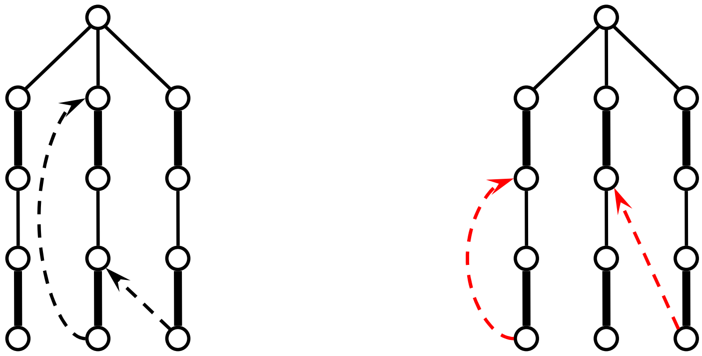
General Matching

- More difficult than bipartite matching:

  Presence of *Blossoms* (*odd length* alternating cycles)

  (In bipartite graphs, all cycles are *even* length)

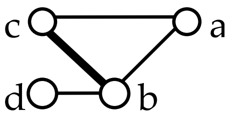
Bipartite Graphs *General Graphs*



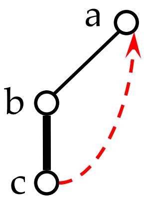
There are only *even* cycles. There are *odd* cycles (*blossoms*!). (May cause problem if not correctly handled.)

-- Cannot ignore the back edges

(will *miss* some augmenting path)



Graph

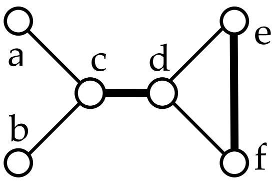


Alternating tree

If the back edge is ignored, then will never find the augment path a-c=b-d

-- Cannot just simply use back edge to "grow" the tree

(will create a *wrong* augmenting path)

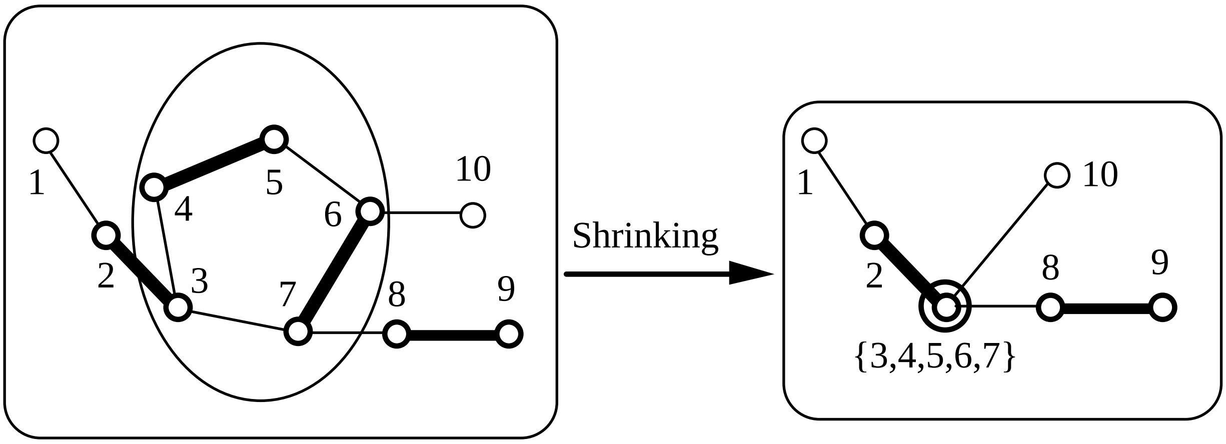


If the back edge is just simply used as a leaf edge, then may get wrong augmenting path. eg: a-c=d-e=f-d=c-b

General Matching - Blossoms

Blossom Shrinking:

Whenever a blossom is detected,
it is shrunk into a a single *pseudonode*,
which is an *outer* node.



Lemma: (Edmonds, 1965)

∃ augmenting path in G before shrinking	⇔	∃ augmenting path in G′ after shrinking

⚜ However, two key issues to handle

- detection of blossoms
- shrinking of blossoms
- expansion of blossoms

◎ Edmond's algorithm, 1965 O(n^4)

(Details in [PaSt82] and [Edmo65])

J. Edmonds, "Path, Trees and Flowers," Canadian J. Math, 17, (1965), pp. 449-467.

General Matching - History

⚜ Algorithms for General Matching

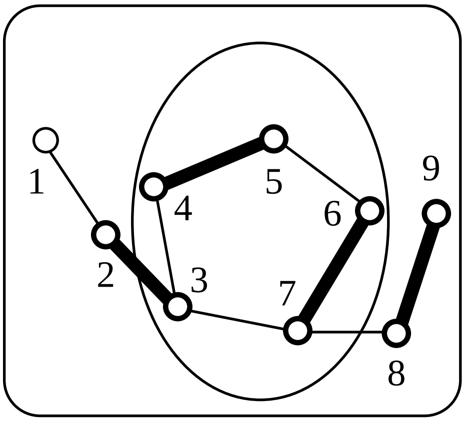
Year	Time	Authors	Remarks
1957		Berge,	Augmenting Path Theorem
1965	O(n^4)	Edmonds,	Blossom Shrinking
1965	O(mn^2)	Witzgall & Zahn	Modified Edmonds' alg
1967	O(n^3)	Balinski	
1974	O(mn)	Kameda & Munro	better blossom handling
1976 1976	O(n^3)	Gabow Lawler	Better blossom handling, no explicit shrinking/expansion
1980	O(n^3)	Pape & Conradt	simple blossom handling, FORTRAN code
1983	O(mn)	Gabow & Tarjan	
1975	O($n^{2.5}$)	Even & Kariv	not-practical, high storage extremely complicated
1980	O($\sqrt{n}\, m$)	Micali & Vazirani	theoretically fastest, not-pract.

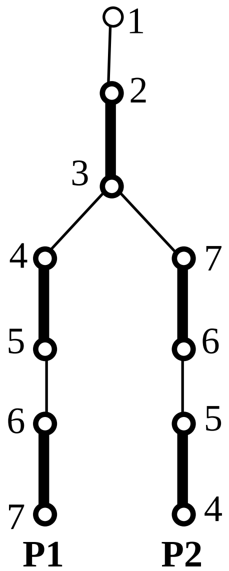
Pape and Conradt's Algorithm

⚜ Pape and Conradt's Implementation, 1980

Syslo, Deo, Kowalik, Prentice-Hall, 1983
Discrete Optimization Algorithms (Ch-3.7)

◎ Instead of shrinking a blossom,
it "grows" blossom in two alternating paths





Blossom {3,4,5,6,7} is grown in **two different alt-paths** . Node 5 is an inner node on path P1, and an outer node on path P2.

Similarly, each node in the blossom is **both** an inner and outer node .

◎ node v is on a blossom ⇔ v is ***both*** an outer node and an inner node in T

⚜ Greedy Initial Matching

Start with all vertices unmatched ;
for every exposed node $v \in V$ **do**
Try to match v with an unmatched vertex $w \in Adj(v)$;

Pape & Conradt's Algorithm (cont)

Implementation Details...

- Maintain the following data-structures
  - mate[v] : vertex matched with v, (= 0 if exposed)
  - Q : a queue of unexplored outer nodes in T
  - gf [v] : grandfather of node v in T, (used in back-tracing)
  - inner[v] : boolean (=1 if node v is a root or an inner node in T)

- Initialization of these data structures
  - mate[v] := 0 for all $v \in V$;
  - Q := ϕ ;
  - gf [v] := 0 for all $v \in V$;
  - inner[v] := false for all $v \in V$; inner[root] := true;

Pape & Conradt's Algorithm (cont)

Note: ∃ serious flaw in Alg 3-8(b) of [SDK83], p329.

• Use this pseudo-code instead, (and catch any bugs in it)

Algorithm Maximum Matching:

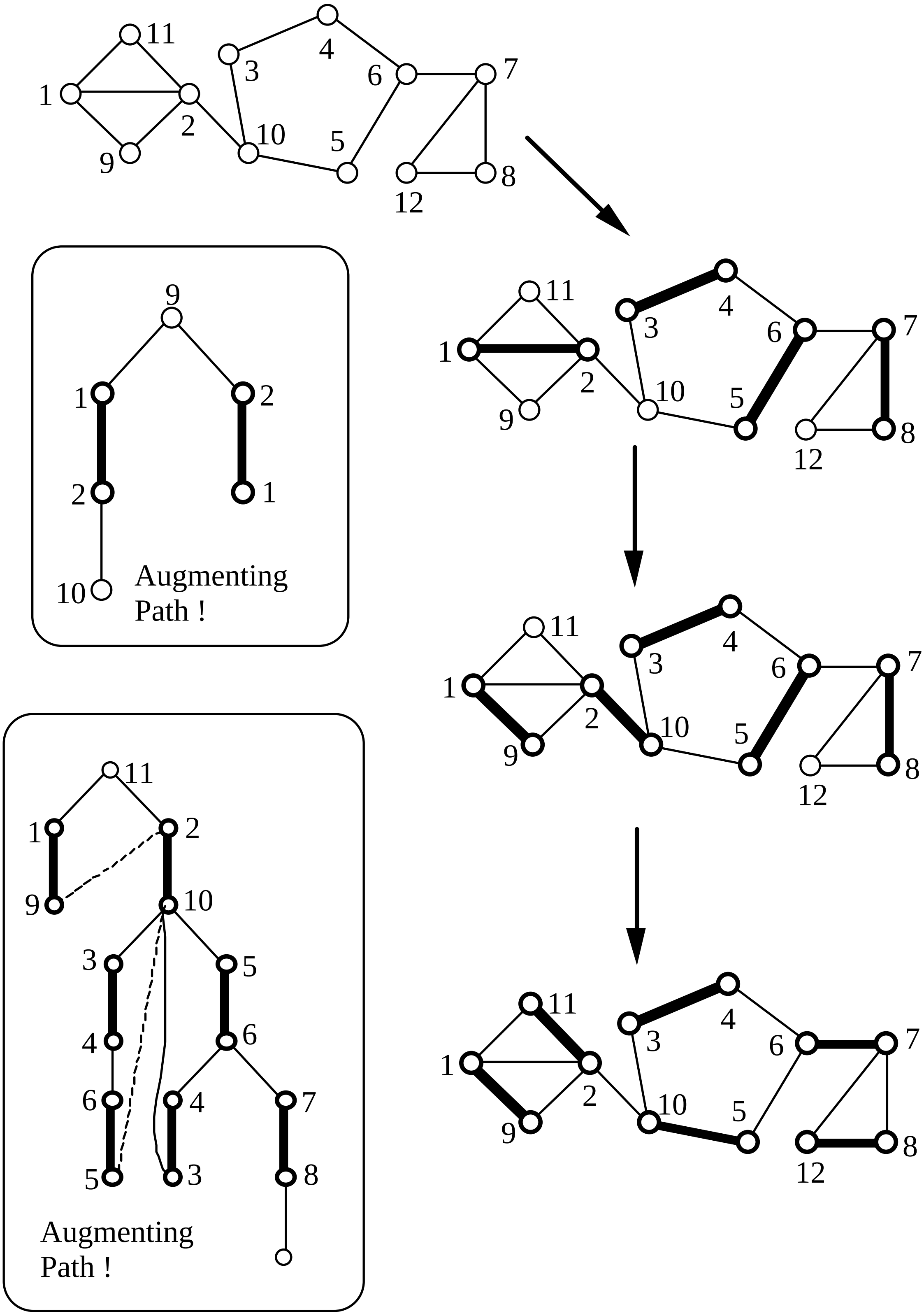
```
Start with initial matching ;
for every exposed r ∈ V do
    begin (* Grow alternating tree rooted at r *)
        for all v∈V do inner[v] := false ;     (* Init inner *)
        inner[r] := true ;
        Q := { r } ;
        while (Q ≠ ϕ) and (not found) do
            begin
                Delete x from Q ;
            for y∈Adj(x) do
                case 1 : (inner[y]=true)
                    (* Even cycle -- Ignore node y *)

                case 2 : (inner[y]=false) & (y is exposed)
                        Augmenting M ;
                        found := true ;

                case 3 : (inner[y]=false) and (y not ancestor x)
                    inner[y] := true; (* Grow Tree T; *)
                    gf[mate[y]] := x;
                    Insert mate[y] into Q ;

                case 4 : (inner[y]=false) and (y ancestor of x)
                    (* Blossom found -- Ignore node y *)
                endcase ;
        end ; {while (Q≠...}
    end; {for...}
```

Pape & Conradt's Algorithm - Example



Pape & Conradt's Algorithm - Analysis

- Initial Matching O(m)

- Growing Alternating Tree

	Total-cost
• Step 6,7,9,19. Queue Operations	O(n)
• Step 4,5. Initialize/Updating arrays inner, gf	O(n)
• Step 10. Checking (inner[y]=false)	O(m)
• **Case 1.** Step 11,12.	O(m+n)
• **Case 2.** Step 13,14,15.	
• **Case 3.** Step 16,17,18,19.	O(n)
• **Case 4.** Step 20,21.	O(n)

Observation:

> In each tree-growing phase,
> inner[v]=true at most *once*!

Homework: Complete the O(n^3) analysis!
[Read also [SDK83].