Critical-Trunk Based Obstacle-Avoiding Rectilinear Steiner Tree Routings for Delay and Slack Optimization

ISPD-2009

Yen-Hung Lin, Shu-Hsin Chang, Yih-Lang Li

Dept. of CSIE, National Chiao Tung University

OUTLINE

- Introduction
- Critical-Trunk Based Tree Growth
  - Performance-Driven OARST
  - Slack-Driven OARST
- Experimental Results
- Conclusions

INTRODUCTION

- OARST: Obstacle-Avoiding Rectilinear Steiner Tree
- Conventional OARST:
  - To minimize total wirelength
  - Maze-routing based manner
  - Spanning-graph based manner
    - Global view about pins and obstacles
    - Non-intersecting property
- Minimization of wirelength may worsen the performance.
- Objectives:
  - To construct an OARST and consider delay simultaneously
  - To adopt routing algorithm in spanning-graph based manner

RELATION BETWEEN RADIUS AND DELAY

(a) (b)

Figure 1. Relation between radius and delay in Shortest Path Tree (SPT)

(a) (b)

Figure 2. Relation between radius and delay in Minimal Steiner Tree (MST)

CRITICAL-TRUNK BASED TREE GROWTH

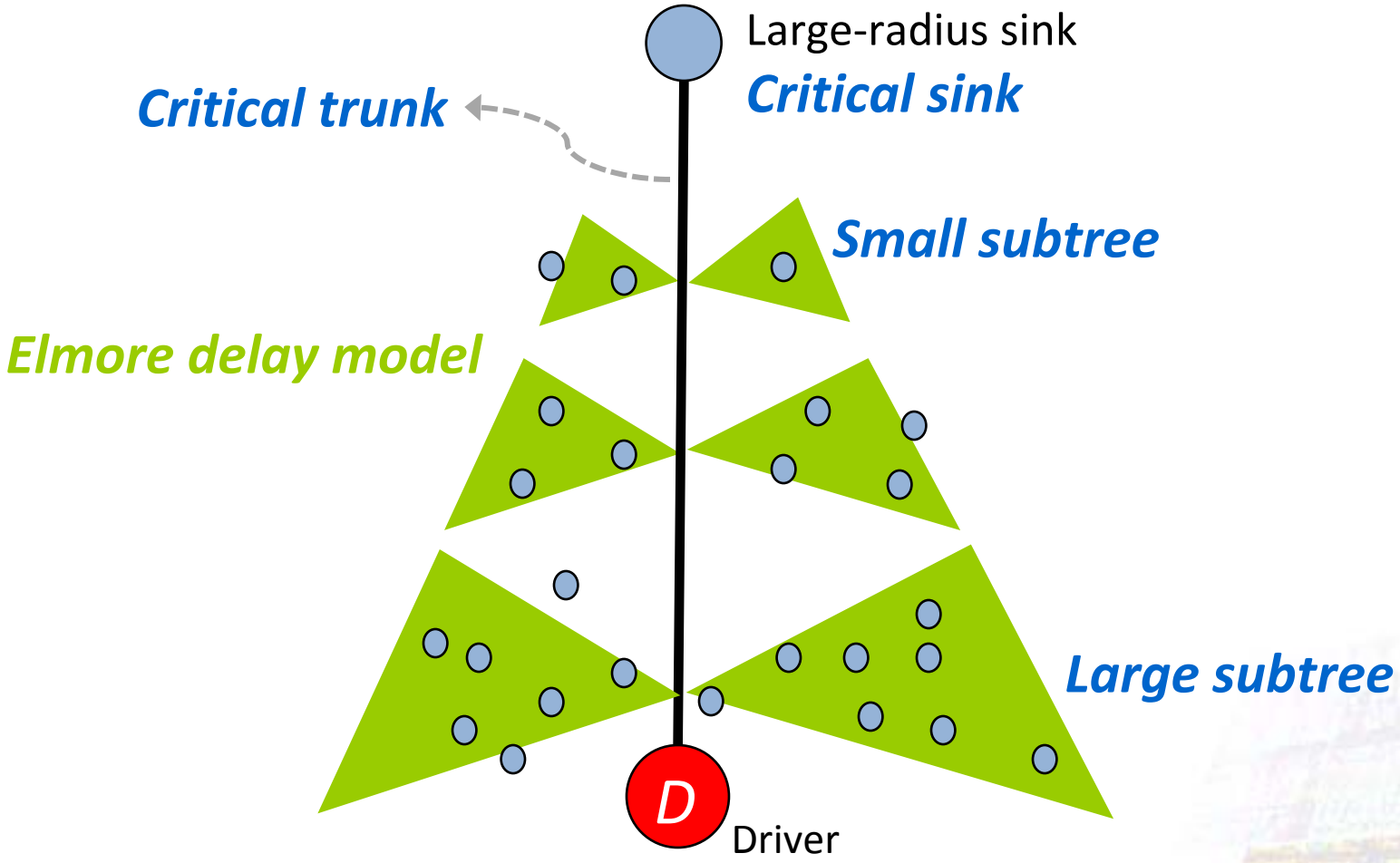
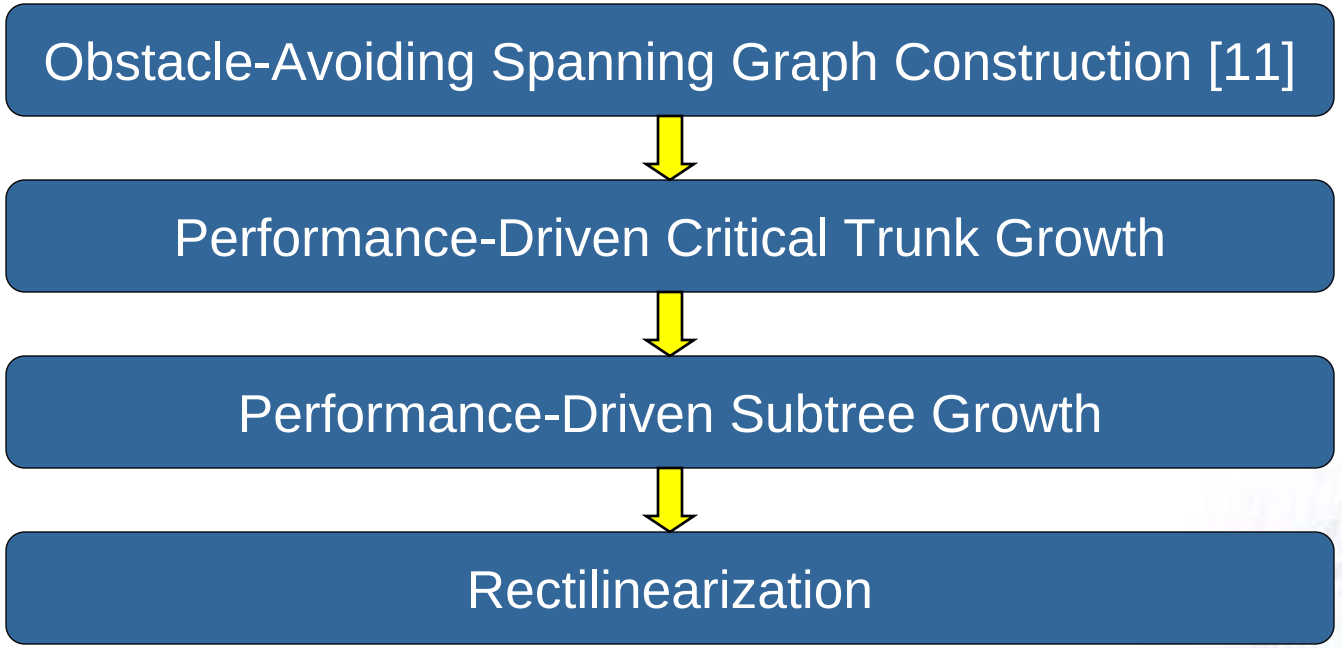


Figure 3. Ideal tree topology for critical-trunk based tree growth

PERFORMANCE-DRIVEN OARST

- To minimize the maximum sink's delay
- Overall flow of PDOARST:
  - 

[11] J. Long, H. Zhou, and S. O. Memik, "An O(*n*log*n*) edge-based algorithm for obstacle-avoiding rectilinear Steiner tree construction," *Proceedings of Intl. Symposium on Physical Design*, pp. 126-133, 2008.

PERFORMANCE-DRIVEN CRITICAL TRUNK GROWTH

- 2-pin net generation for routing algorithms
- Multi-source single-target maze routing

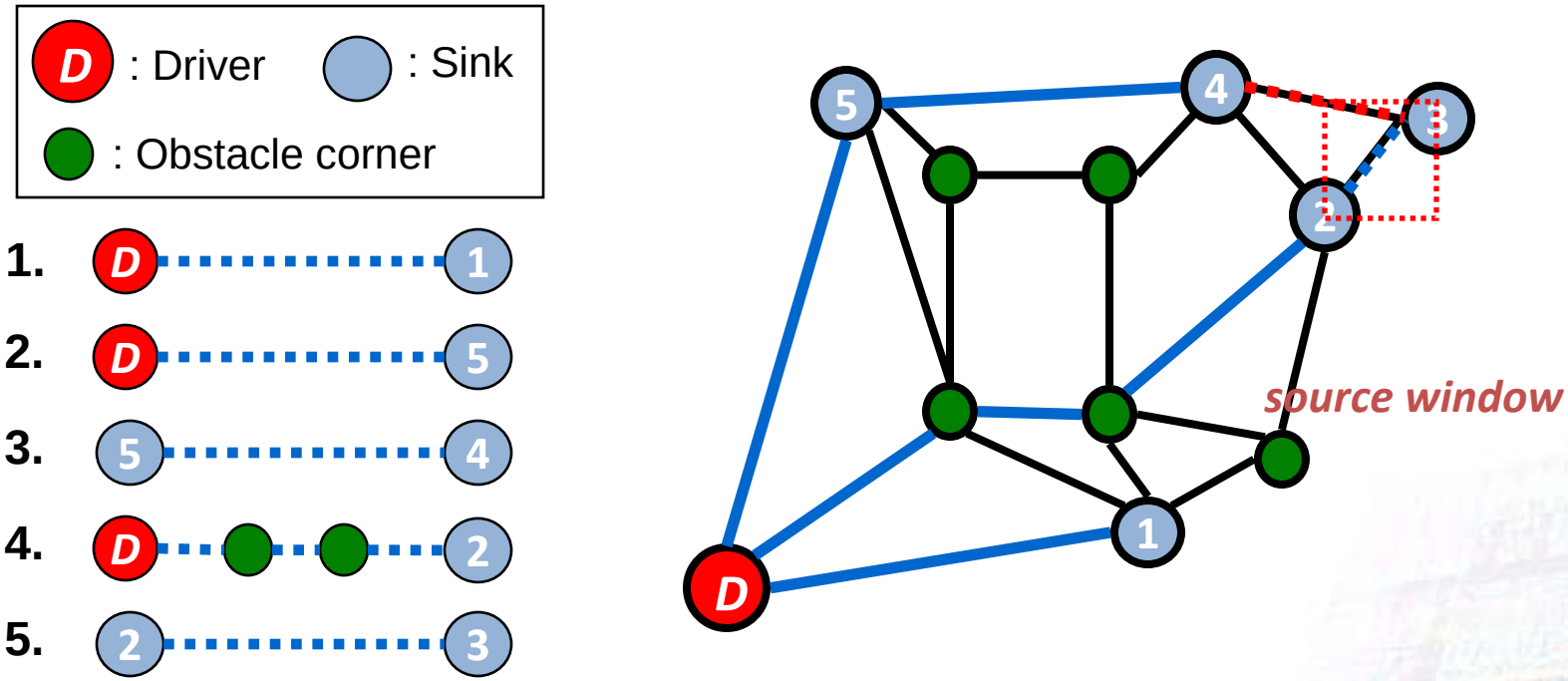


Figure 4. Performance-driven critical trunk growth

PERFORMANCE-DRIVEN CRITICAL TRUNK GROWTH

- Identification of performance-driven critical trunks
  - PDCTF: Performance-driven criticality threshold factor
    - $PDCTF = \dfrac{\text{average sink delay}}{\text{worst sink delay}}$
  - PDCR: Performance-driven Critical Radius
    - $PDCR = PDCTF \times \max(radius)$
  - A sink is critical if its radius exceeds PDCR.

PERFORMANCE-DRIVEN SUBTREE GROWTH

- Delay penalty factor (PDF)
  - To make the tree topology similar to the ideal one.
  - $$DPF(i) = \begin{cases} \dfrac{R_i}{R_{\max}}, i \in N_{cr} \\ 0, otherwise \end{cases}$$

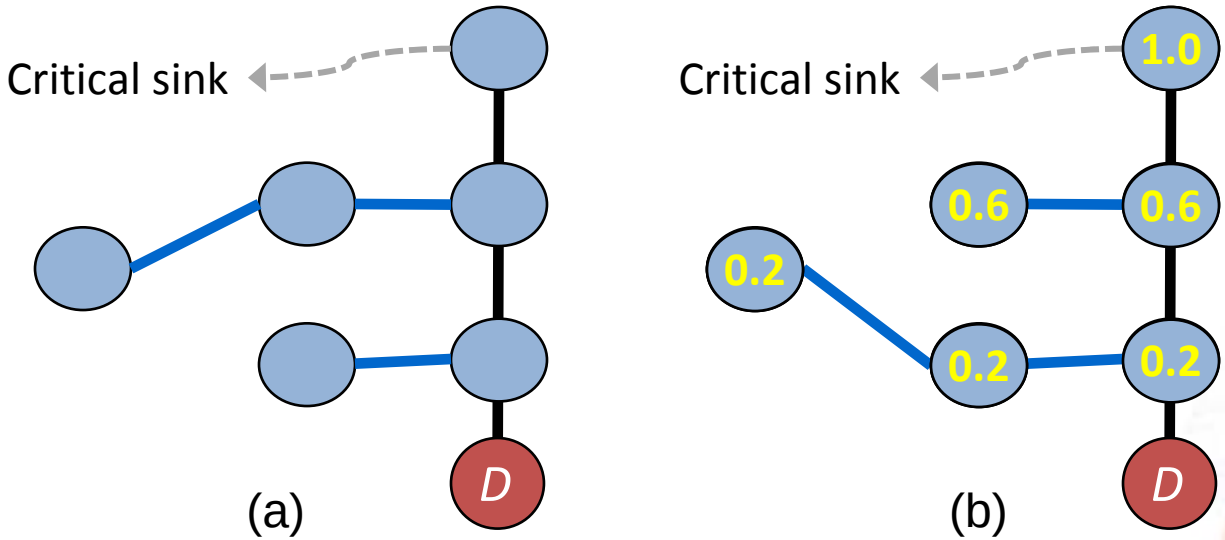
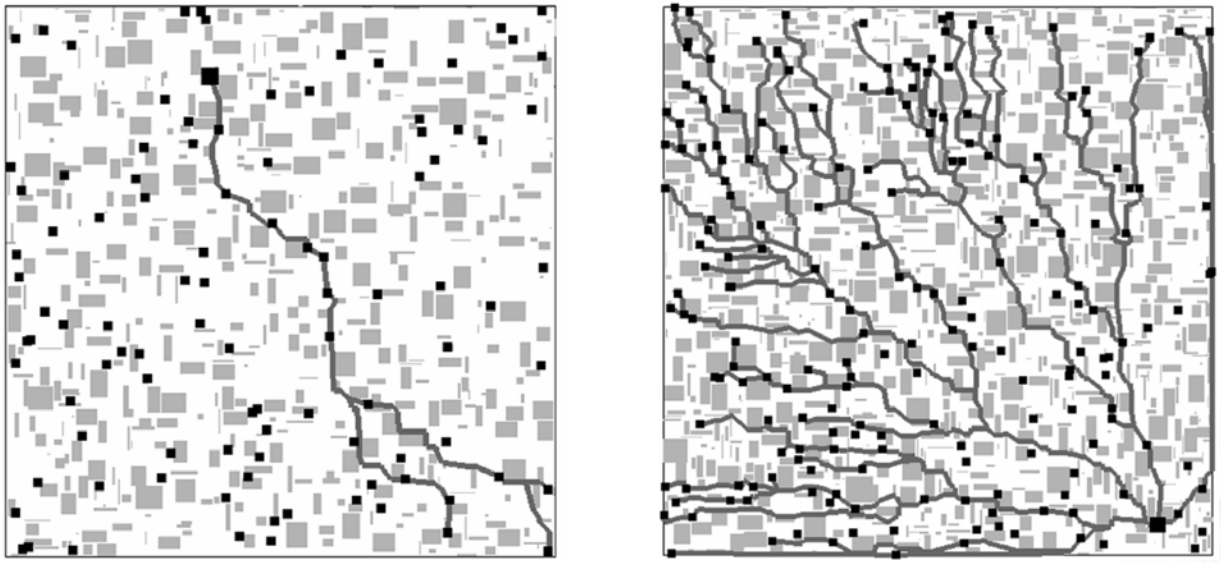


Figure 5. DPF and DPF inheritance

PERFORMANCE-DRIVEN SUBTREE GROWTH

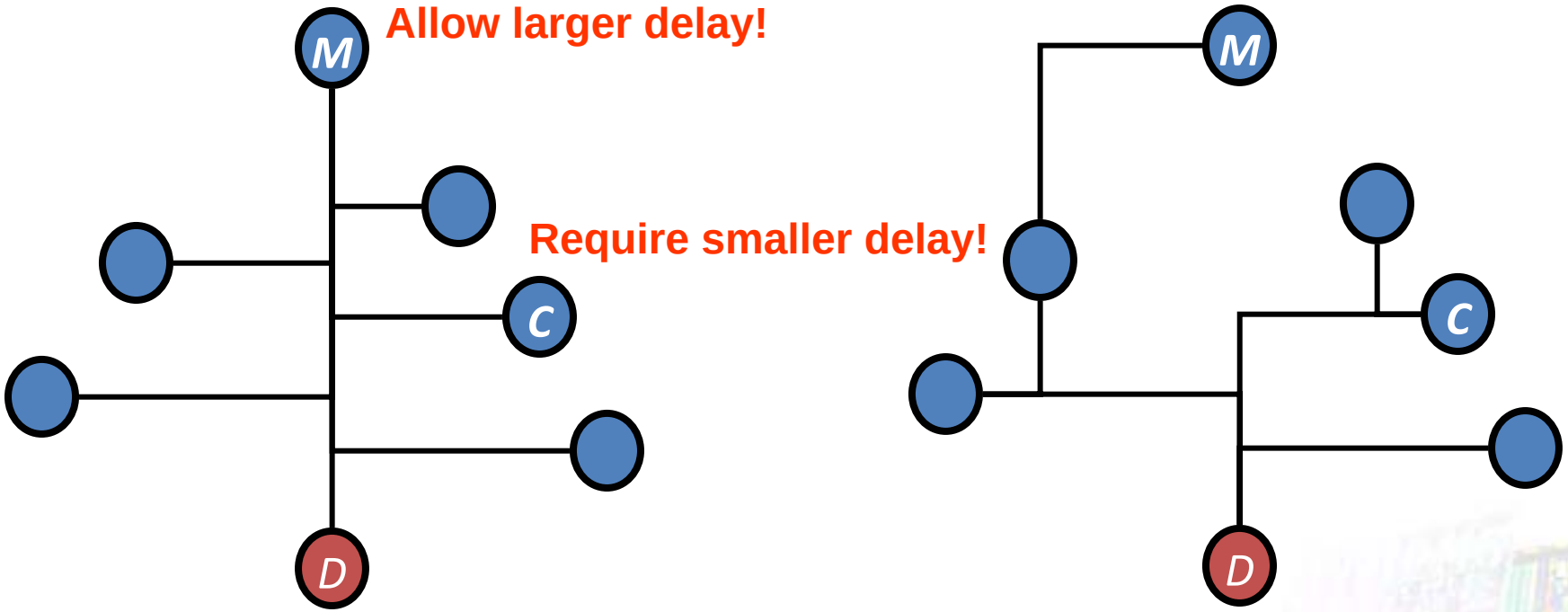
- A* search like function
  - $f(x) = g(x) + h(x)$
  - $g(x) = dist_{sx} + dist_{ds} \times DPF(s)^2 \times (1 - PDCTF)$



(a) PDCTF=0.854 (b) PDCTF=0.473

Figure 6. The relation between performance-driven critical trunk & PDCTF

PERFORMANCE-DRIVEN vs. SLACK-DRIVEN

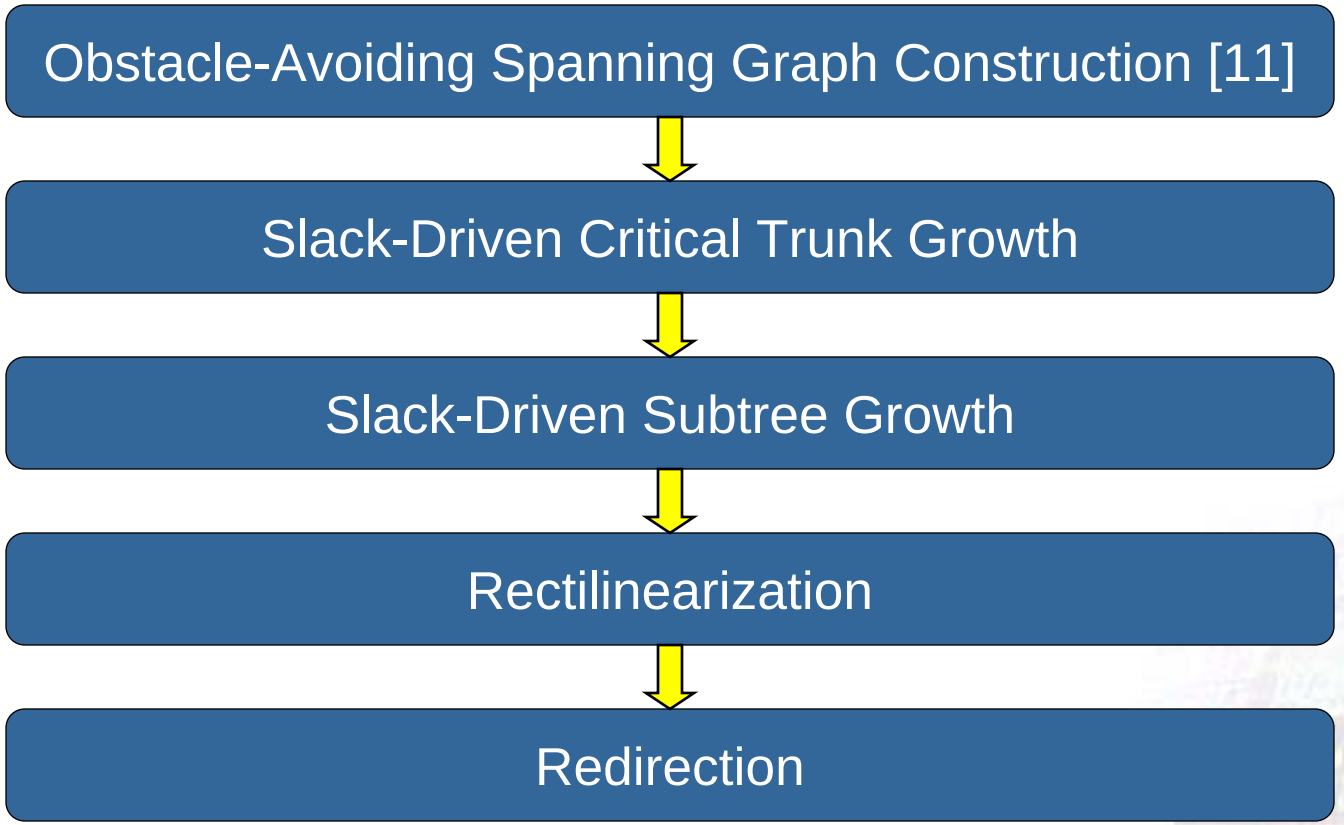


(a) Minimizing the maximum delay (b) Satisfying the timing constraint

Figure 7. Steiner tree with different objective

SLACK-DRIVEN OARST

- To maximize the worst negative slack (WNS)
- Overall flow of SDOARST:



SLACK-DRIVEN CRITICAL TRUNK GROWTH

- Sinks with smaller slacks prefer small delays.
- To guide 2-pin net generation with slack

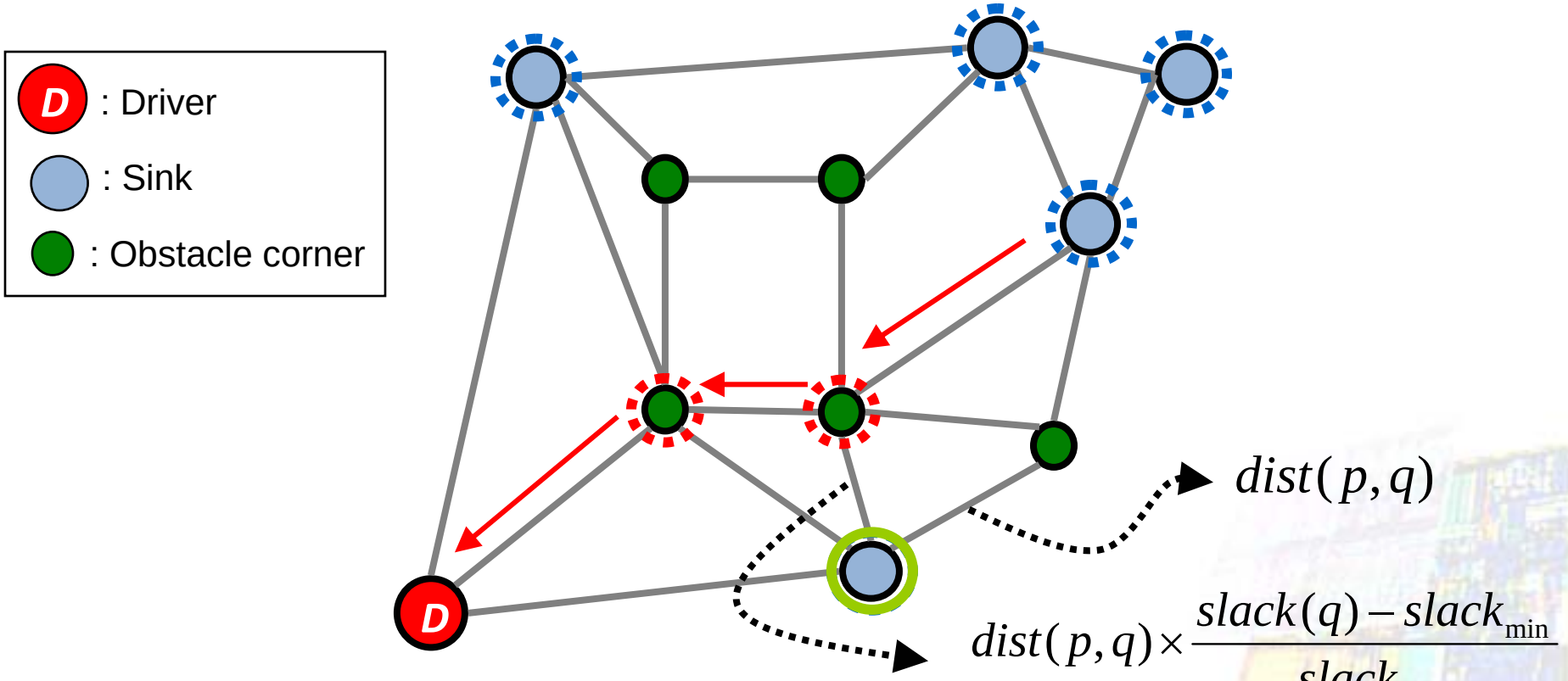


Figure 8. Slack-driven critical trunk growth

SLACK-DRIVEN CRITICAL TRUNK GROWTH

- Identification of slack-driven critical trunks
  - To compute *priority* of each sink
    - $priority(i) = slack(i) - delay(i)$
    - Small priority means that the attached sink has higher possibility to violate timing constraints.
  - SDCP: Slack-Driven Critical Priority
    - Average priority of all sinks

SLACK-DRIVEN SUBTREE GROWTH

- Slack determines principally the allowable delay of a sink.
- Single-source single-target maze routing

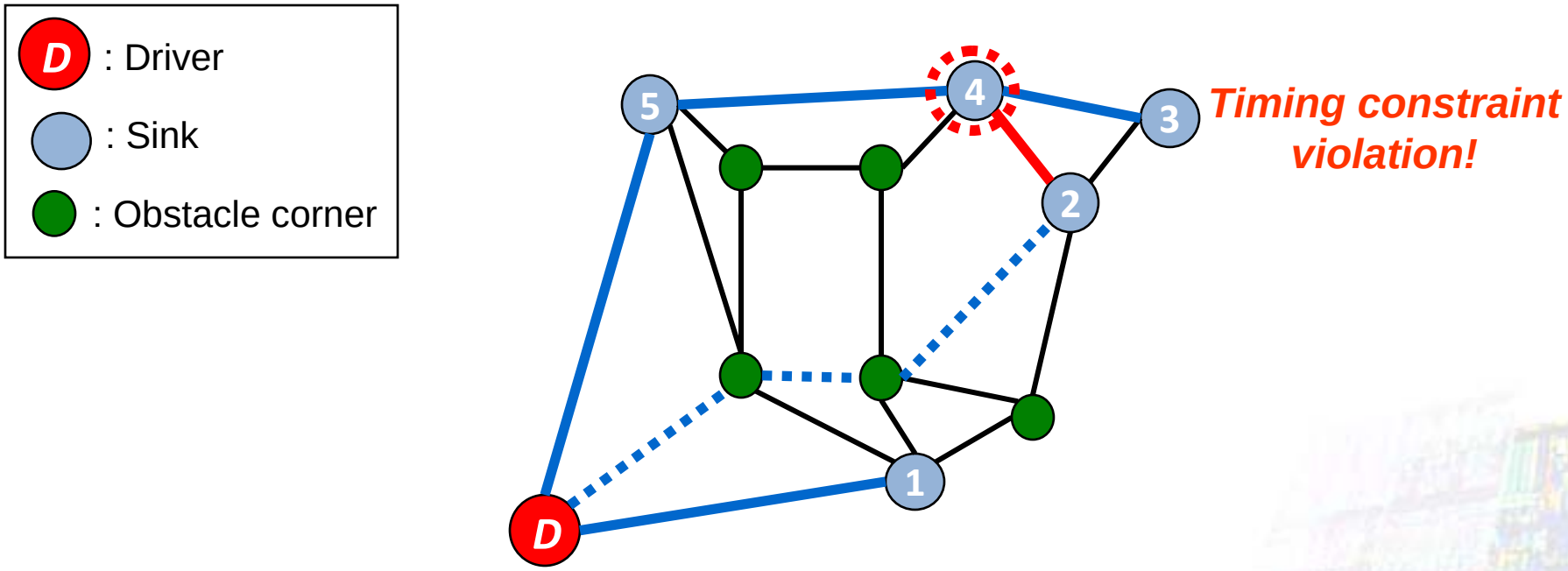


Figure 9. Slack-driven subtree growth

REDIRECTION

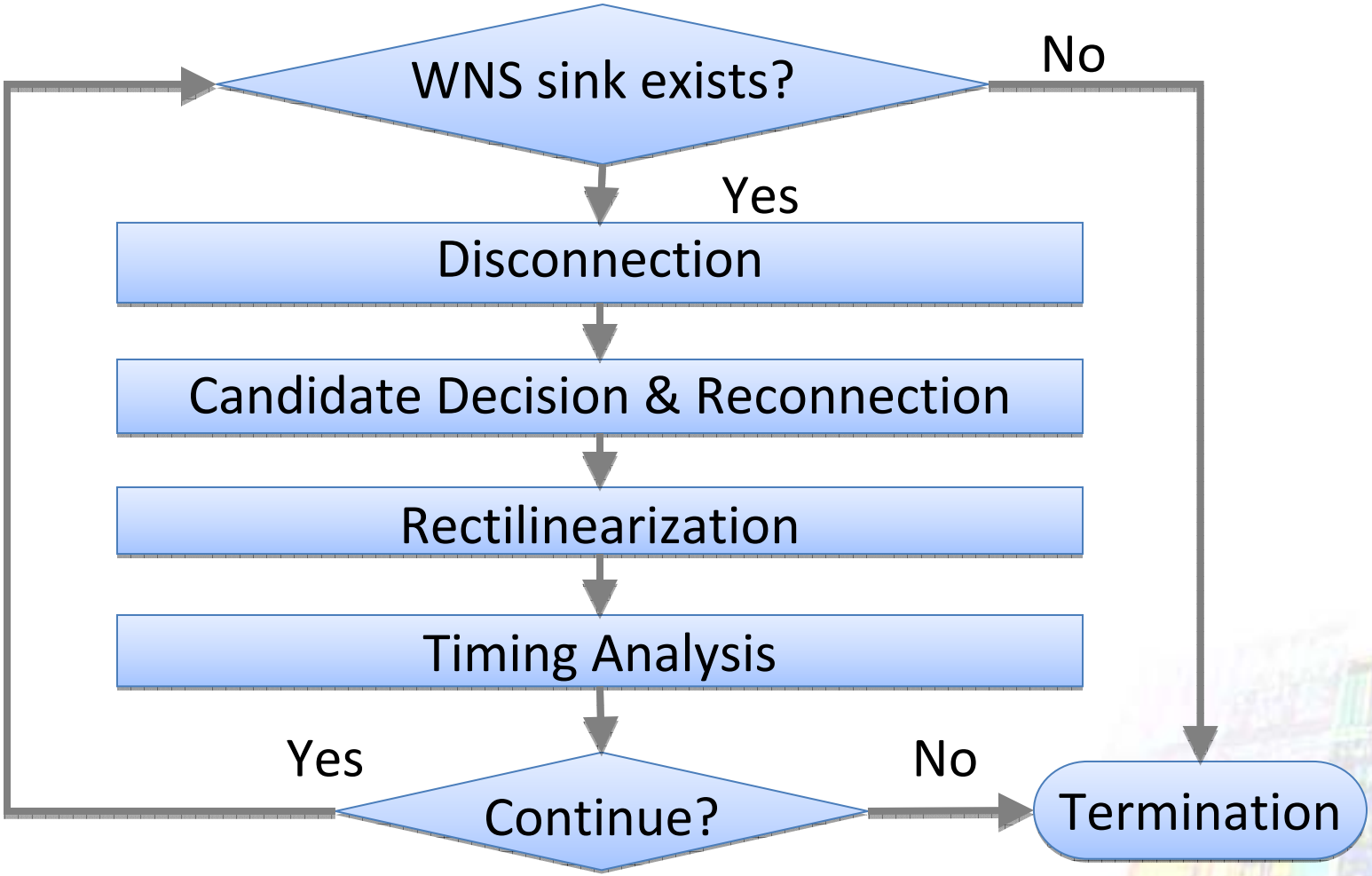


Figure 10. Flow of redirection mechanism

EXPERIMENTAL RESULTS

- Platforms
  1. A PC with 2.1 GHz AMD Athlon 64 Dual Core CPU and 1.5GB memory
  2. A workstation with 1.2 GHz CPU and 4GB memory
- Benchmarks

Table 1. The statistics of benchmarks

Case	Pin	Obs	**Case**	Pin	Obs
rc01	10	10	**rc07**	200	500
rc02	30	10	**rc08**	200	800
rc03	50	10	**rc09**	200	1000
rc04	70	10	**rc10**	500	100
rc05	100	10	**rc11**	1000	100
rc06	100	500	**rc12**	1000	10000

ROUTING BASED TREE CONSTRUCTION

- To simplify the PDOARST only considering wirelength
- To compare with works which minimize the total wirelength
  - [9] Z. Shen, C. C. N. Chu, and Y.-M. Li, "Efficient rectilinear Steiner tree construction with rectilinear blockages," *Proceedings of IEEE Intl. Conference on Computer Design*, pp. 38-44, Oct. 2005.
  - [10] C.-W. Lee, S.-Y. Chen, C.-F. Li, Y.-W. Chang, and C.-L. Yang, "Obstacle-avoiding rectilinear Steiner tree construction based on spanning graph," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 4, pp. 643-653, Apr. 2008.
  - [11] J. Long, H. Zhou, and S. O. Memik, "An O(*n*log*n*) edge-based algorithm for obstacle-avoiding rectilinear Steiner tree construction," *Proceedings of Intl. Symposium on Physical Design*, pp. 126-133, 2008

ROUTING BASED TREE CONSTRUCTION

Table 2. Comparison between wirelength and runtimes

Case	Wirelength				Runtime			
	our OARST (um)	**[9] *diff* (%)**	**[10] *diff* (%)**	**[11] *diff* (%)**	**our OARST (s)**	**[9] *spdup* (%)**	**[10] *spdup* (%)**	**[11] *spdup* (%)**
rc01	26810	0.34	0.34	2.72	0.01	0.00	0.00	0.00
rc02	42280	1.32	-0.17	-0.59	0.01	0.00	0.00	0.00
rc03	56160	0.50	-0.73	-0.16	0.01	0.00	0.00	0.00
rc04	60710	0.21	-0.59	-1.91	0.01	0.00	0.00	0.00
rc05	77330	-0.47	-1.29	-1.31	0.01	1.00	0.00	0.00
rc06	86299	0.12	-3.40	1.31	0.06	3.50	0.83	0.33
rc07	116801	0.54	-3.03	-0.90	0.06	5.50	1.17	0.17
rc08	123004	0.29	-3.46	1.50	0.09	5.22	2.33	0.67
rc09	120062	-0.26	-3.24	0.41	0.15	3.80	1.53	0.60
rc10	170600	0.50	0.05	-1.02	0.03	6.67	5.33	0.00
rc11	238905	-0.33	-.096	-1.30	0.10	5.40	7.60	-0.20
rc12	858310	-1.72	-8.06	-0.69	2.89	14.39	19.23	0.37
Ave.		**0.34**	**-2.05**	**-0.01**		**3.79**	**3.17**	**0.16**

- [*] *diff* of wirelength = ([*]-ours)/ours ×100.
- [*] *spdup* of runtime = [*]/ours-1.

PERFORMANCE-DRIVEN OARST

Table 3. Comparison between wirelength, worst delays, and runtimes of our simplified OARST and PDOARST

Case	Wirelength		WorstDelay		Runtime	
	our OARST (um)	**PD *diff* (%)**	**our OARST (ps)**	**PD *diff* (%)**	**our OARST (s)**	**PD *diff* (1X)**
rc01	26810	8.69	3709.40	-8.78	0.01	0.00
rc02	42280	-0.50	4757.91	-0.85	0.01	1.00
rc03	56160	8.53	8906.42	-35.91	0.01	0.00
rc04	60710	11.55	8124.20	-31.81	0.02	-0.50
rc05	77330	12.17	11690.13	-39.79	0.03	0.00
rc06	86299	5.19	10685.59	-2.99	0.21	0.71
rc07	116801	5.19	13450.84	-12.69	0.20	1.80
rc08	123004	4.71	16169.9	-21.87	0.29	2.21
rc09	120062	12.62	20957.15	-19.98	0.59	1.88
rc10	170600	2.87	25946.16	-33.25	0.11	2.36
rc11	238905	2.37	36459.46	-16.92	0.38	1.21
rc12	858310	39.84	464903.00	-64.58	15.95	8.41
Ave.		**9.44**		**-24.12**		**1.59**

SLACK-DRIVEN OARST

Table 4. Comparison between wirelength, worst delays, WNSs, and runtimes of PDOARST and SDOARST

Case	Wirelength		WorstDelay		WNS		Runtime	
	PD (um)	**SD *diff* (%)**	**PD (ps)**	**SD *diff* (%)**	**PD (ps)**	**SD *imp (%)***	**PD (s)**	**SD *diff* (1X)**
rc01t	29140	3.88	3383.65	-9.08	-635.78	100	0.01	0.00
rc02t	42070	14.67	4717.57	-3.87	-1669.34	67.60	0.02	-0.50
rc03t	60590	7.66	5708.50	20.82	-924.33	39.05	0.01	0.00
rc04t	67720	15.39	5540.01	5.18	-651.98	100	0.01	0.00
rc05t	86740	12.23	7038.70	-2.42	-148.84	100	0.03	-0.33
rc06t	90777	34.33	10365.80	-8.64	-671.98	100	0.36	0.19
rc07t	122858	44.67	11744.40	7.78	-189.52	100	0.56	0.29
Rc08t	128803	41.07	12634.00	3.38	0	0	0.93	0.60
rc09t	135215	31.70	16769.19	-11.82	-3687.34	100	1.7	-0.17
rc10t	175500	18.83	17319.80	-12.49	-2912.73	100	0.37	-0.24
rc11t	244572	17.09	30291.00	-32.79	-9966.89	95.09	0.84	-0.12
rc12t	1200290	18.69	164655.00	-4.54	-32595	96.55	150.12	0.10
Ave.		**21.68**		**-4.04**		**83.19**		**-0.02**

CONCLUSIONS

- We apply an routing algorithm to construct Steiner tree in spanning-graph based approach.
- We propose a critical-trunk based tree growth mechanism.
- We construct an obstacle-avoiding rectilinear Steiner tree with different objective.
  - Minimization of maximum delay
  - Maximization of the worst negative slack