Minimum Vertex Cover

CSC3412 – Class Presentation

Spring 2007

Outline

- Problem Introduction
- Problem Complexity
- Problem Solving Methods
- Demonstration Program

Problem Introduction

- Let G = (V, E)

  A graph with vertex set V and edge set E.

- A vertex cover of G is a subset of V' $\subseteq$ V such that every edge in E is incident on some vertex in V'.

  Figure 1 shows an example of vertex cover.

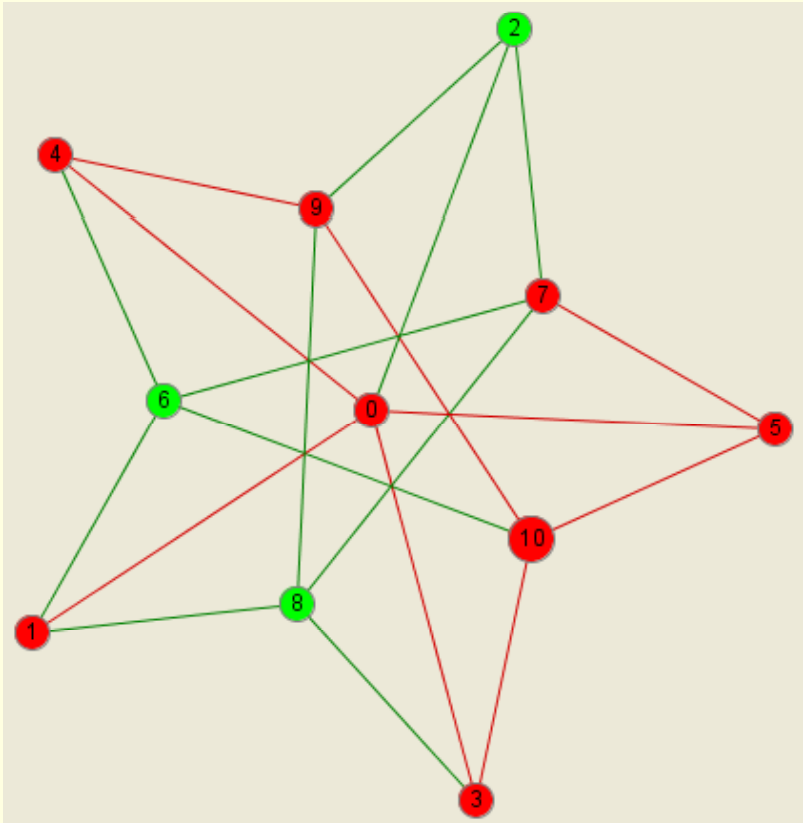


Figure 1: Red vertices set is a vertex cover of the graph

Problem Introduction (cont.)

- **Minimum vertex cover problem**

  The minimum vertex cover problem (in our goal) is to find a vertex cover of a given graph with the fewest number vertices.

  Figure 2 is a minimum vertex cover of the given graph

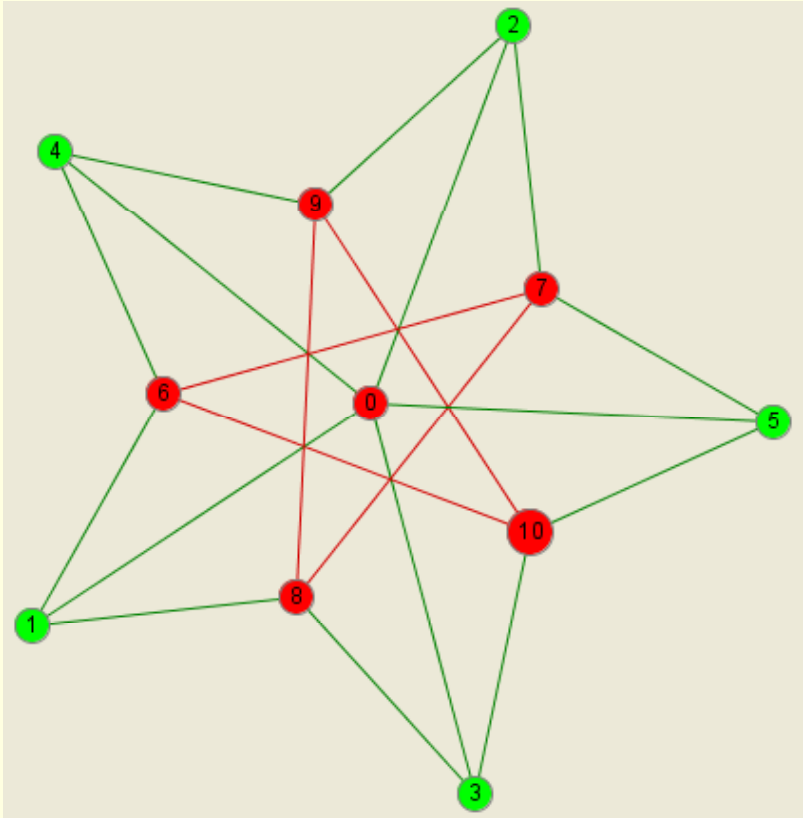


Figure 2: Red vertices set is a minimum vertex cover of the graph

Problem Complexity

The minimum vertex cover problem (VC) is a NP-Complete problem.

Proof

1) It is very easy to decide for a given subset *V'* of vertices, whether all edges are covered, i.e. whether *V'* is a vertex cover, by just iterating over all edges. This takes a polynomial running time.

Problem Complexity (cont.)

2) VC is Complete:

Based on the transformation between 3-SAT and VC shown in figure 3.

- The transformation takes a polynomial number of operations
- 3-SAT is satisfiable if and only if there exists a vertex cover V' of G with size $|V'| \leq K$.

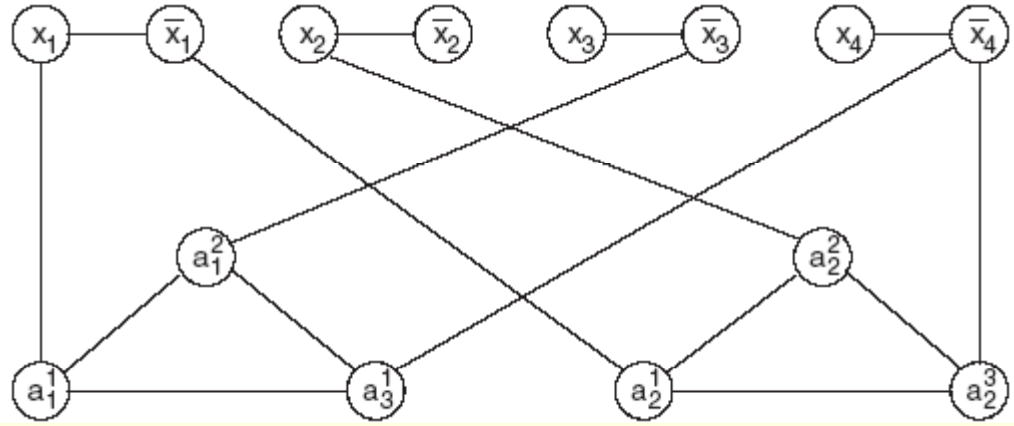


Figure 3: Transformation between 3-SAT & VC

Problem Solving Methods

- There is no complete algorithm with the complexity of polynomial-time for the problem
- There are two types of algorithm for the problem:
  - Complete algorithms with the complexity of exponential-time.
    - Guarantee to find the optimum or true solution.
    - The solution space is searched in principle completely.
  - Incomplete algorithms
    - Not ensured that the true solution or the global optimum is found.
    - Very often sufficient for practical applications.
    - The solution found by an incomplete algorithm is considered as an approximation of the real optimal solution.

Incomplete algorithms

- It is not ensured that the true solution or the global optimum is found. But they are very often sufficient for practical applications.
- Approximation algorithms
  - Definition: An algorithm A for an optimization problem Π is an approximation algorithm with approximation ratio $\rho(I)$ if, for any instance $I \in D_{\Pi}$, it finds a feasible solution $s \in S(I)$ such that:

$$OPT(I) / \rho(I) \leq cost(s) \leq \rho(I)\, OPT(I)$$

$\rho(I)$ is usually a $|I|$ based parameter.
A is a ρ-Approximation algorithm for the optimization problem.

Incomplete algorithms (cont.)

- VC-Greedy, a 2-Approximation algorithm for the VC

Algorithm: VC-Greedy

Input: An undirected graph G with minimum vertex cover size t + 1
Output: A vertex cover for G of size at most 2(t + 1)
begin
    cover = $\emptyset$
    while $E(G) \neq \emptyset$ do
        Choose arbitrarily an edge $(u, v) \in E(G)$
        Cover – cover $\cup$ {u, v}
        $E(G) = E(G) \setminus \{E(u) \cup E(v)\}$
    Output cover
end

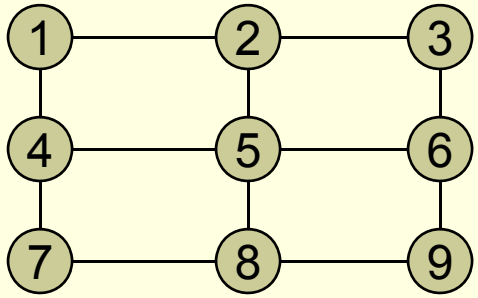
Incomplete algorithms (cont.)

- Min-cover, a generalized heuristic algorithm which provides a ρ-Approximation algorithm for the VC

```
Algorithm min-cover(G)
begin
        initialize V_vc = ∅;
        while there are uncovered edges do
        begin
                take one vertex i that best satisfies the heuristic;
                mark i as covered: V_vc = V_vc ∪ {i};
                remove all incident edges{i, j } from E;
                remove vertex i from V;
        end;
        return(V_vc);
end
```

Incomplete algorithms (cont.)

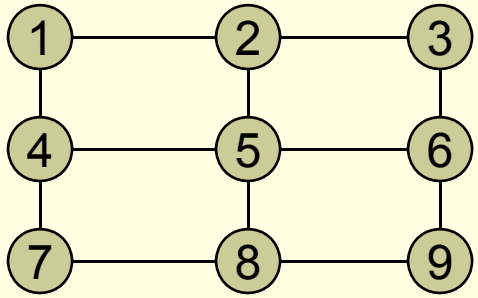
- Generalized heuristic: Vertices with high degree
  - d(5) = 4
  - According to the heuristic rule, vertex 5 is the best.
  - The only solution is {2,4,6,8} that does not contain vertex 5.

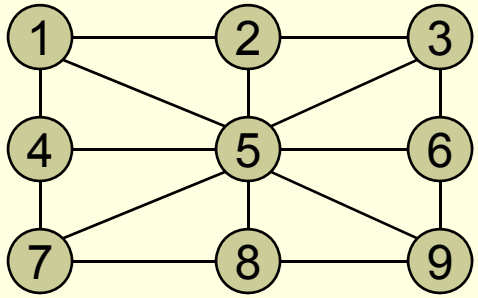


Incomplete algorithms (cont.)
A new proposed heuristic

- Idea: Instead of finding the covered vertices, we try to find the uncovered ones.
- Vertices are selected by their degrees in order from low to high:
  - A vertex of degree 0 will not be covered.
  - A vertex of degree 1 will not be covered but its neighbor.
  - A vertex of degree 2 is covered only if its both neighbors have degrees of 2.
  - A vertex of degree greater than 2 is not covered but all of its neighbors.
  - Once a vertex is covered, all its incident edges are removed.

- Using the proposed heuristic:
  - 1 is removed then 2, 4 are covered.
  - The degrees of 3,5,7 are reduced.
  - 3 is removed then 6 is covered and the degree of 9 is reduced
  - 7 is removed then 8 is covered.





Complete Algorithms

- They guarantee to find the exact answer, even if the time required will, in general, grow exponentially with the graph size.
- The naïve Algorithm: Since each vertex can be either covered or uncovered, the most direct approach is to enumerate all possible 2^N configuration, store all those being VCs, and finally select one of those having minimal VC cardinality. Obviously, the time-complexity of this approach is $O(2^N)$.

☞ The goal is to reduce the exponent value as small as possible.

Complete Algorithms (cont.) Significant Efforts [HW03]

- Tarjan & Trojanowski: Use divide-and-conquer approach and attain an $O(2^{N/2})$ time complexity.
  - Find all connected components of the graph.
  - Find the minimum-size vertex covers for each then combine them for the whole one MVC.
- Shindo M. & Tomita E. use the same approach and attain an $O(2^{N/2.863})$.

☞ Divide-and-conquer approach helps finding only one solution for the VC. In order to enumerate all solutions, branch-and-bound approaches must be applied.

Complete Algorithms (cont.) Branch and Bound approach

- It is similar to divide-and-conquer in branching but the concept of domination cannot be used, instead a bound for each branch specified to avoid a bump.
- It tries to build the full configuration tree
  - Try certain choices of putting covering marks.
  - Once no VC of desired size found, some marks removed and repositioned elsewhere (backtracking). When simpler bound is used: the algorithm continues to branch into sub-trees if covering marks available and no vertex cover found.
  - Being done in a systematic way allowing investigation of the full configuration space.
  - The usual running time $O(2^N)$ is reduced by omitting sub trees of the full tree by using a bound: trees where for sure no minimum-energy configuration is located can be omitted.

Branch and Bound Algorithm

- Divide he problem is divided into n branches then finds the minimum vertex cover from a VC called $C_i = V - \{v_i\}$ (i=1,2,…n)
- The minimum vertex cover C_i is found by omitting removable vertices until it reach to the size k
- If it's not found in all branches, for each pair of above vertex covers C_i, C_j, a VC called C_{ij} is created ($C_{ij} = C_i \cup C_j$) and processed in the same way
- The results of C_{ij} will be the minimum vertex covers of the problem

Branch and Bound Algorithm (cont.)

- The algorithm uses a parameter **k** as a bound value to limit the depth of search trees
- Search trees are smaller
- Search speed is improved
- Determine value of **k**: every graph with *n* vertices and maximum vertex degree Δ must have a minimum vertex cover of size at most

$$\mathbf{k = n - \lceil n/(\Delta + 1) \rceil}$$

Branch and Bound Algorithm (cont.)

Procedure A

Input: Graph G with n vertices, Vertex cover C of G

Proc_A(G, C) {
While the vertex cover C has removable vertices
For each removable vertex *v* of *C*
r(*C*-{*v*}) = the number of removable vertices of Vertex cover *C*-{*v*}.
v_{max} = v where r(*C*-{*v*}) is a maximum
C = *C*-{v_{max}}
Return C
}

Procedure B

Input: Graph G with n vertices, minimal vertex cover C of G

Proc_B(G, C, n) {
For i=1 to n
Find **v** $\in$ **C** such that v has exactly one neighbor **w** $\notin$ **C**
If v is found
$C^{v,w}$ = (C $\cup$ {w}) – {v}
C = **A** (G,$C^{v,w}$)
Return C
}

Branch and Bound Algorithm (cont.)

Input: Graph G with n vertices, value k **(size of vertex cover is at most k)**
The algorithm will stop

BranchAndBound(G, k) {

//Part I

For i = 1 to n

C_i = V-{i}

C_i = **Proc_A**(G,C_i)

For r = 1 to n-k

C_i = **Proc_B**(G, C_i, r)

//Part II

For each pair of C_i, C_j found in Part I

$C_{i,j} = C_i \cup C_j$

$C_{i,j}$ = **Proc_A**(G,$C_{i,j}$)

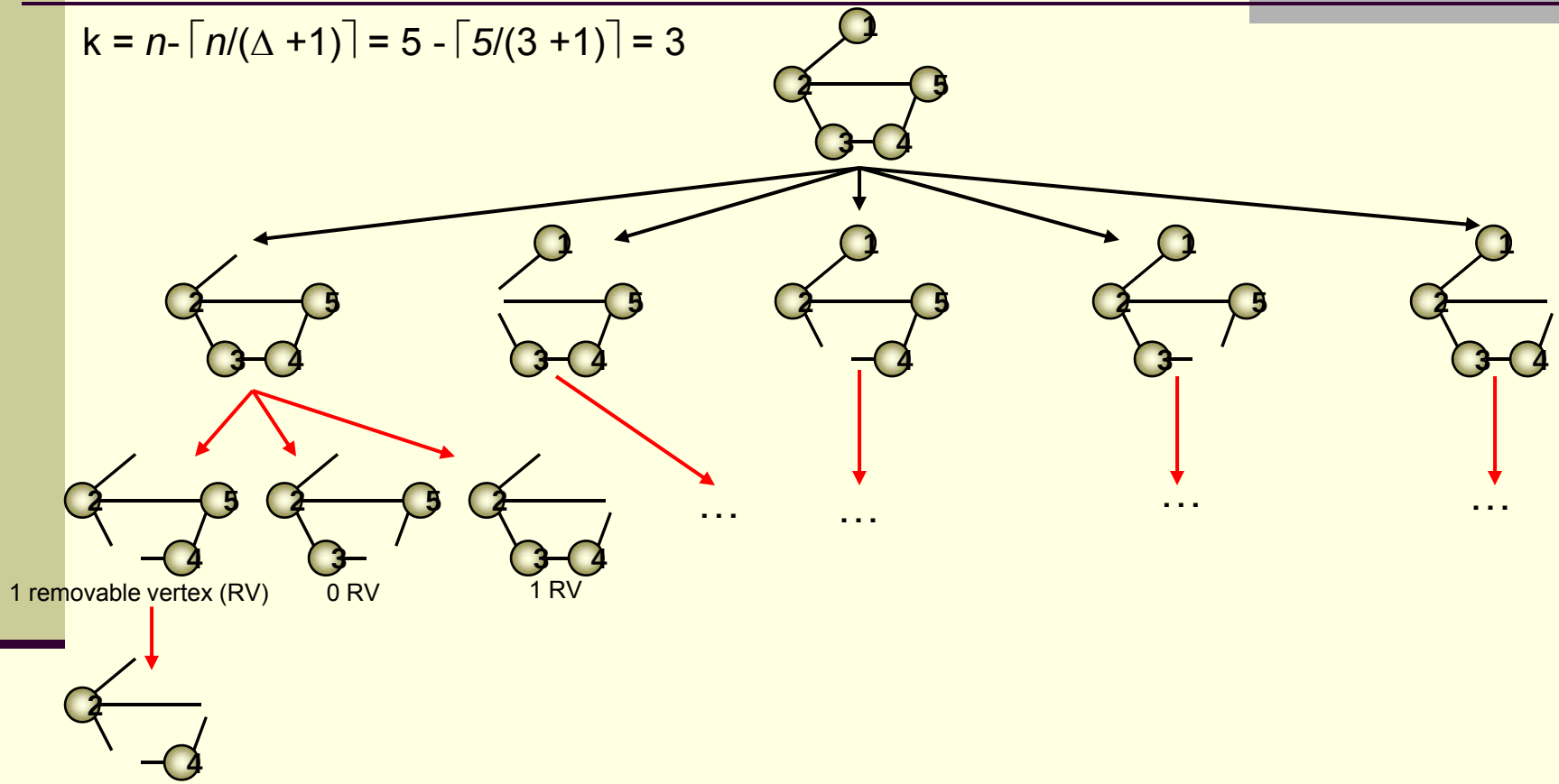
For r = 1 to n-k

$C_{i,j}$ = **Proc_B**(G, $C_{i,j}$, r)

}

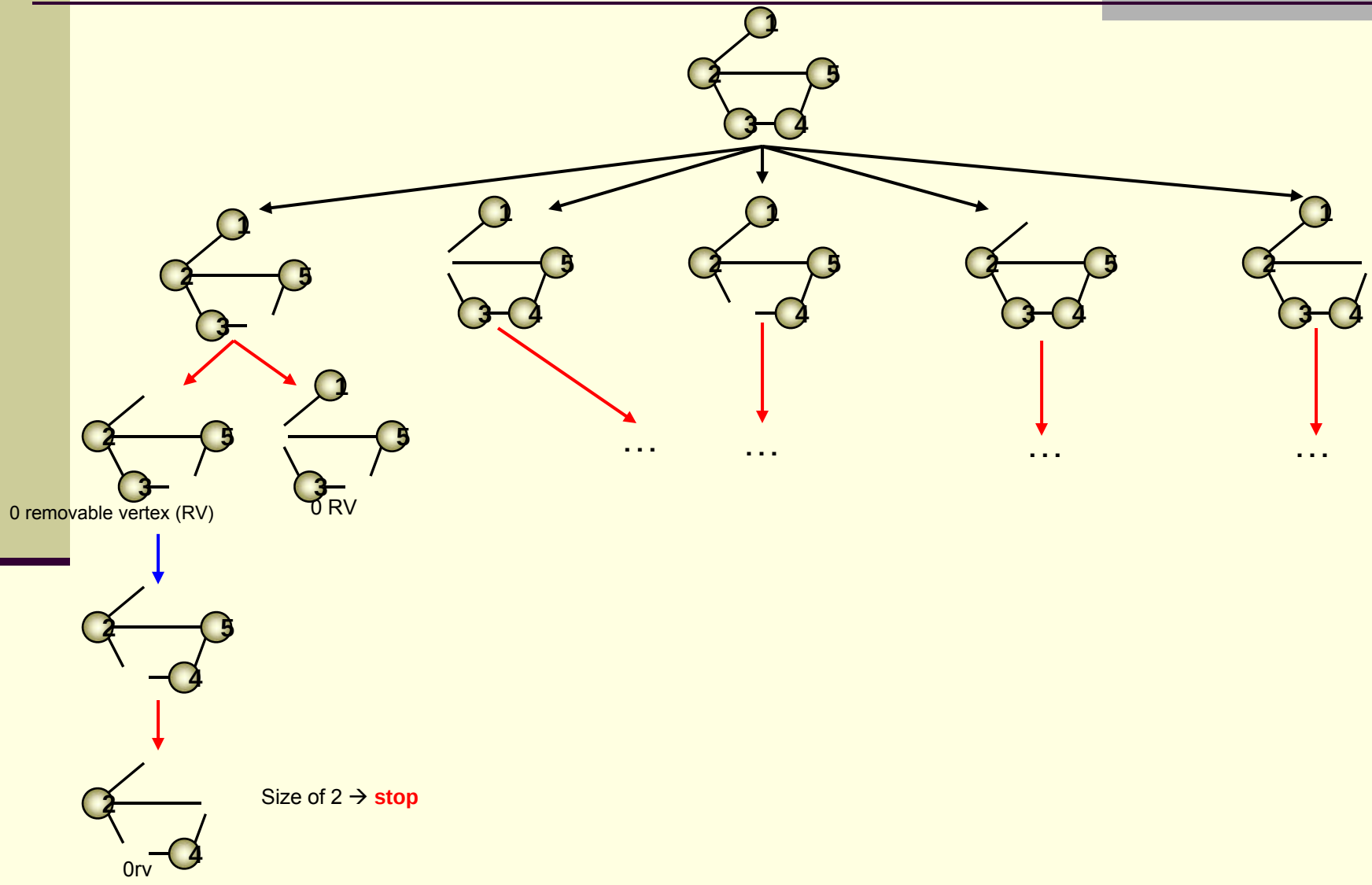
Branch and Bound Algorithm (cont.)

k = $n - \lceil n/(\Delta + 1) \rceil = 5 - \lceil 5/(3 + 1) \rceil = 3$

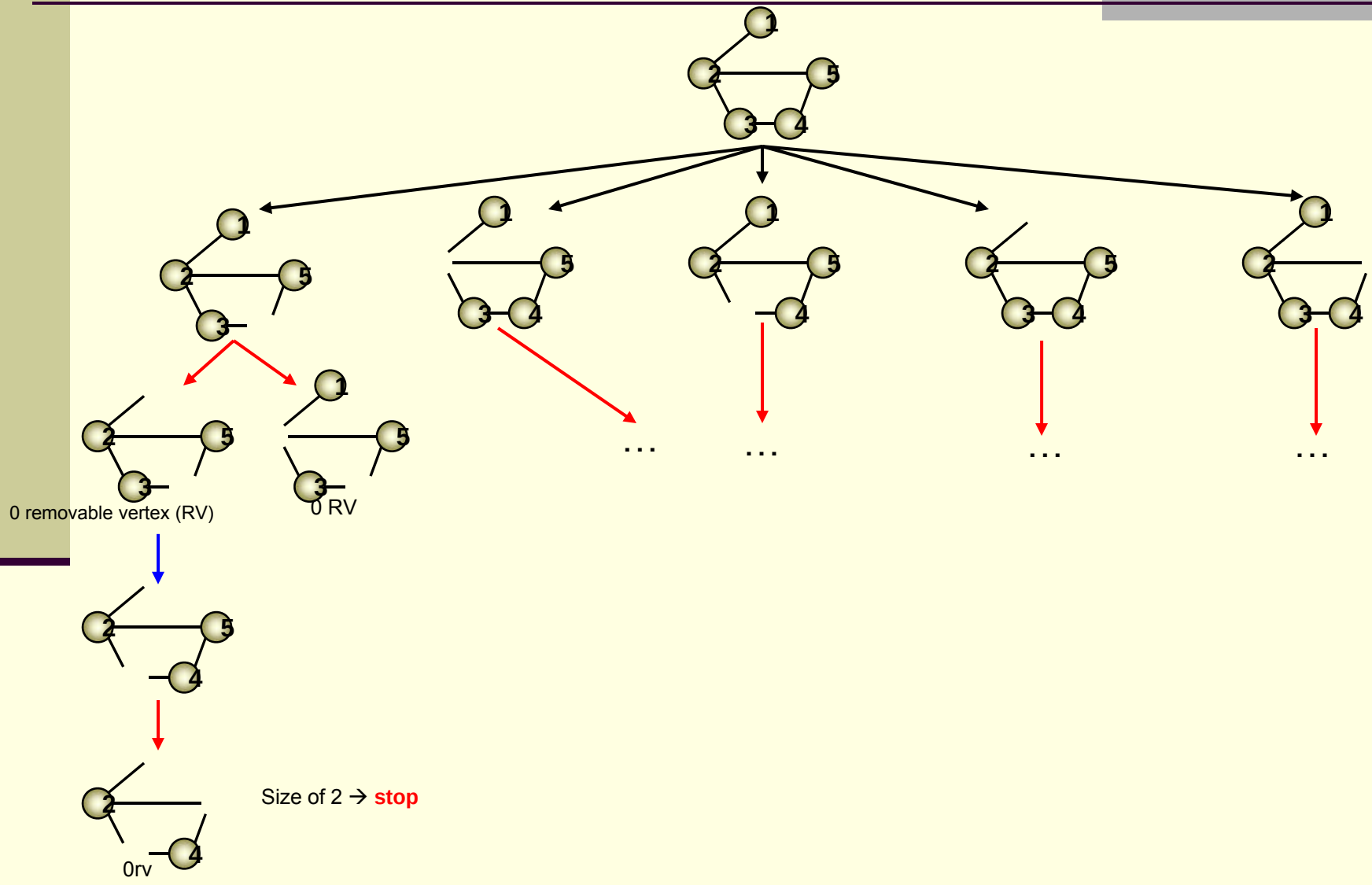


Size of 2 ≤ k → **stop**

Branch and Bound Algorithm (cont.)



Branch and Bound Algorithm (cont.)



Branch and Bound Algorithm (cont.)

Running time

- The algorithm terminates in polynomial-time
- For a graph with n vertices:
  - **Part I** takes at most $n^7+n^6+n^4+n^2$
  - **Part II** takes at most $n^7+n^8+n^5+n^3$

➔Running time: $O(n^8+2n^7+n^6+n^5+n^4+n^3+n^{2)}$

Demonstration !!!

References

- [HW03] Hartmann A., Weigt M. *Statistical mechanics of the vertex-cover problem*, Journal of Physics A: Mathematical And General 2003.