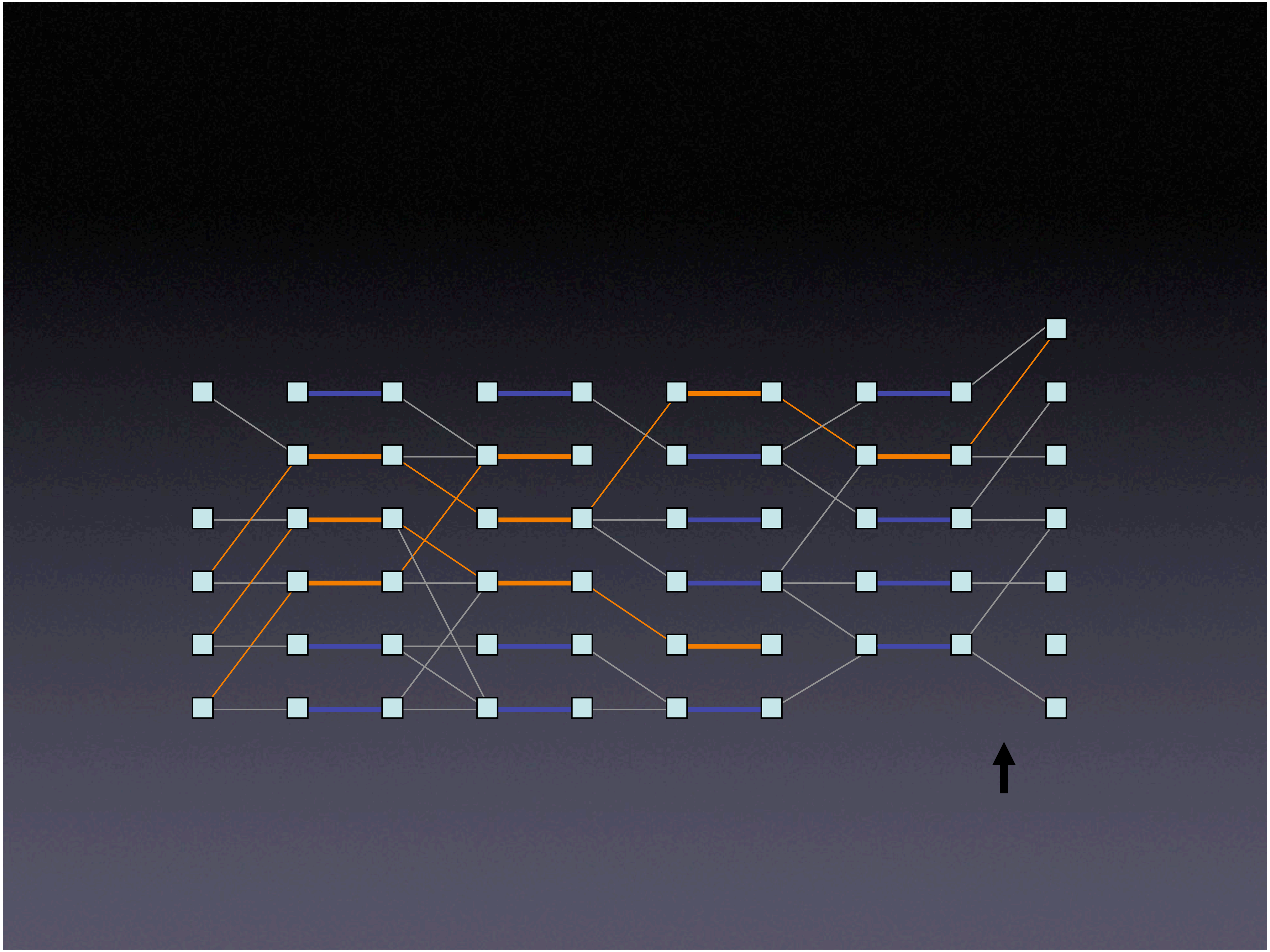


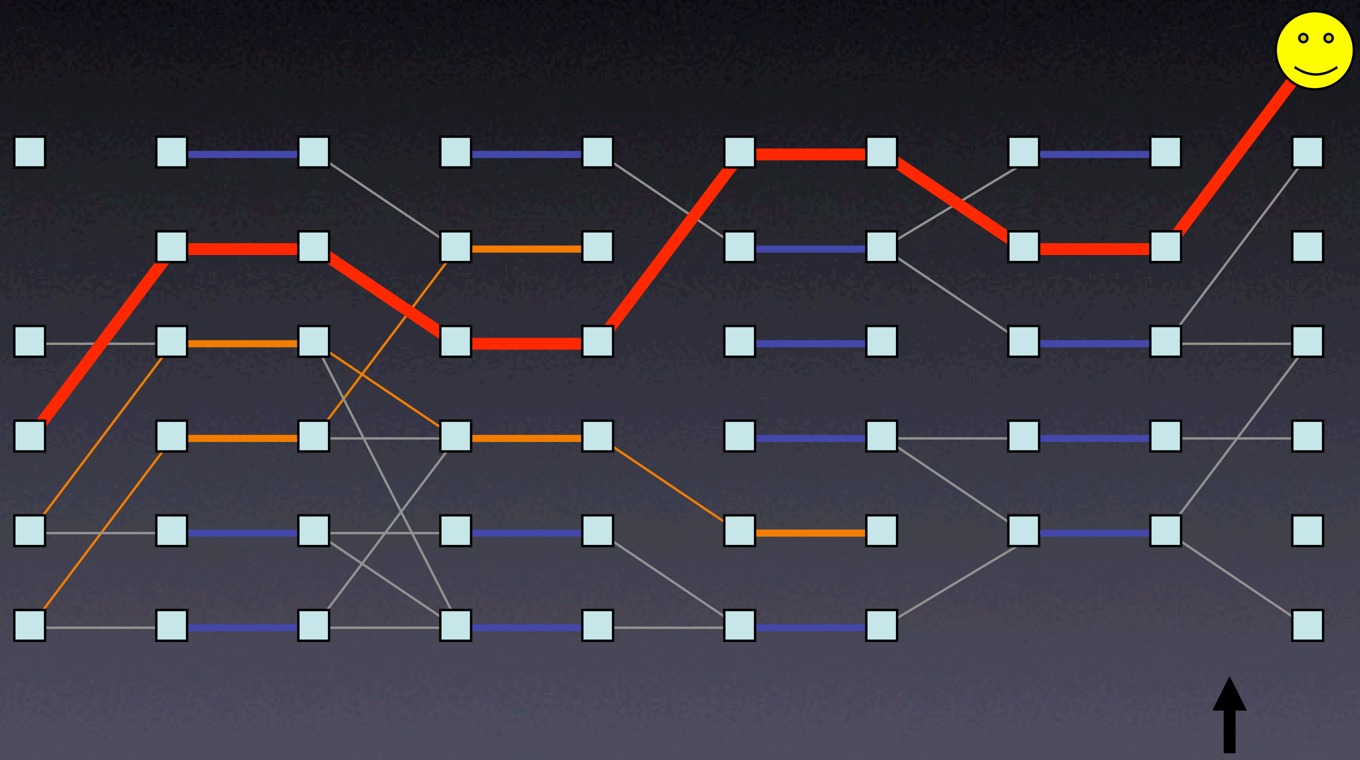


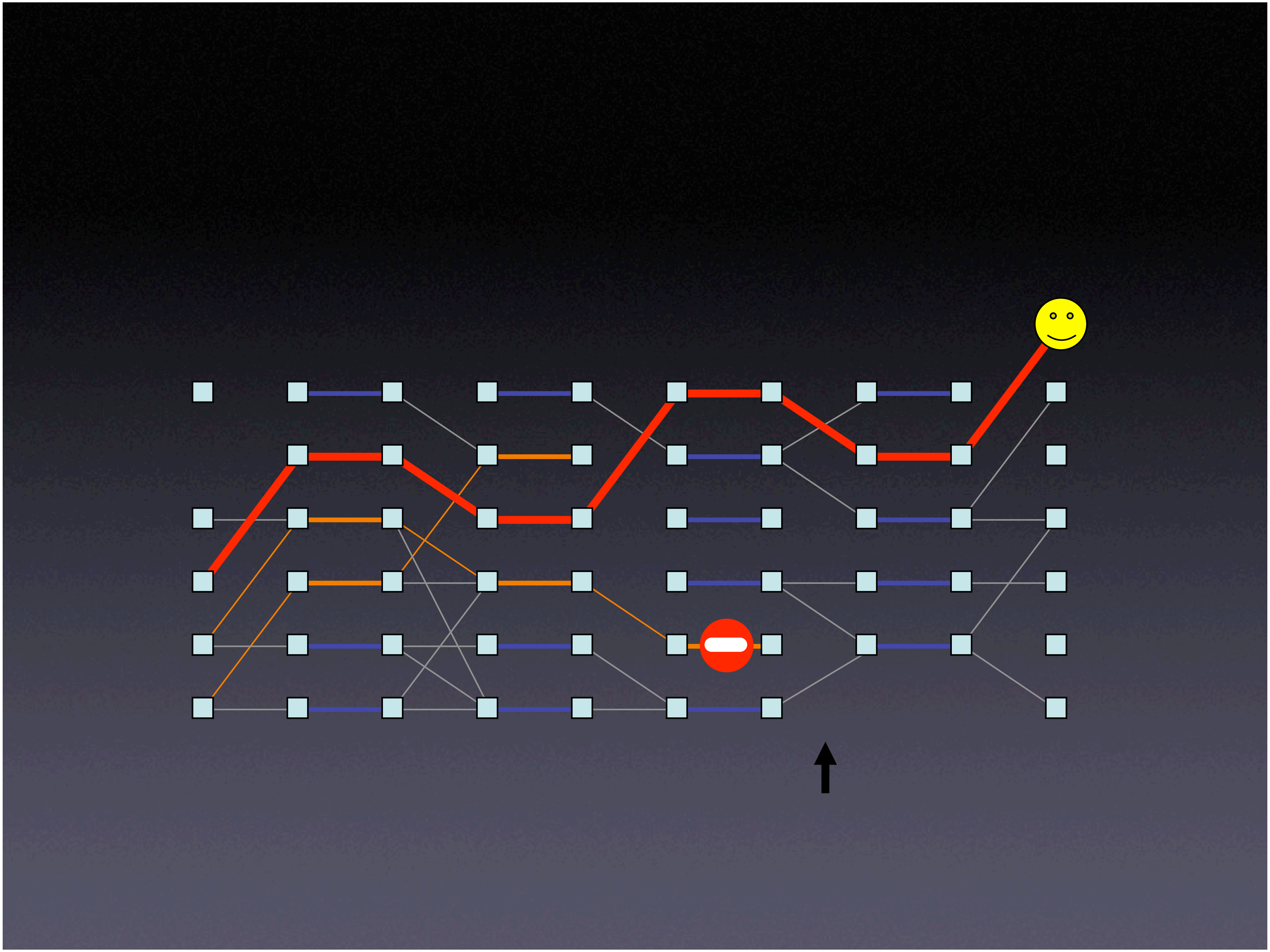


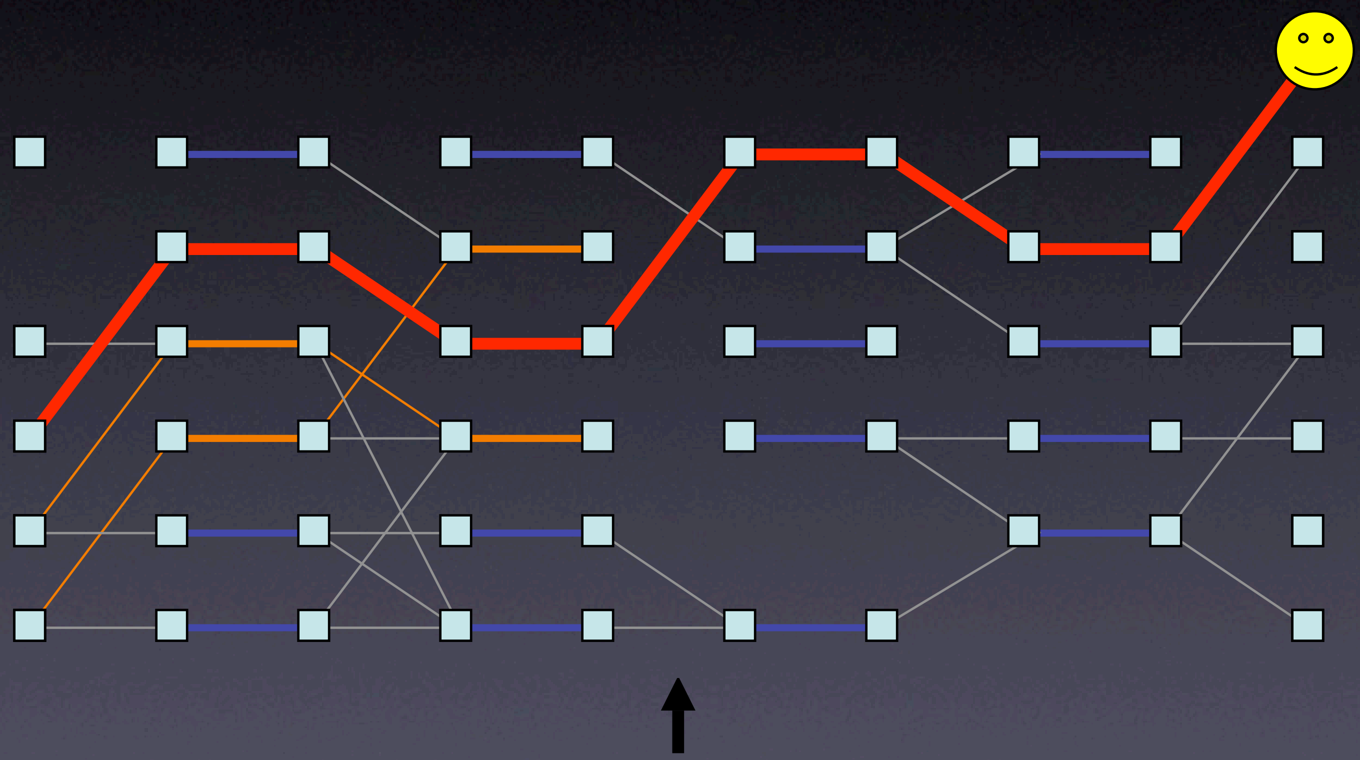


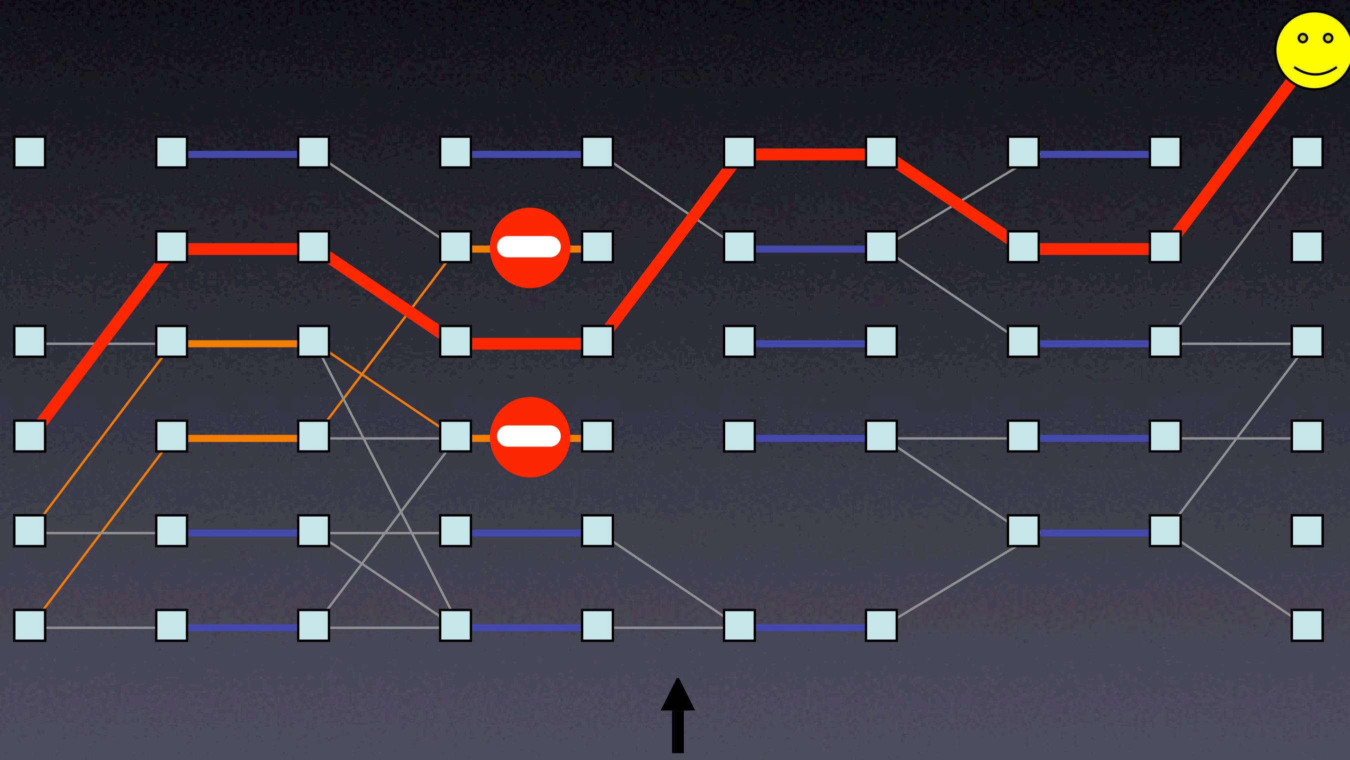


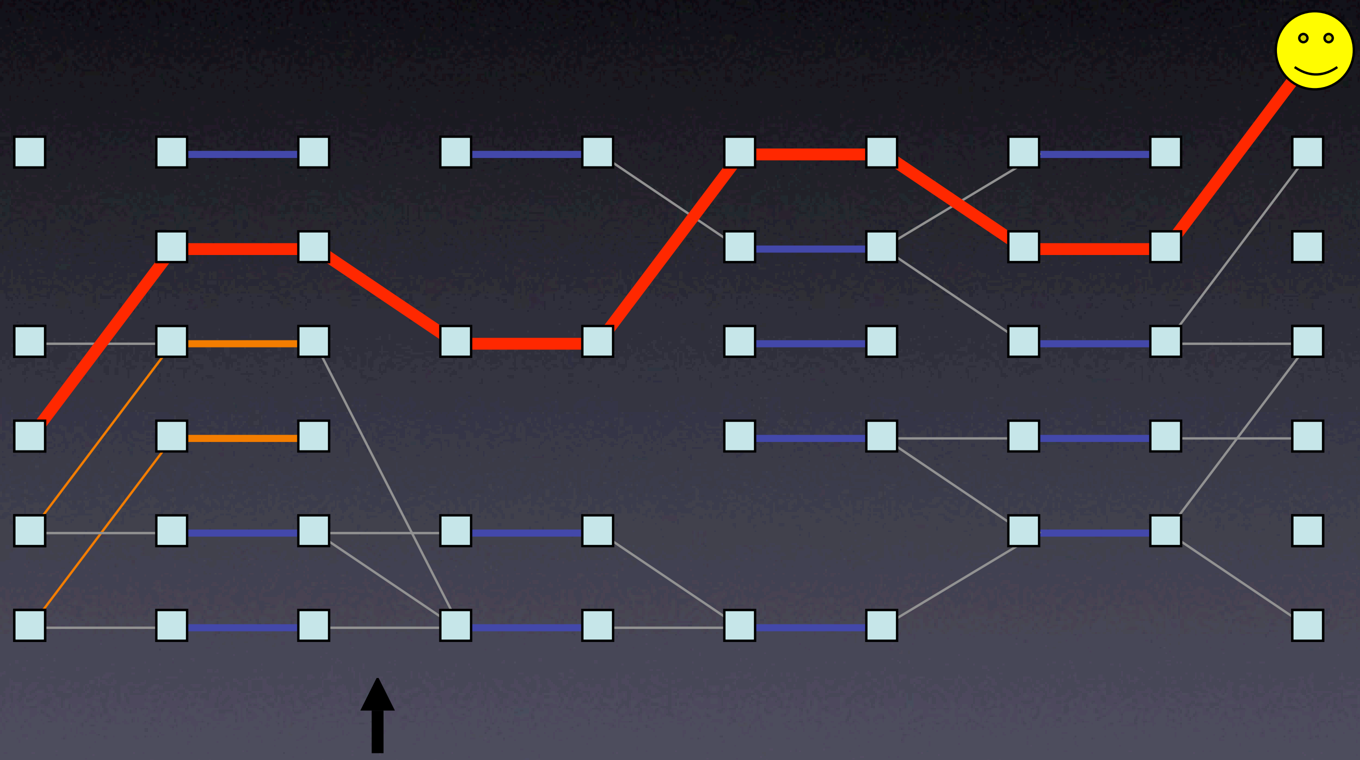


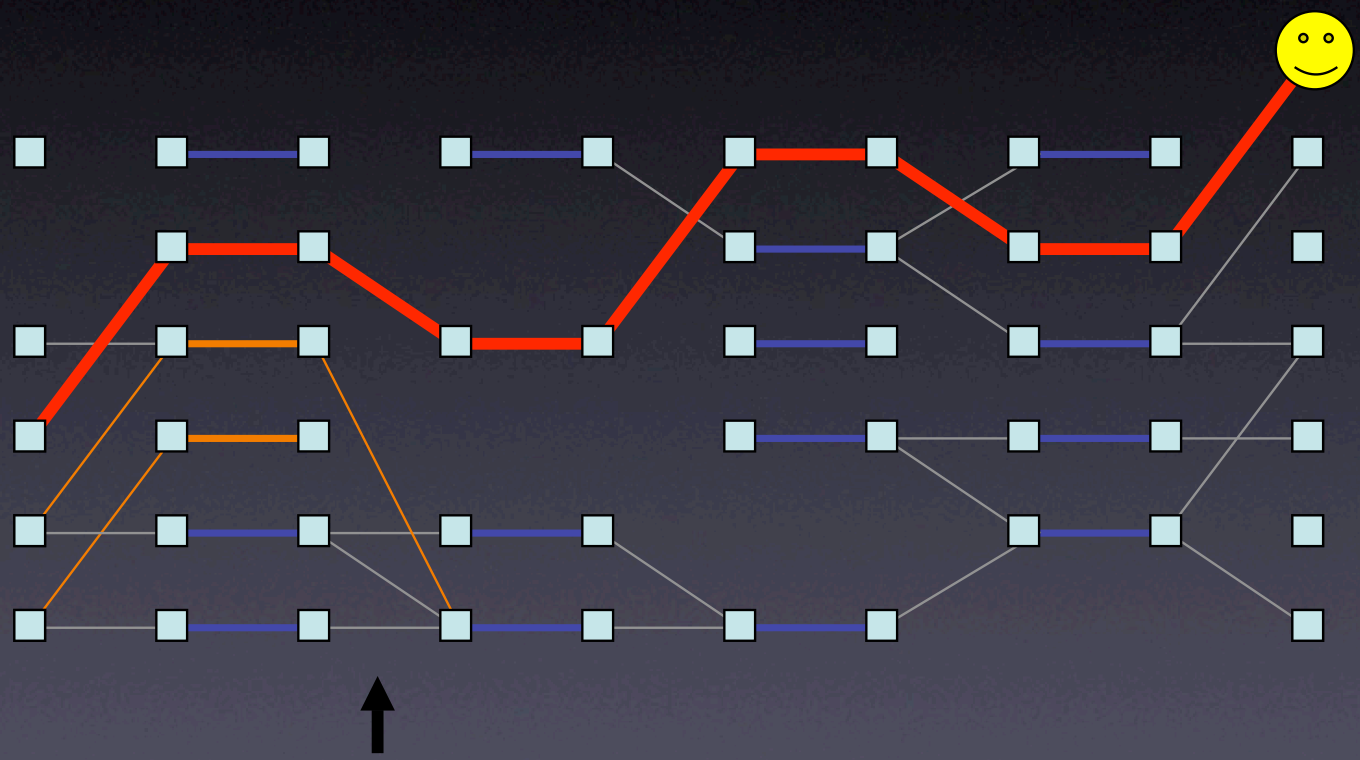


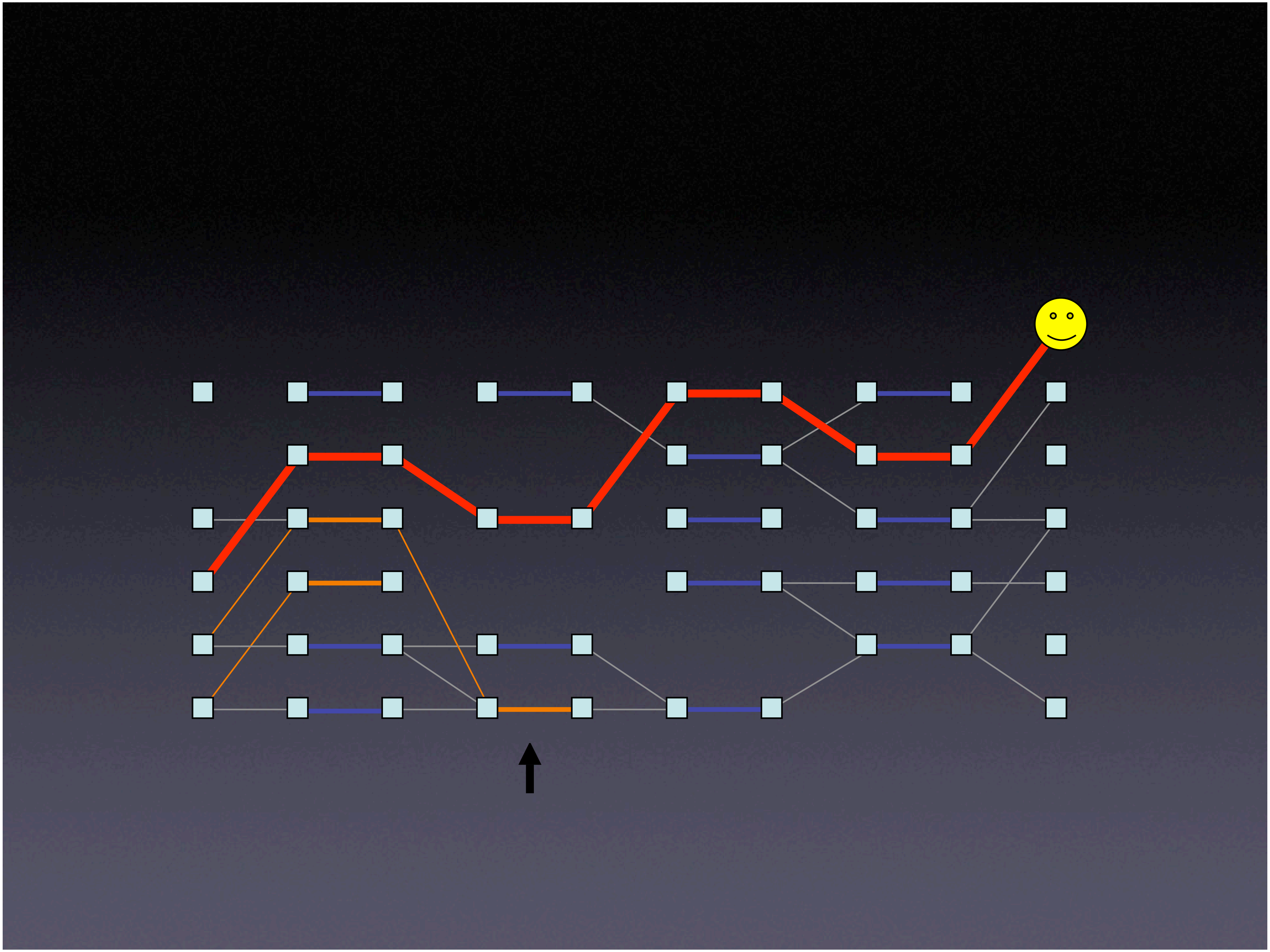


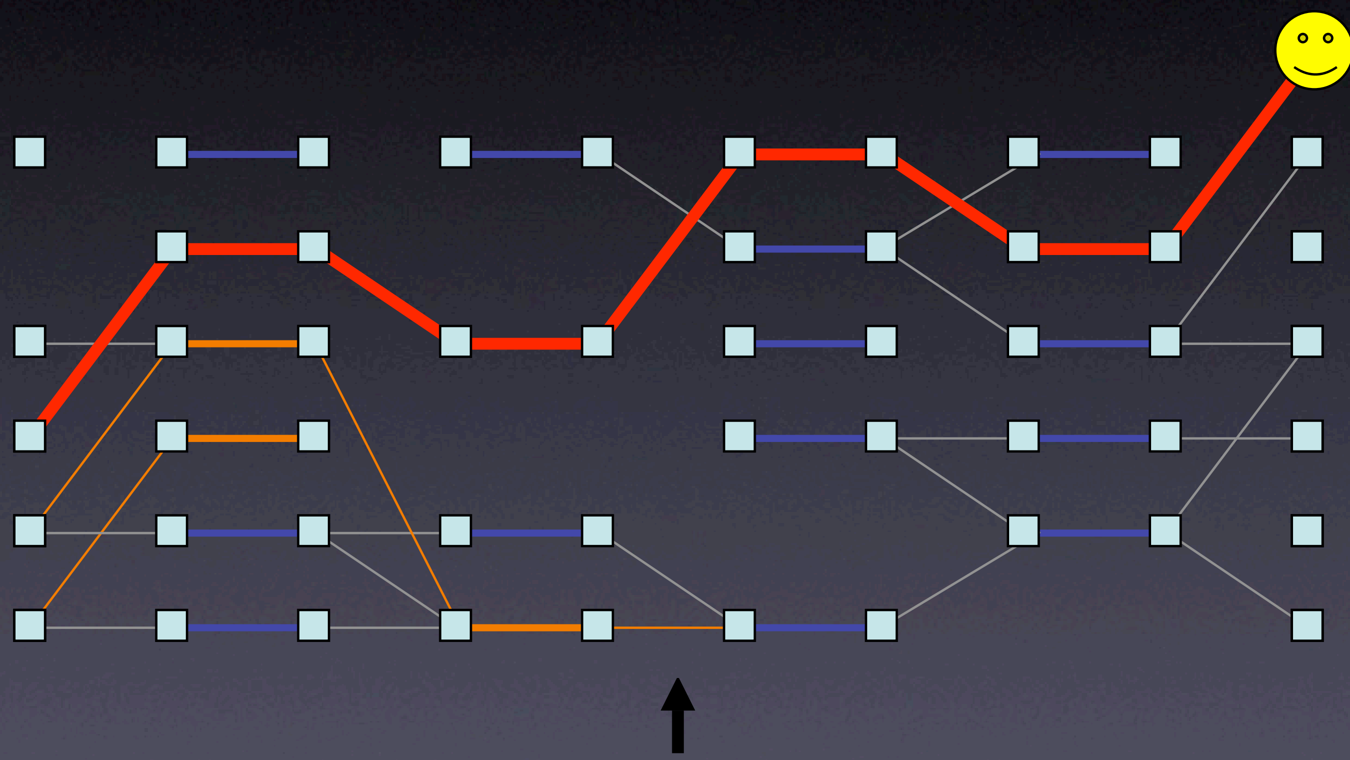


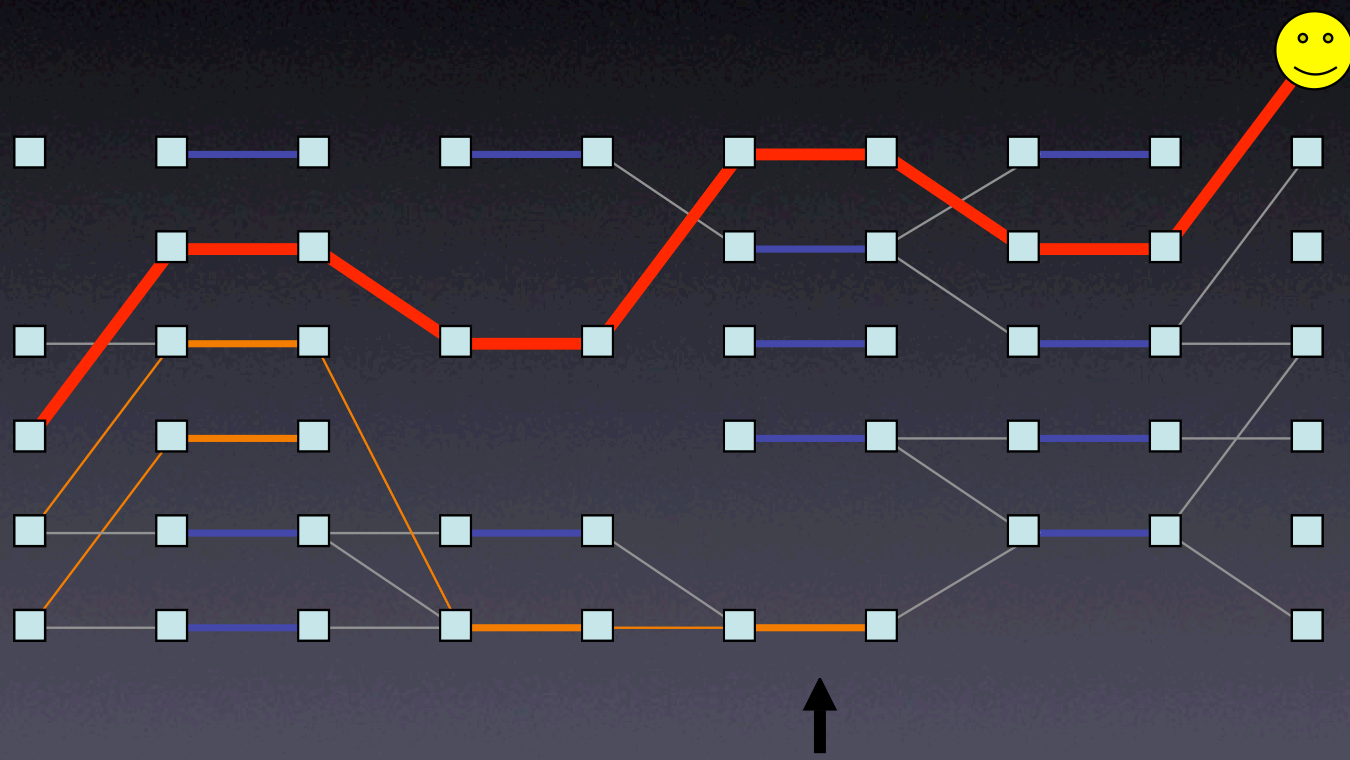


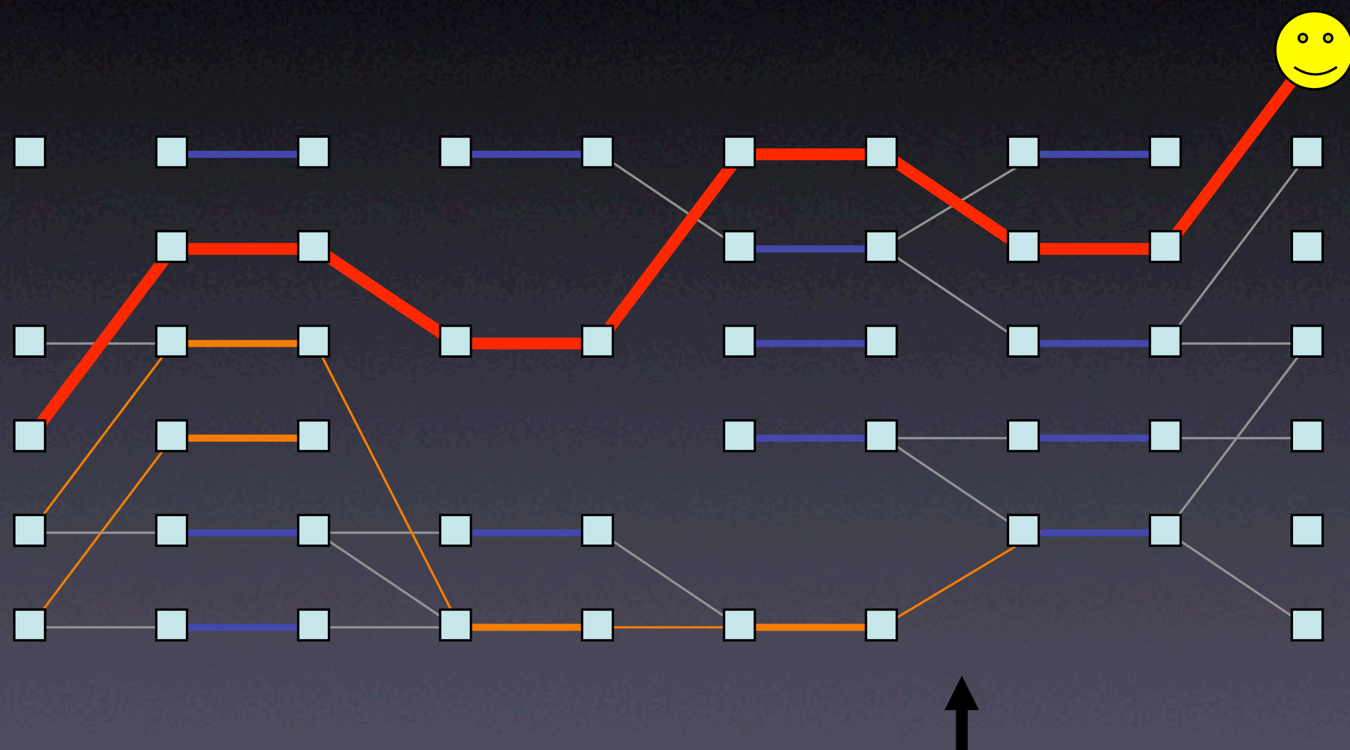


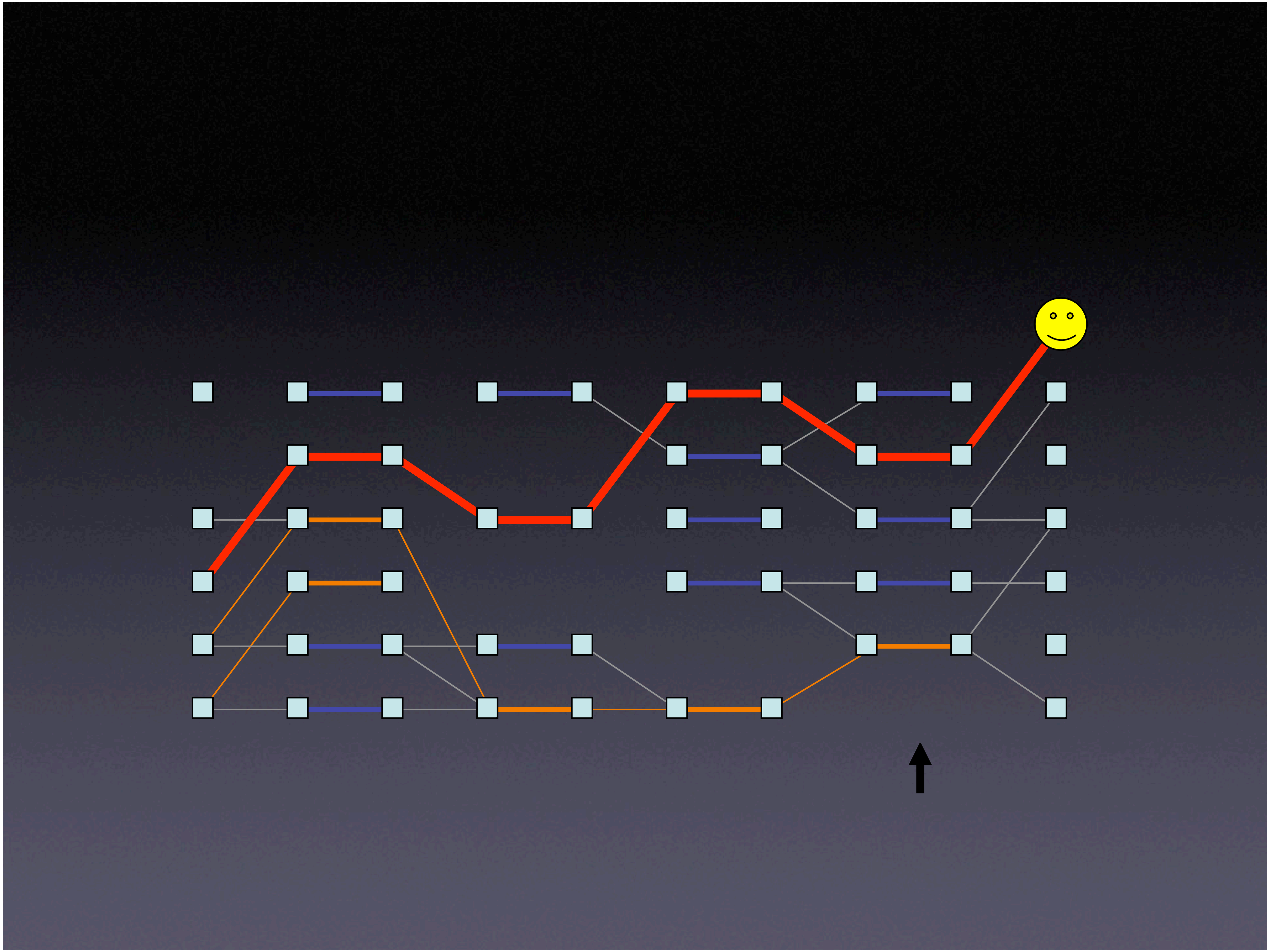


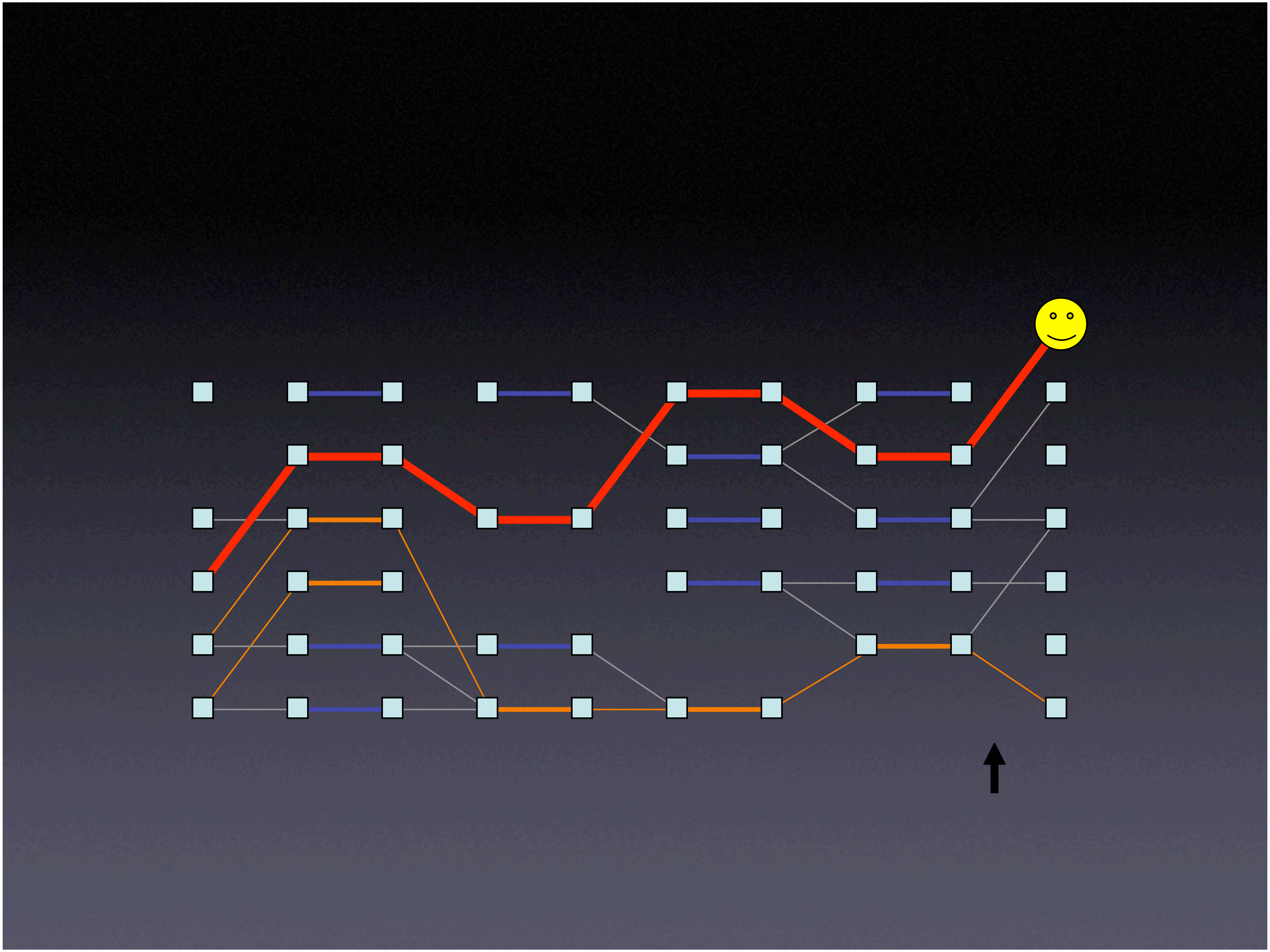


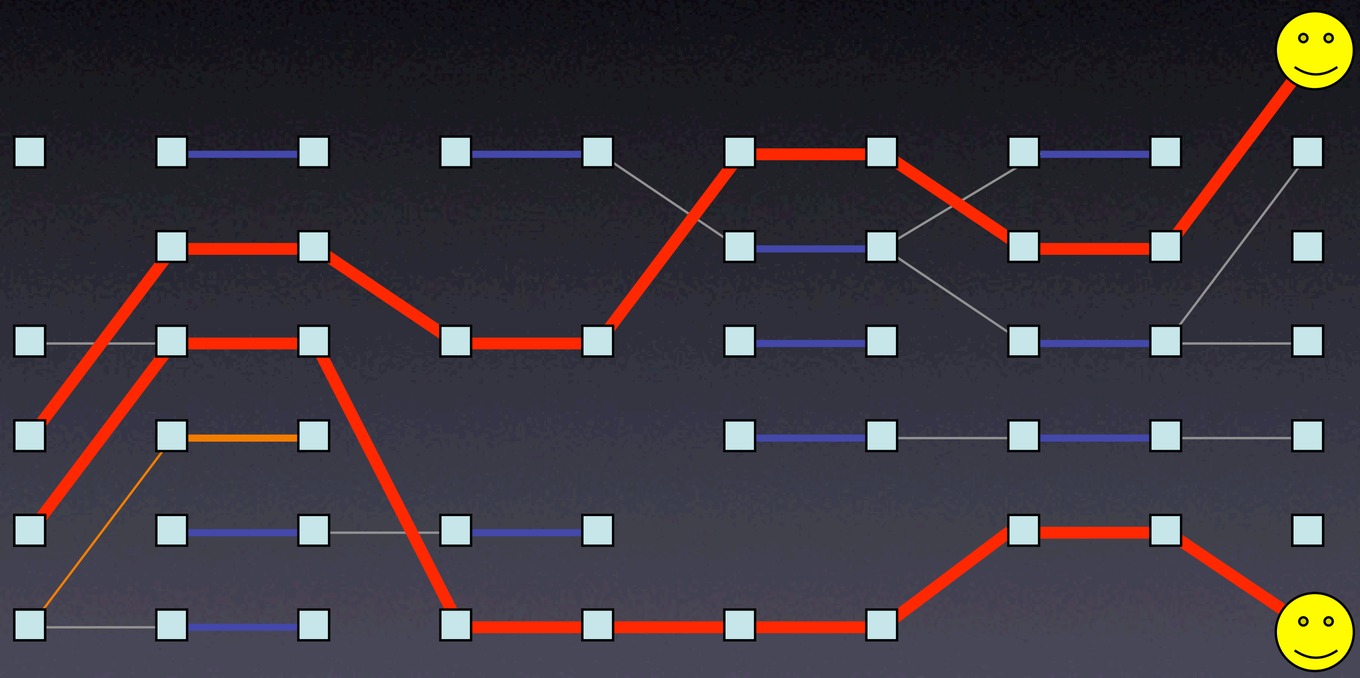




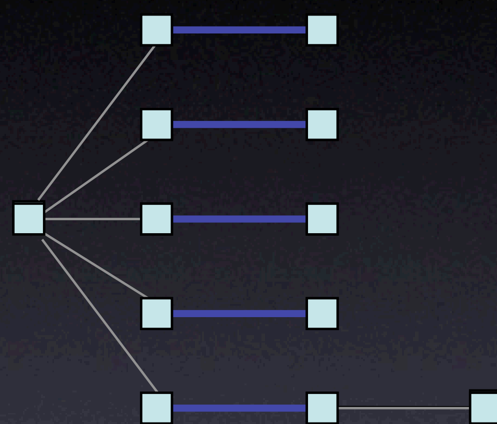




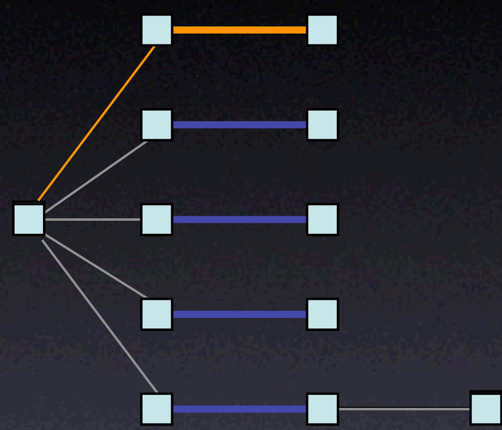




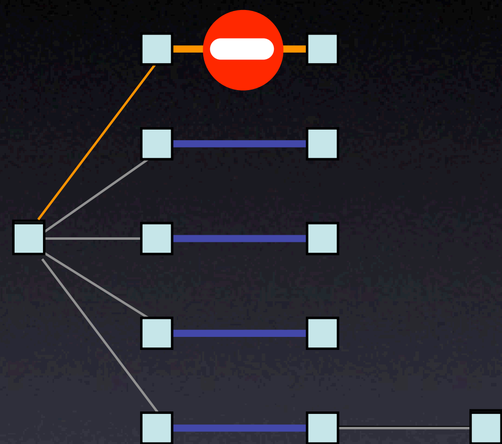
Limiting Backtracking



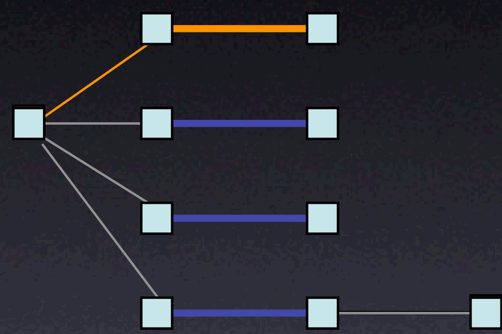
Limiting Backtracking



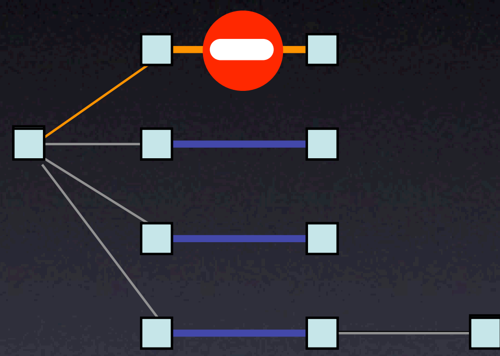
Limiting Backtracking



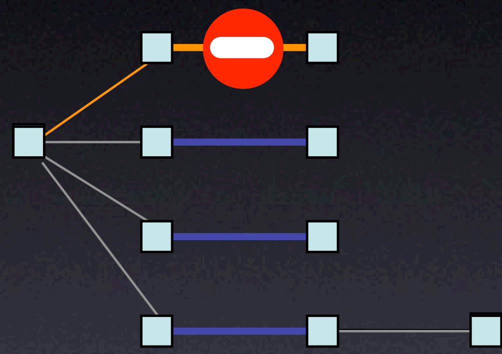
Limiting Backtracking



Limiting Backtracking



Limiting Backtracking



- Solution: If number of paths being grown falls below threshold δn then delete and backtrack.

  Good: Only backtrack a constant number of times

  Bad: Don't find a maximal set of node disjoint paths

- In a constant number of passes, we find a constant fraction of length i node disjoint paths/augmenting paths.

Weighted Matching.

Single Pass $3+2\sqrt{2}$ Approximation

Single Pass $3+2\sqrt{2}$ Approximation

- At all times we store some matching *M*

Single Pass $3+2\sqrt{2}$ Approximation

- At all times we store some matching M
- For each edge e:

    Compute total weight W of edges e_1, e_2 in M incident to e

    If $w(e) > (1+\gamma)\, W$ then $M \leftarrow M \cup \{e\} \setminus \{e_1, e_2\}$

Single Pass $3+2\sqrt{2}$ Approximation

- At all times we store some matching M
- For each edge e:

  Compute total weight W of edges e_1, e_2 in M incident to e

  If $w(e) > (1+\gamma)\, W$ then $M \leftarrow M \cup \{e\} \setminus \{e_1,e_2\}$

- We say e is "BORN" and "KILLED" e_1 and e_2

Proof (Sketch)

Proof (Sketch)

- We say an edge e is a **SURVIVOR** if it is born and was never killed.

Proof (Sketch)

- We say an edge e is a **SURVIVOR** if it is born and was never killed.
- Let S = all survivors.

Proof (Sketch)

- We say an edge e is a **SURVIVOR** if it is born and was never killed.
- Let S = all survivors.
- For survivor e we define the **TRAIL OF THE DEAD** $T(e)$ to be the transitive closure of edges killed by e.

Proof (Sketch)

- We say an edge e is a SURVIVOR if it is born and was never killed.
- Let S = all survivors.
- For survivor e we define the TRAIL OF THE DEAD $T(e)$ to be the transitive closure of edges killed by e.
- Claim 1: $w(T(e)) \leq w(e)/\gamma$

Proof (Sketch)

- We say an edge e is a **SURVIVOR** if it is born and was never killed.
- Let S = all survivors.
- For survivor e we define the **TRAIL OF THE DEAD** $T(e)$ to be the transitive closure of edges killed by e.
- Claim 1: $w(T(e)) \leq w(e)/\gamma$
- Claim 2: Can charge the weights of edges in OPT such that:
  - At most $(1+\gamma)\, w(T(e))$ is charged to $T(e)$
  - At most $2(1+\gamma)\, w(e)$ is charged to e

Proof (Sketch)

- We say an edge e is a **SURVIVOR** if it is born and was never killed.
- Let S = all survivors.
- For survivor e we define the **TRAIL OF THE DEAD** $T(e)$ to be the transitive closure of edges killed by e.
- Claim 1: $w(T(e)) \le w(e)/\gamma$
- Claim 2: Can charge the weights of edges in OPT such that:
  - At most $(1+\gamma)\, w(T(e))$ is charged to $T(e)$
  - At most $2(1+\gamma)\, w(e)$ is charged to e
- Hence $w(OPT) \le (1+\gamma)\, w(T(S)) + 2(1+\gamma)\, w(S) < (3+2\sqrt{2})\, w(S)$

Multi-pass $2+\epsilon$ Approximation

Multi-pass 2+ϵ Approximation

- First pass: find a constant approximate M_1

Multi-pass 2+ϵ Approximation

- First pass: find a constant approximate M_1
- Subsequent passes: create M_i from M_{i-1} by running the previous algorithm with $\gamma(\epsilon)$

Multi-pass 2+ϵ Approximation

- First pass: find a constant approximate M_1
- Subsequent passes: create M_i from M_{i-1} by running the previous algorithm with $\gamma(\epsilon)$
- Repeat if $|M_i|/ |M_{i-1}|> 1+\kappa(\epsilon)$

Multi-pass 2+ϵ Approximation

- First pass: find a constant approximate M_1
- Subsequent passes: create M_i from M_{i-1} by running the previous algorithm with $\gamma(\epsilon)$
- Repeat if $|M_i|/|M_{i-1}| > 1+\kappa(\epsilon)$
- Claim 1: A constant number of passes suffices

Multi-pass 2+ϵ Approximation

- First pass: find a constant approximate M_1
- Subsequent passes: create M_i from M_{i-1} by running the previous algorithm with $\gamma(\epsilon)$
- Repeat if $|M_i|/ |M_{i-1}|> 1+\kappa(\epsilon)$
- Claim 1: A constant number of passes suffices
- Claim 2: When $|M_i|/ |M_{i-1}| \leq 1+\kappa$ we have a $2+\epsilon$ approx.

Conclusions

- Unweighted Matchings:

  $1+\epsilon$ approximation in constant passes.

- Weighted Matchings:

  $3+2\sqrt{2}$ approximation in single pass.

  $2+\epsilon$ approximation in constant passes.


Thanks.