Maximum Matching in General Graphs

Ahmad Khayyat
Department of Electrical & Computer Engineering
ahmad.khayyat@ece.queensu.ca

Course Project
CISC 879 — Algorithms and Applications
Queen's University

November 19, 2008

Outline

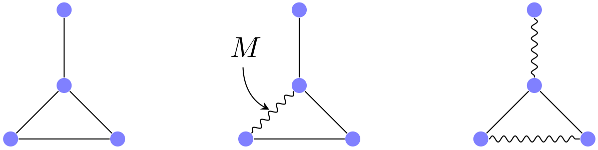
1. Introduction
2. Paths, Trees and Flowers
3. Efficient Implementation of Edmonds' Algorithm
4. Reachability Problem Approach
5. Conclusion

Outline

1. Introduction
   - Terminology
   - Berge's Theorem
   - Bipartite Matching
2. Paths, Trees and Flowers
3. Efficient Implementation of Edmonds' Algorithm
4. Reachability Problem Approach
5. Conclusion

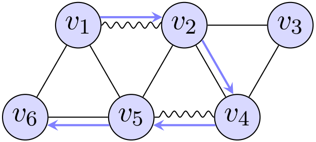
Maximum Matching

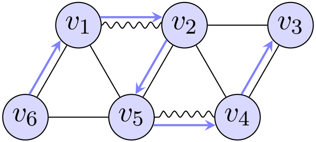
- $G = (V, E)$ is a finite undirected graph: $n = |V|, m = |E|$.
- A matching M in G, (G, M), is a subset of its edges such that no two meet the same vertex.
- M is a maximum matching if no other matching in G contains more edges than M.
- A maximum matching is not necessarily unique.
- Given (G, M), a vertex is exposed if it meets no edge in M.



Augmenting Paths

- An alternating path in (G, M) is a simple path whose edges are alternately in M and not in M.
- An augmenting path is an alternating path whose ends are distinct exposed vertices.

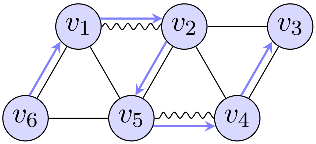


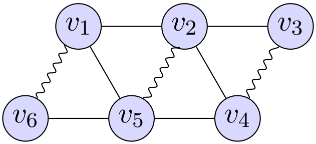


Berge's Theorem

Berge's Theorem (1957)

A matched graph (G, M) has an augmenting path if and only if M is not maximum.





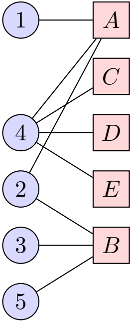
Unique?

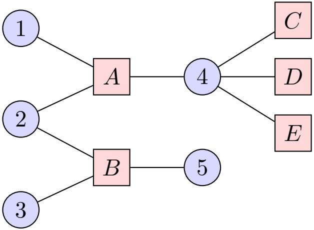
An Exponential Algorithm:

Exhaustively search for an augmenting path starting from an exposed vertex.

Bipartite Graphs

- A bipartite graph $G = (A, B, E)$ is a graph whose vertices can be divided into two disjoint sets A and B such that every edge connects a vertex in A to one in B.
- Equivalently, it is a graph with no odd cycles.

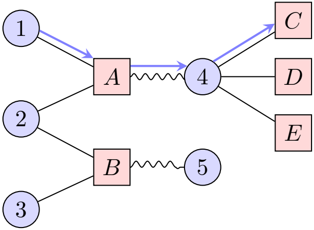




Bipartite Graph Maximum Matching

$O(nm)$ {

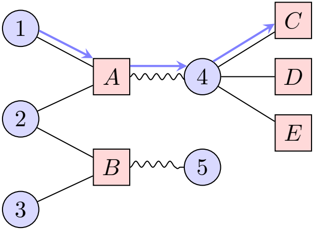
for all $v \in A$, v is exposed **do**
  Search for simple alternating paths starting at v
  if path P ends at an exposed vertex $u \in B$ **then**
    P is an augmenting path {Update M}
  end if
end for
Current M is maximum {No more augmenting paths}

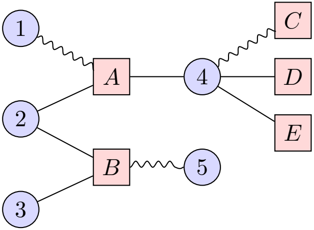


Bipartite Graph Maximum Matching

$O(nm)$

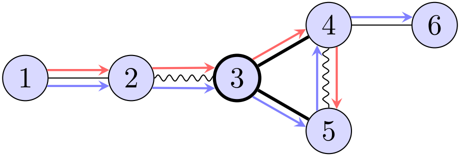
for all $v \in A$, v is exposed **do**
  Search for simple alternating paths starting at v
  if path P ends at an exposed vertex $u \in B$ **then**
    P is an augmenting path {Update M}
  end if
end for
Current M is maximum {No more augmenting paths}

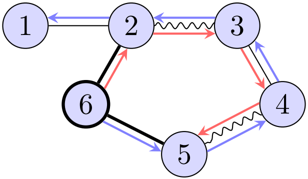




Non-Bipartite Matching

Problem: Odd cycles . . .





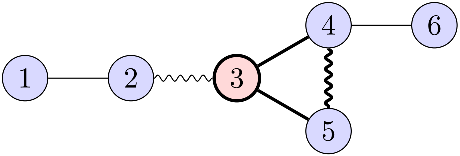
Outline

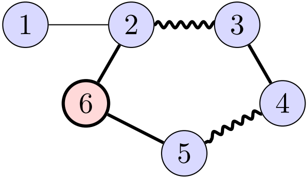
1. Introduction
2. Paths, Trees and Flowers
   - Blossoms
   - The Algorithm
3. Efficient Implementation of Edmonds' Algorithm
4. Reachability Problem Approach
5. Conclusion

Blossoms

Blossoms

A blossom B in (G, M) is an odd cycle with a unique exposed vertex (the base) in $M \cap B$.

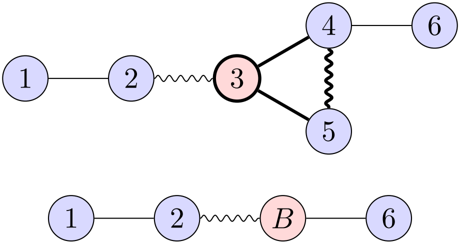




Edmonds' Blossoms Lemma

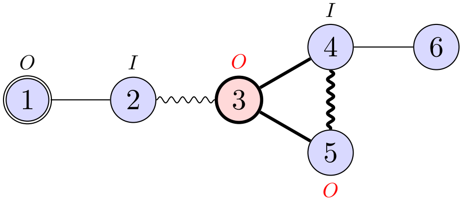
Blossoms Lemma

Let G' and M' be obtained by contracting a blossom B in (G, M) to a single vertex.
The matching M of G is maximum iff M' is maximum in G'.



Detecting Blossoms

- Performing the alternating path search of the bipartite matching algorithm (starting from an exposed vertex):
  - Label vertices at even distance from the root as "outer";
  - Label vertices at odd distance from the root as "inner".
- If two outer vertices are found adjacent, we have a blossom.



Edmonds' Algorithm (1965)

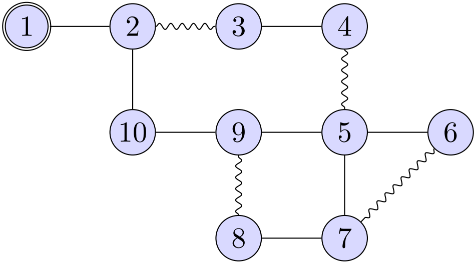
$O(n^3)$ $\{$ $O(n^2)$

for all $v \in V$, v is exposed **do**
  Search for simple alternating paths starting at v
    Shrink any found blossoms
  if path P ends at an exposed vertex **then**
    P is an augmenting path {Update M}
  else if no augmenting paths found **then**
    Ignore v in future searches
  end if
end for
Current M is maximum {No more augmenting paths}

Complexity: $O(n^4)$

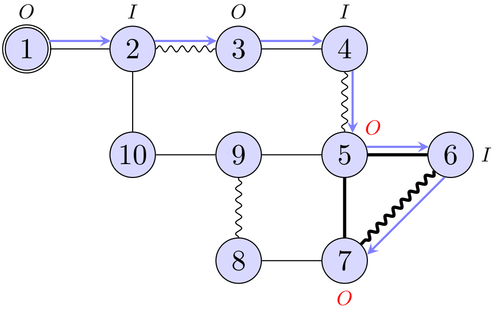
Example

$|M| = 4$



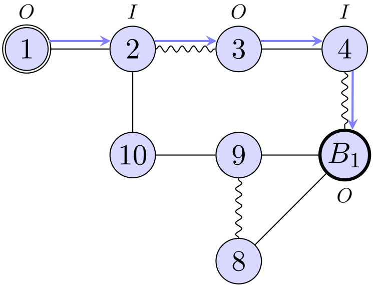
Example

$|M| = 4$



Example

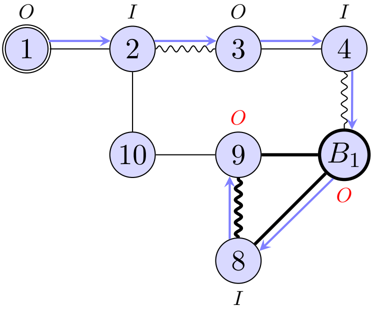
$|M| = 4$



$B_1 = 5, 6, 7$

Example

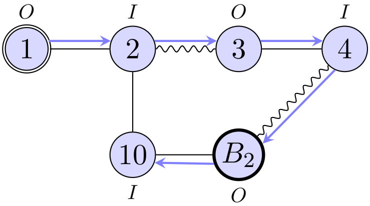
$|M| = 4$



$B_1 = 5, 6, 7$

Example

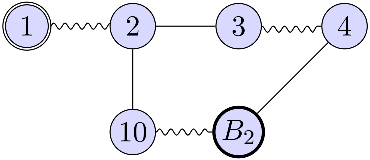
$|M| = 4$



$B_1 = 5, 6, 7$
$B_2 = B_1, 8, 9 = 5, 6, 7, 8, 9$

Example

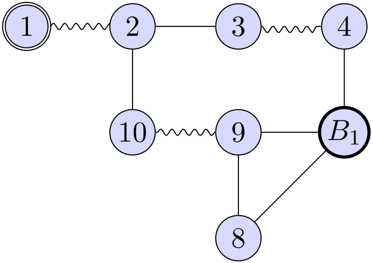
$|M| = 4$



$B_1 = 5, 6, 7$
$B_2 = B_1, 8, 9 = 5, 6, 7, 8, 9$

Example

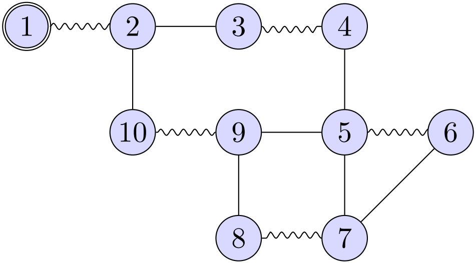
$|M| = 4$



$B_1 = 5, 6, 7$
$B_2 = B_1, 8, 9 = 5, 6, 7, 8, 9$

Example

$|M| = 4$ $|M| = 5$



$B_1 = 5, 6, 7$
$B_2 = B_1, 8, 9 = 5, 6, 7, 8, 9$

Outline

1. Introduction
2. Paths, Trees and Flowers
3. Efficient Implementation of Edmonds' Algorithm
   - Data Structures
   - The Algorithm
   - Performance
4. Reachability Problem Approach
5. Conclusion

Three Arrays

- u is an exposed vertex.
- A vertex v is outer if there is a path $P(v) = (v, v_1, \ldots, u)$, where $vv_1 \in M$.

1. *MATE:* Specifies a matching. An entry for each vertex:
   - $vw \in M \Rightarrow MATE(v) = w$ and $MATE(w) = v$.
2. *LABEL:* Provides a type and a value:

$$\begin{aligned} LABEL(v) &\geq 0 & &\rightarrow v \text{ is outer} \\ LABEL(u) & & &\rightarrow \text{start label, } P(u) = (u) \\ 1 \leq LABEL(v) &\leq n & &\rightarrow \text{vertex label} \\ n+1 \leq LABEL(v) &\leq n+2m & &\rightarrow \text{edge label} \end{aligned}$$

3. $START(v)$ = the first non-outer vertex in $P(v)$.

Gabow's Algorithm (1976)

for all $u \in V, u$ is exposed **do**
  while $\exists$ an edge xy, x is outer AND
      no augmenting path found **do**
    if y is exposed, $y \neq u$ **then**
      $(y, x, \ldots, u)$ is an augmenting path
    else if y is outer **then**
      Assign edge labels to $P(x)$ and $P(y)$
    else if $MATE(y)$ is non-outer **then**
      Assign a vertex label to $MATE(y)$
    end if
  end while
end for

Gabow's Algorithm (1976)

$O(n)$ **for all** $u \in V, u$ is exposed **do**
  while $\exists$ an edge xy, x is outer AND no augmenting path found **do**
$O(1)$ **if** y is exposed, $y \neq u$ **then**
$O(n)$ $(y, x, \ldots, u)$ is an augmenting path
$O(n)$ **else if** y is outer **then**
$O(n)$ Assign edge labels to $P(x)$ and $P(y)$
$O(n)$ **else if** $MATE(y)$ is non-outer **then**
$O(1)$ Assign a vertex label to $MATE(y)$
  end if
  end while
end for

Complexity: $O(n^3)$

Experimental Performance

Using an implementation in Algol W on the IBM 360/165

- Worst-case graphs:
  - Efficient Implementation: run times proportional to $n^{2.8}$.
  - Edmond: run times proportional to $n^{3.5}$.
- Random graphs: times one order of magnitude faster than worst-case graphs.
- Space used is $5n + 4m$.

Outline

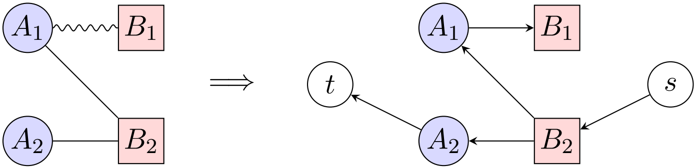
1. Introduction
2. Paths, Trees and Flowers
3. Efficient Implementation of Edmonds' Algorithm
4. Reachability Problem Approach
   - Reachability and Graphs
   - The Algorithm
5. Conclusion

The Reachability Problem in Bipartite Graphs

- Construction:

$$\begin{array}{ccccc} \text{Bipartite graph} & + & \text{Matching} & \rightarrow & \text{Directed graph} \\ G=(A,B,E) & + & M & \rightarrow & G'=(V',E') \end{array}$$

  - $V' = V \cup \{s, t\}$
  - $\forall\, xy \in M, x \in A, y \in B \rightarrow (x, y) \in E'$ $\qquad e \in M \Rightarrow e : A \rightarrow B$
  - $\forall\, xy \notin M, x \in A, y \in B \rightarrow (y, x) \in E'$ $\qquad e \notin M \Rightarrow e : B \rightarrow A$
  - $\forall\, b \in B, b$ is exposed $\rightarrow$ add (s, b) to E'
  - $\forall\, a \in A, a$ is exposed $\rightarrow$ add (a, t) to E'

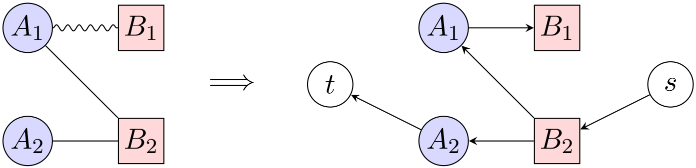


The Reachability Problem in Bipartite Graphs

- Construction:

$$\begin{array}{ccccc} \text{Bipartite graph} & + & \text{Matching} & \rightarrow & \text{Directed graph} \\ G = (A, B, E) & + & M & \rightarrow & G' = (V', E') \end{array}$$

  - $V' = V \cup \{s, t\}$
  - $\forall\ xy \in M, x \in A, y \in B \rightarrow (x, y) \in E'$ $e \in M \Rightarrow e : A \rightarrow B$
  - $\forall\ xy \notin M, x \in A, y \in B \rightarrow (y, x) \in E'$ $e \notin M \Rightarrow e : B \rightarrow A$
  - $\forall\ b \in B, b$ is exposed $\rightarrow$ add (s, b) to E'
  - $\forall\ a \in A, a$ is exposed $\rightarrow$ add (a, t) to E'



- An augmenting path in $G \Leftrightarrow$ A simple path from s to t in G'.

The Reachability Problem in General Graphs

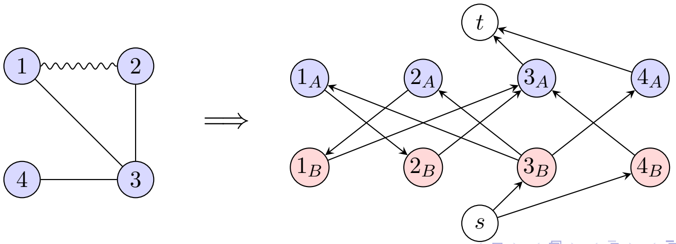
Construction

- For each $v \in V$, we introduce two nodes v_A and v_B

$$V' = \{v_A, v_B | v \in V\} \cup \{s, t\} \qquad s, t \notin V, s \neq t$$

- $e \in M \Rightarrow e : A \rightarrow B, \qquad e \notin M \Rightarrow e : B \rightarrow A$

$$\begin{aligned} E' = \quad & \{(x_A, y_B), (y_A, x_B) \,|\, (x, y) \in M\} \\ & \cup \{(x_B, y_A), (y_B, x_A) \,|\, (x, y) \notin M\} \\ & \cup \{(s, x_B) \,|\, x \text{ is exposed}\} \cup \{(x_A, t) \,|\, x \text{ is exposed}\} \end{aligned}$$



The Reachability Problem in General Graphs

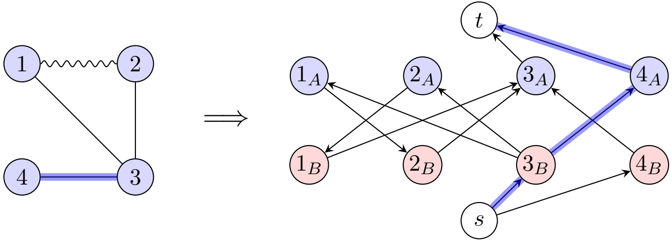
Strongly Simple Paths

A path P in G' is strongly simple if:

- P is simple.
- $v_A \in P \Rightarrow v_B \notin P$.

Theorem

There is an augmenting path in G if and only if there is a strongly simple path from s to t in G'.

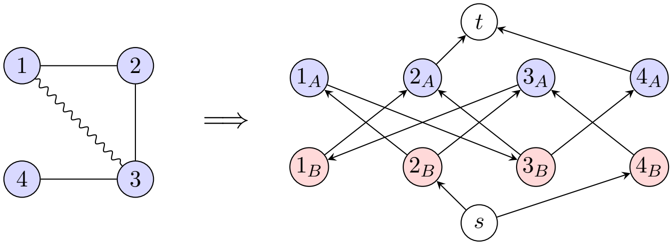


Solving the Reachability Problem

Solution: A strongly simple path from s to t in G':

- Depth-First Search (DFS) for t starting at s.
- DFS finds simple paths.
- We need to find strongly simple paths only.
- We use a Modified Depth-First Search (MDFS) algorithm.

Example:

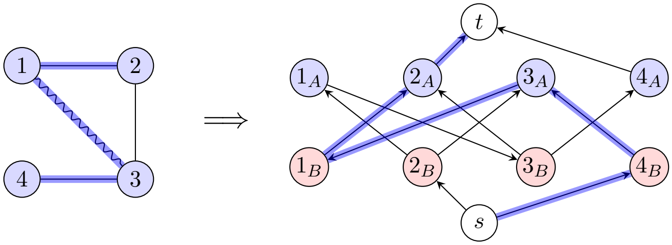


Solving the Reachability Problem

Solution: A strongly simple path from s to t in G':

- Depth-First Search (DFS) for t starting at s.
- DFS finds simple paths.
- We need to find strongly simple paths only.
- We use a Modified Depth-First Search (MDFS) algorithm.

Example:



Data Structures

- Stack K
  - $\mathrm{TOP}(K)$: the last vertex added to the stack K.
  - Vertices in K form the current path.
  - In each step, the MDFS algorithm considers an edge $(\mathrm{TOP}(K), v)$, $v \in V'$.
- List $L(v_A)$
  - To get a strongly simple path, v_A and v_B cannot be in K simultaneously (we may ignore a vertex, *temporarily*).
  - List $L(v_A)$ keeps track of such vertices.

Hopcroft and Karp Algorithm for Bipartite Graphs (1973)

Step 1: $M \leftarrow \phi$

Step 2: Let $l(M)$ be the length of a shortest augmenting path of M
Find a maximal set of paths $\{Q_1, Q_2, \ldots, Q_t\}$ such that:

2.1 For each i, Q_i is an augmenting path of M, $|Q_i| = l(M)$, Q_i are vertex-disjoint.

2.2 Halt if no such paths exists.

Step 3: $M \leftarrow M \oplus Q_1 \oplus Q_2 \oplus \cdots \oplus Q_t$; Go to 1.

Hopcroft and Karp Theorem

If the cardinality of a maximum matching is s, then this algorithm constructs a maximum matching within $2\lfloor\sqrt{s}\rfloor + 2$ executions of Step 2.

Step 2 complexity: $O(m) \Rightarrow$ Overall complexity: $O(\sqrt{n}m)$

An $O(\sqrt{n}m)$ Algorithm for General Graphs

- Blum describes an $O(m)$ implementation of Step 2 for general graphs, using a Modified Breadth-First Search (MBFS).
- Blum's Step 2 Algorithm:

Step 1: Using MBFS, compute $\overline{G'}$

Step 2: Using MDFS, compute a maximal set of strongly simple paths from s to t in $\overline{G'}$.

Blum's Theorm

A maximum matching in a general graph can be found in time $O(\sqrt{n}m)$ and space $O(m+n)$.

Outline

1 Introduction

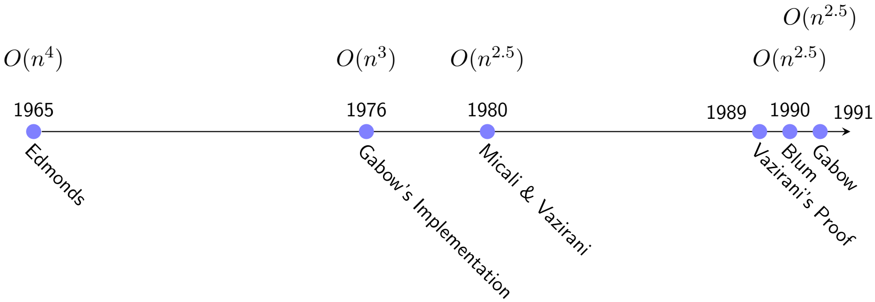
2 Paths, Trees and Flowers

3 Efficient Implementation of Edmonds' Algorithm

4 Reachability Problem Approach

5 Conclusion
- Summary
- Questions

Summary


$O(n^4)$
$O(n^3)$
$O(n^{2.5})$
$O(n^{2.5})$
$O(n^{2.5})$
1965
1976
1980
1989
1990
1991
Edmonds
Gabow's Implementation
Micali & Vazirani
Vazirani's Proof
Blum
Gabow

References

- Jack Edmonds
  Paths, Trees, and Flowers
  Canadian Journal of Mathematics, 17:449–467, 1965.
  http://www.cs.berkeley.edu/~christos/classics/edmonds.ps

- Harold N. Gabow
  An Efficient Implementation of Edmonds' Algorithm for Maximum Matching on Graphs
  Journal of the ACM (JACM), 23(2):221–234, 1976.

- Norbert Blum
  A New Approach to Maximum Matching in General Graphs
  Lecture Notes in Computer Science: Automata, Languages and Programming, 443::586–597, Springer Berlin / Heidelberg, 1990.

Thank You

Thank You!

Questions

???