Chapter 8 DOMINATING SETS

Distributed Computing Group

Mobile Computing

Summer 2004

Overview

- Motivation
- Dominating Set
- Connected Dominating Set

- The "Greedy" Algorithm
- The "Tree Growing" Algorithm
- The "Marking" Algorithm
- The "k-Local" Algorithm
- The "Dominator!" Algorithm
- The "Largest ID" Algorithm

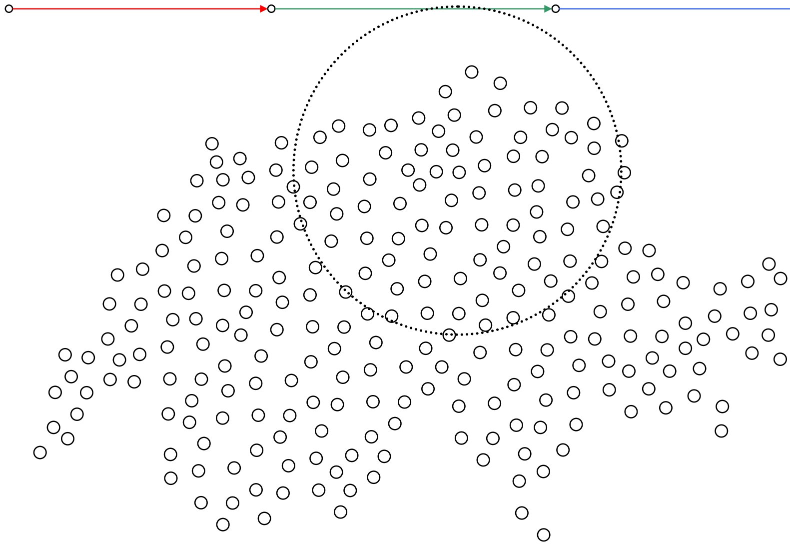
Discussion

- We have seen: 10 Tricks → 2^{10} routing algorithms
- In reality there are almost that many!

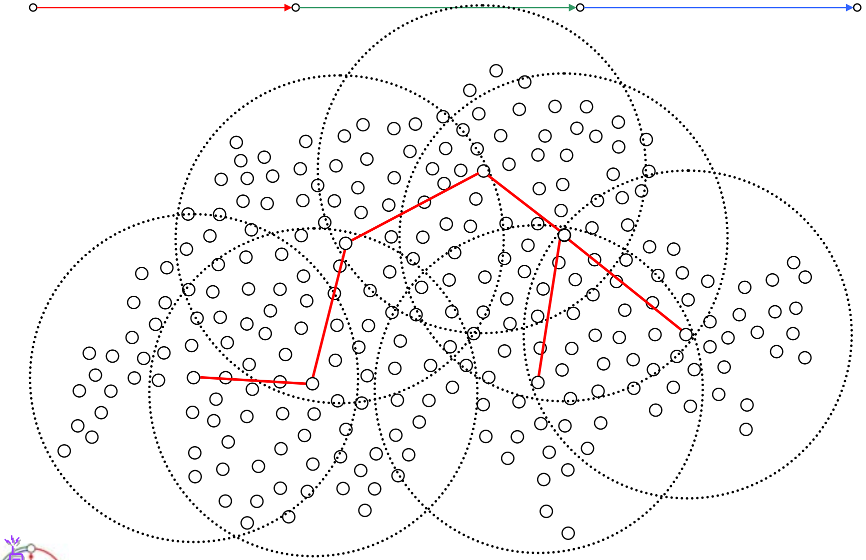
- Q: How good are these routing algorithms?!? Any hard results?
- A: Almost none! Method-of-choice is simulation…
- Perkins: "if you simulate three times, you get three different results"

- Flooding is key component of (many) proposed algorithms, including most prominent ones (AODV, DSR)
- At least flooding should be efficient

Finding a Destination by Flooding



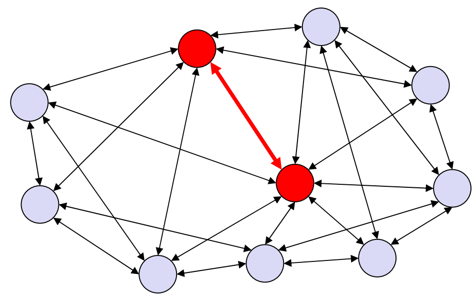
Finding a Destination *Efficiently*



Backbone

- Idea: Some nodes become backbone nodes (gateways). Each node can access and be accessed by at least one backbone node.

- Routing:
1. If source is not a gateway, transmit message to gateway
2. Gateway acts as proxy source and routes message on backbone to gateway of destination.
3. Transmission gateway to destination.



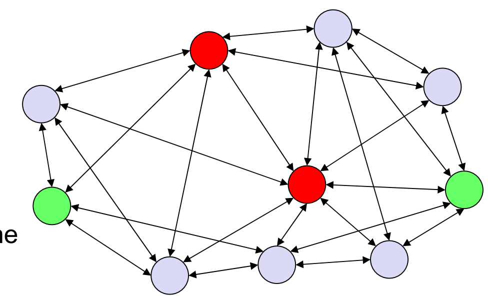
(Connected) Dominating Set

- A Dominating Set DS is a subset of nodes such that each node is either in DS or has a neighbor in DS.

- A Connected Dominating Set CDS is a connected DS, that is, there is a path between any two nodes in CDS that does not use nodes that are not in CDS.

- A CDS is a good choice for a backbone.

- It might be favorable to have few nodes in the CDS. This is known as the Minimum CDS problem



Formal Problem Definition: M(C)DS

- Input: We are given an (arbitrary) undirected graph.

- Output: Find a Minimum (Connected) Dominating Set, that is, a (C)DS with a minimum number of nodes.

- Problems
  - M(C)DS is NP-hard
  - Find a (C)DS that is “close” to minimum (approximation)
  - The solution must be local (global solutions are impractical for mobile ad-hoc network) – topology of graph “far away” should not influence decision who belongs to (C)DS

Greedy Algorithm for Dominating Sets

- Idea: Greedy choose “good” nodes into the dominating set.

- Black nodes are in the DS
- Grey nodes are neighbors of nodes in the CDS
- White nodes are not yet dominated, initially all nodes are white.

- Algorithm: Greedily choose a node that colors most white nodes.

- One can show that this gives a log Δ approximation, if Δ is the maximum node degree of the graph. (The proof is similar to the “Tree Growing” proof on 6/14ff.)
- One can also show that there is no polynomial algorithm with better performance unless P$\approx$NP.

CDS: The “too simple tree growing” algorithm

- Idea: start with the root, and then greedily choose a neighbor of the tree that dominates as many as possible new nodes

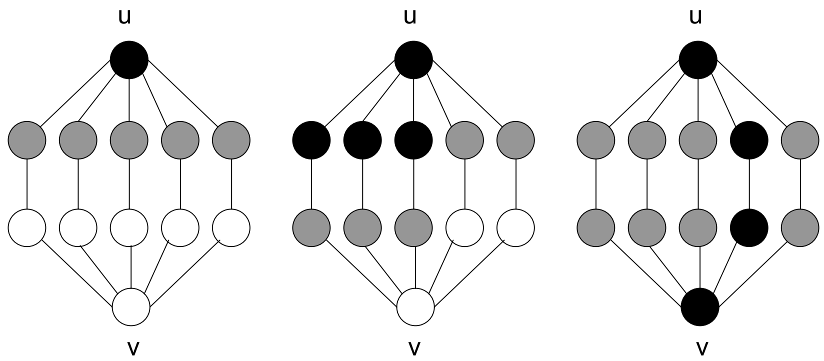
- Black nodes are in the CDS
- Grey nodes are neighbors of nodes in the CDS
- White nodes are not yet dominated, initially all nodes are white.

- Start: Choose the node a maximum degree, and make it the root of the CDS, that is, color it black (and its white neighbors grey).

- Step: Choose a grey node with a maximum number of white neighbors and color it black (and its white neighbors grey).

Example of the “too simple tree growing” algorithm

Graph with 2n+2 nodes; tree growing: |CDS|=n+2; Minimum |CDS|=4

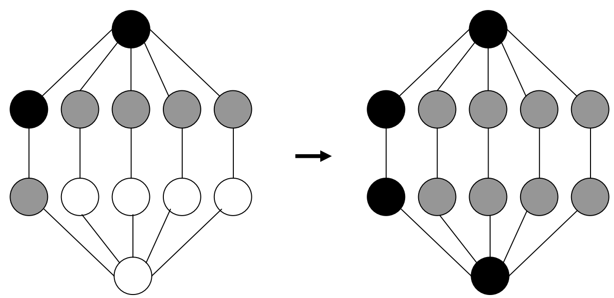


tree growing: start … Minimum CDS

Tree Growing Algorithm

- Idea: Don’t scan one but two nodes!

- Alternative step: Choose a grey node and its white neighbor node with a maximum sum of white neighbors and color both black (and their white neighbors grey).



Analysis of the tree growing algorithm

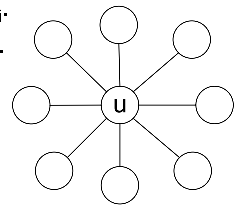
- Theorem: The tree growing algorithm finds a connected set of size $|CDS| \leq 2(1+H(\Delta)) \cdot |DS_{OPT}|$.
- DS_{OPT} is a (not connected) minimum dominating set
- Δ is the maximum node degree in the graph
- H is the harmonic function with $H(n) \approx \log(n)+0.7$
- In other words, the connected dominating set of the tree growing algorithm is at most a $O(\log(\Delta))$ factor worse than an optimum minimum dominating set (which is NP-hard to compute).
- With a lower bound argument (reduction to set cover) one can show that a better approximation factor is impossible, unless $P \approx NP$.

Proof Sketch

- The proof is done with amortized analysis.
- Let S_u be the set of nodes dominated by $u \in DS_{OPT}$, or u itself. If a node is dominated by more than one node, we put it in one of the sets.
- We charge the nodes in the graph for each node we color black. In particular we charge all the newly colored grey nodes. Since we color a node grey at most once, it is charged at most once.
- We show that the total charge on the vertices in an S_u is at most $2(1+H(\Delta))$, for any u.

Charge on S_u

- Initially $|S_u| = u_0$.
- Whenever we color some nodes of S_u, we call this a step.
- The number of white nodes in S_u after step i is u_i.
- After step k there are no more white nodes in S_u.



- In the first step $u_0 - u_1$ nodes are colored (grey or black). Each vertex gets a charge of at most $2/(u_0 - u_1)$.
- After the first step, node u becomes eligible to be colored (as part of a pair with one of the grey nodes in S_u). If u is not chosen in step i (with a potential to paint u_i nodes grey), then we have found a better (pair of) node. That is, the charge to any of the new grey nodes in step i in S_u is at most $2/u_i$.

Adding up the charges in S_u

$$C \leq \frac{2}{u_0 - u_1}(u_0 - u_1) + \sum_{i=1}^{k-1} \frac{2}{u_i}(u_i - u_{i+1})$$

$$= 2 + 2\sum_{i=1}^{k-1} \frac{u_i - u_{i+1}}{u_i}$$

$$\leq 2 + 2\sum_{i=1}^{k-1} \big(H(u_i) - H(u_{i+1})\big)$$

$$= 2+2(H(u_1)-H(u_k)) = 2(1+H(u_1)) = 2(1+H(\Delta))$$

Discussion of the tree growing algorithm

- We have an extremely simple algorithm that is asymptotically optimal unless P≈NP. And even the constants are small.
- Are we happy?
- Not really. How do we implement this algorithm in a real mobile network? How do we figure out where the best grey/white pair of nodes is? How slow is this algorithm in a distributed setting?
- We need a fully distributed algorithm. Nodes should only consider local information.

The Marking Algorithm

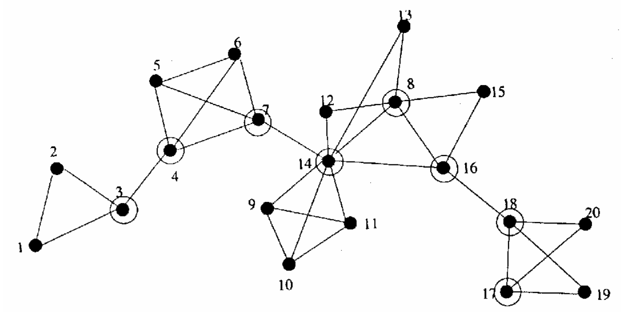
- Idea: The connected dominating set CDS consists of the nodes that have two neighbors that are not neighboring.

1. Each node u compiles the set of neighbors N(u)
2. Each node u transmits N(u), and receives N(v) from all its neighbors
3. If node u has two neighbors v,w and w is not in N(v) (and since the graph is undirected v is not in N(w)), then u marks itself being in the set CDS.

\+ Completely local; only exchange N(u) with all neighbors
\+ Each node sends only 1 message, and receives at most Δ
\+ Messages have size $O(\Delta)$

- Is the marking algorithm really producing a connected dominating set? How good is the set?

Example for the Marking Algorithm



[J. Wu]

Correctness of Marking Algorithm

- We assume that the input graph G is connected but not complete.
- Note: If G was complete then constructing a CDS would not make sense. Note that in a complete graph, no node would be marked.
- We show:

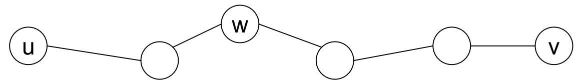
The set of marked nodes CDS is
a) a dominating set
b) connected
c) a shortest path in G between two nodes of the CDS is in CDS

Proof of a) dominating set

- Proof: Assume for the sake of contradiction that node u is a node that is not in the dominating set, and also not dominated. Since no neighbor of u is in the dominating set, the nodes $N^+(u) := u \cup N(u)$ form:
- a complete graph
  - if there are two nodes in N(u) that are not connected, u must be in the dominating set by definition
- no node $v \in N(u)$ has a neighbor outside N(u)
  - or, also by definition, the node v is in the dominating set
- Since the graph G is connected it only consists of the complete graph $N^+(u)$. We precluded this in the assumptions, therefore we have a contradiction

Proof of b) connected, c) shortest path in CDS

- Proof: Let p be any shortest path between the two nodes u and v, with $u,v \in$ CDS.
- Assume for the sake of contradiction that there is a node w on this shortest path that is not in the connected dominating set.



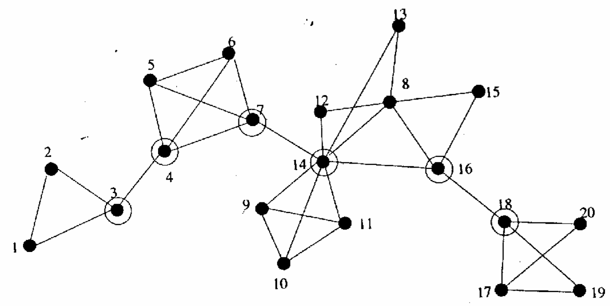
- Then the two neighbors of w must be connected, which gives us a shorter path. This is a contradiction.

Improving the Marking Algorithm

- We give each node u a unique id(u).
- Rule 1: If $N^+(v) \subseteq N^+(u)$ and id(v) < id(u), then do not include node v into the CDS.
- Rule 2: Let $u,w \in N(v)$. If $N(v) \subseteq N(u) \cup N(w)$ and id(v) < id(u) and id(v) < id(w), then do not include v into the CDS.
- (Rule 2+: You can do the same with more than 2 covering neighbors, but it gets a little more intricate.)
- …for a quiet minute: Why are the identifiers necessary?

Example for improved Marking Algorithm

- Node 17 is removed with rule 1
- Node 8 is removed with rule 2



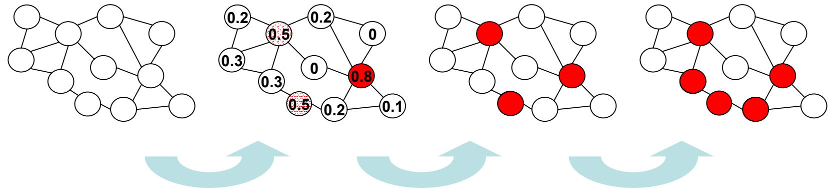
Quality of the Marking Algorithm

- Given an Euclidean chain of n homogeneous nodes
- The transmission range of each node is such that it is connected to the k left and right neighbors, the id's of the nodes are ascending.

- An optimal algorithm (and also the tree growing algorithm) puts every k'th node into the CDS. Thus $|CDS_{OPT}| \approx n/k$; with $k = n/c$ for some positive constant c we have $|CDS_{OPT}| = O(1)$.
- The marking algorithm (also the improved version) does mark all the nodes (except the k leftmost ones). Thus $|CDS_{Marking}| = n - k$; with $k = n/c$ we have $|CDS_{Marking}| = \Omega(n)$.
- The worst-case quality of the marking algorithm is worst-case! ☺

The k-local Algorithm

Input: Local Graph | Fractional Dominating Set | Dominating Set | Connected Dominating Set



Phase A: Distributed linear program
rel. high degree gives high value

Phase B: Probabilistic algorithm

Phase C: Connect DS by "tree" of "bridges"

Result of the k-local Algorithm

- Distributed Approximation

$$\boxed{\text{Theorem: } E[|DS|] \leq O(\alpha \ln \Delta \cdot |DS_{OPT}|)}$$

- The value of α depends on the number of rounds *k* (the locality)

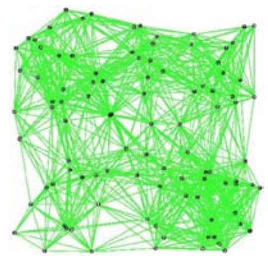
$$\boxed{\alpha \leq (\Delta + 1)^{5/\sqrt{k}}}$$

- The analysis is rather intricate… ☺

Unit Disk Graph

- We are given a set *V* of nodes in the plane (points with coordinates).
- The unit disk graph *UDG*(*V*) is defined as an undirected graph (with *E* being a set of undirected edges). There is an edge between two nodes *u*,*v* iff the Euclidian distance between *u* and *v* is at most 1.
- Think of the unit distance as the maximum transmission range.

- We assume that the unit disk graph *UDG* is connected (that is, there is a path between each pair of nodes)
- The unit disk graph has many edges.
- Can we drop some edges in the *UDG* to reduced complexity and interference?

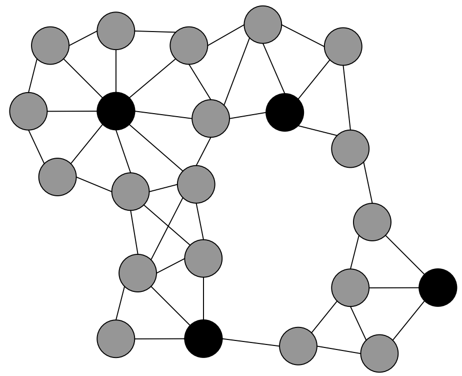


The “Dominator!” Algorithm

- For the important special case of Euclidean Unit Disk Graphs there is a simple marking algorithm that does the job.
- We make the simplifying assumptions that MAC layer issues are resolved: Two nodes u,v within transmission range 1 receive both all their transmissions. There is no interference, that is, the transmissions are locally always completely ordered.
- Initially no node is in the connected dominating set CDS.

1. If a node u has not yet received an “I AM A DOMINATOR, BABY!” message from any other node, node u will transmit “I AM A DOMINATOR, BABY!”
2. If node v receives a message “I AM A DOMINATOR, BABY!” from node u, then node v is dominated by node v.

Example



- This gives a dominating set. But it is not connected.

The “Dominator!” Algorithm Continued

3. If a node w is dominated by more two dominators u and v, and node w has not yet received a message “I am dominated by u and v”, then node w transmits “I am dominated by u and v” and enters the CDS.

- And since this is still not quite enough…

4. If a neighboring pair of nodes w_1 and w_2 is dominated by dominators u and v, respectively, and have not yet received a message “I am dominated by u and v”, or “We are dominated by u and v”, then nodes w_1 and w_2 both transmit “We are dominated by u and v” and enter the CDS.

“Dominator Algorithm”: Results

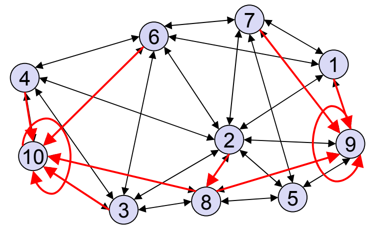
- The “Dominator!” Algorithm produces a connected dominating set.
- The algorithm is completely local. (is it?)
- Each node only has to transmit one or two messages of constant size.
- The connected dominating set is asymptotically optimal, that is, $|CDS| = O(|CDS_{OPT}|)$.
- Routes on backbone (CDS) are only by a constant factor longer than on UDG.

“Dominator Algorithm”: Remarks

- “Dominator” algorithm seems to be very local.
- If two neighbors want to join the DS simultaneously, we have a problem → synchronization between nodes is a problem!
- Algorithm actually calculates a maximal independent set (MIS).
- When taking care of all synchronization problems, best known MIS algorithm needs time O(log n).
- Lower Bound for general graphs: $\Omega\left(\sqrt{\frac{\log n}{\log \log n}}\right)$
- If you want to know more, visit PODC course!

The “Largest-ID” Algorithm

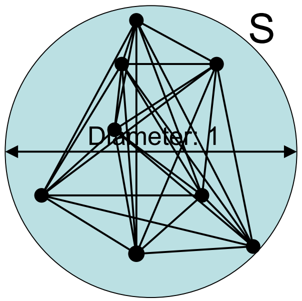
- All nodes have unique IDs
- Algorithm for each node:
  1. Send ID to all neighbors
  2. Tell node with largest ID in neighborhood that it has to join the DS
- Algorithm computes a DS in 2 rounds (extremely local!)



“Largest ID” Algorithm, Analysis I

- Assume, node IDs are at random, graph is UDG.
- We look at a disk S of diameter 1:

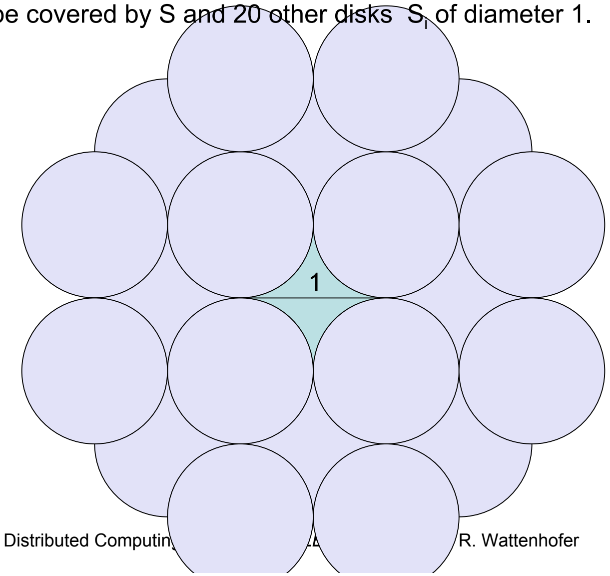
Nodes inside S have distance at most 1.
→ they form a clique



How many nodes in S are selected for the DS?

“Largest ID” Algorithm, Analysis II

- Nodes which select nodes in S are in disk of radius 3/2 which can be covered by S and 20 other disks S_i of diameter 1.



“Largest ID” Algorithm: Analysis III

- How many nodes in S are chosen by nodes in a disk S_i?
- x = # of nodes in S, y = # of nodes in S_i:
- A node u∈S is only chosen by a node in S_i if $\mathrm{ID}(u) > \max_{v \in S_i}\{\mathrm{ID}(v)\}$ (all nodes in S_i see each other).
- The probability for this is: $\frac{1}{1+y}$
- Therefore, the expected number of nodes in S chosen by nodes in S_i is at most:

$$\min\left\{y, \frac{x}{1+y}\right\}$$

Because at most y nodes in S_i can choose nodes in S
and because of linearity of expectation.

“Largest ID” Algorithm, Analysis IV

- From x≤n and y≤n, it follows that: $\min\left\{y, \frac{x}{1+y}\right\} \leq \sqrt{n}$
- Hence, in expectation the DS contains at most $20\sqrt{n}$ nodes per disk with diameter 1.
- An optimal algorithm needs to choose at least 1 node in the disk with radius 1 around any node.
- This disk can be covered by a constant (9) number of disks of diameter 1.
- The algorithm chooses at most $O(\sqrt{n})$ times more disks than an optimal one

“Largest ID” Algorithm, Remarks

- For typical settings, the “Largest ID” algorithm produces very good dominating sets (also for non-UDGs)
- There are UDGs where the “Largest ID” algorithm computes an $\Theta(\sqrt{n})$-approximation (analysis is tight).
- If nodes know the distances to each other, there is a iterative variant of the “Largest ID” algorithm which computes a constant approximation in O(loglog n) time.

Overview of (C)DS Algorithms

Algorithm	Worst-Case Guarantees	Local (Distributed)	General Graphs	CDS
Greedy	Yes, optimal unless P=NP	No	Yes	No
Tree Growing	Yes, optimal unless P=NP	No	Yes	Yes
Marking	No	Yes (const.)	Yes	Yes
k-local	Yes, but with add. factor α	Yes (k-local)	Yes	Yes
“Dominator!”	Asymptotically Optimal	Yes (log n)	No	Yes
“Largest ID” simple / iter.	$O(\sqrt{n})$ / constant	Yes (const / loglog n)	No	Yes