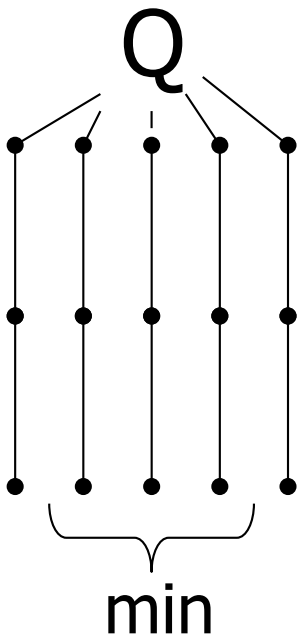
CS 347
Distributed Databases and Transaction Processing
Notes04: Query Optimization

Hector Garcia-Molina

Zoltan Gyongyi

Query optimization

- Cost estimation
- Strategies for exploring plans



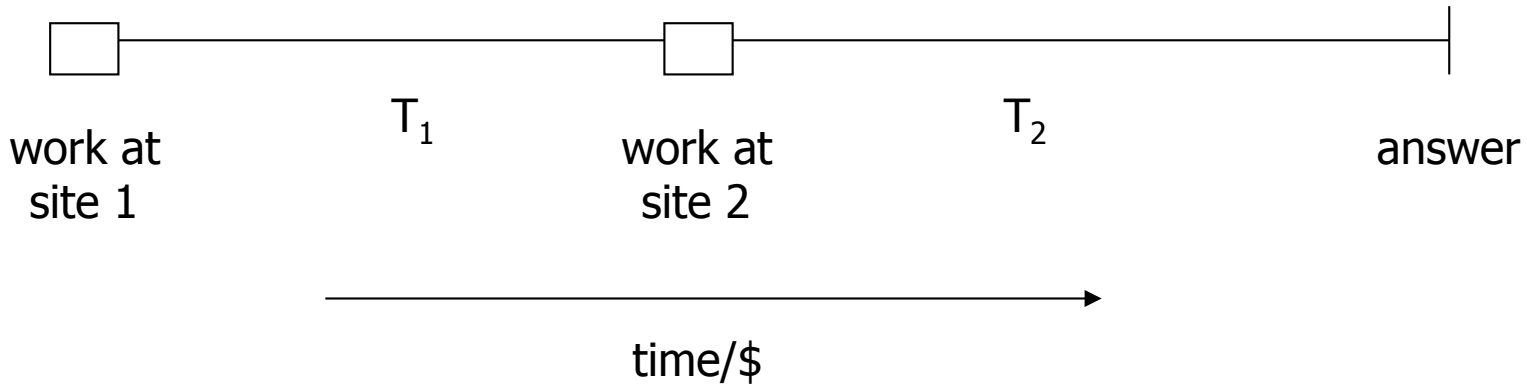
Cost estimation

☞ As in centralized systems:

estimate result sizes

- But # of IOs may not be the best metric

  E.g., transmission time may dominate cost



Another reason why plain IOs is not enough:

parallelism

Plan A

100 IOs

Plan B

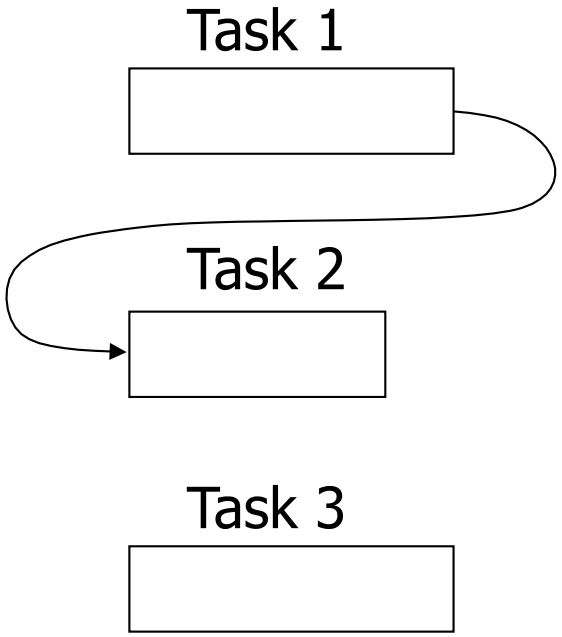
site 1  50 IOs

site 2  70 IOs

site 3  50 IOs

- Cost metrics
  - IOs, bytes transmitted, $, …
  - Can add together

- Response time metric
  - Cannot add
  - Need scheduling and dependency info
  - Skew is important

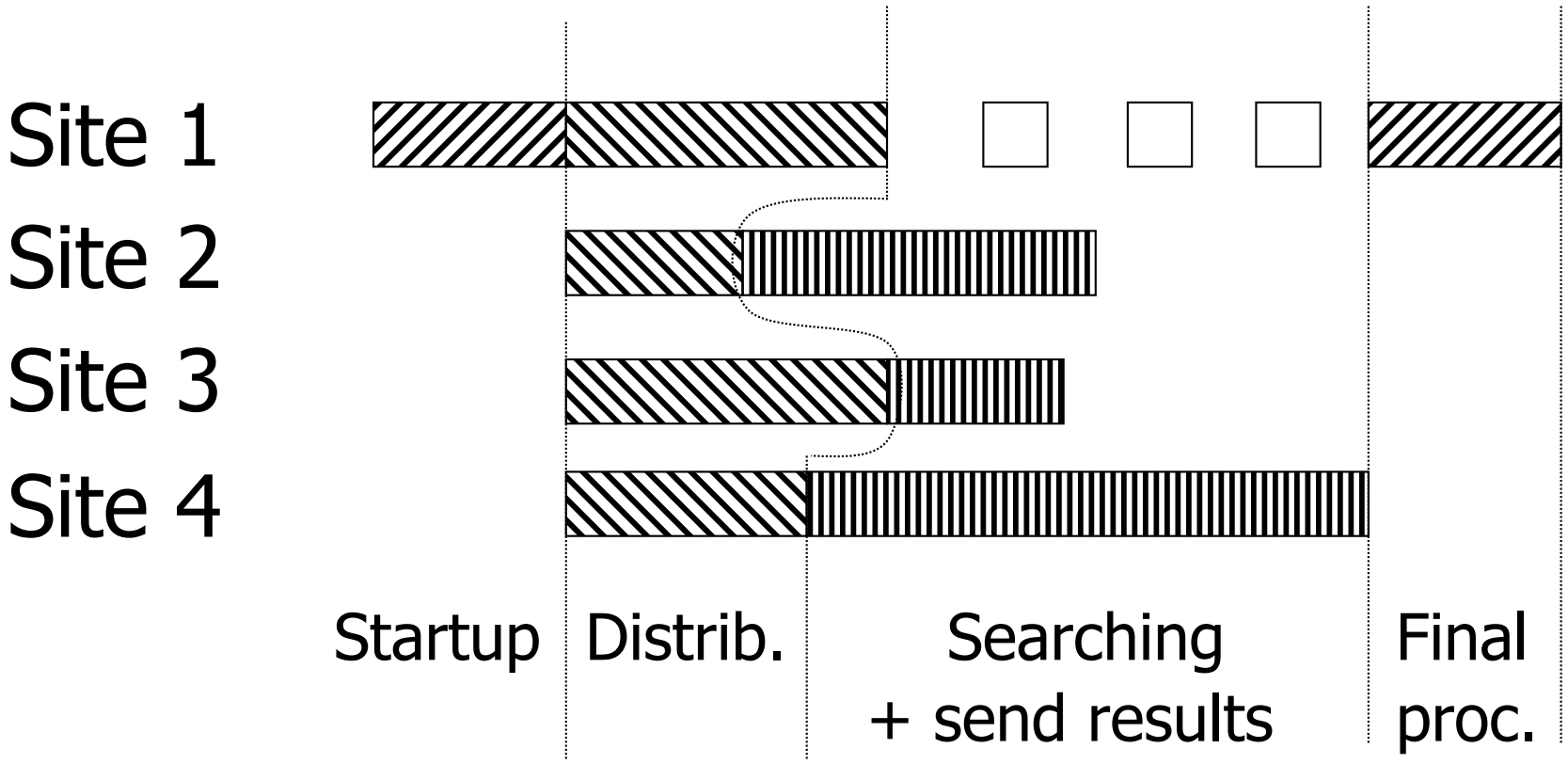


☞ Take into account:

(in parallel/distributed systems)

- Start up costs (for parallel operation)
- Data distribution costs/time
- Contention
  - Memory, disk, network, …
- Assembling result

Example: Response time



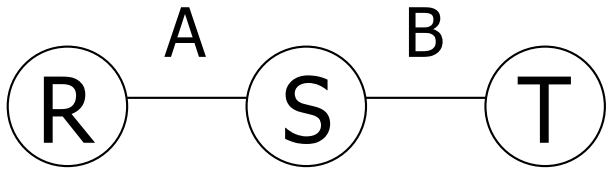
Searching strategies

1) Exhaustive (with pruning)
2) Hill climbing (greedy)
3) Query separation

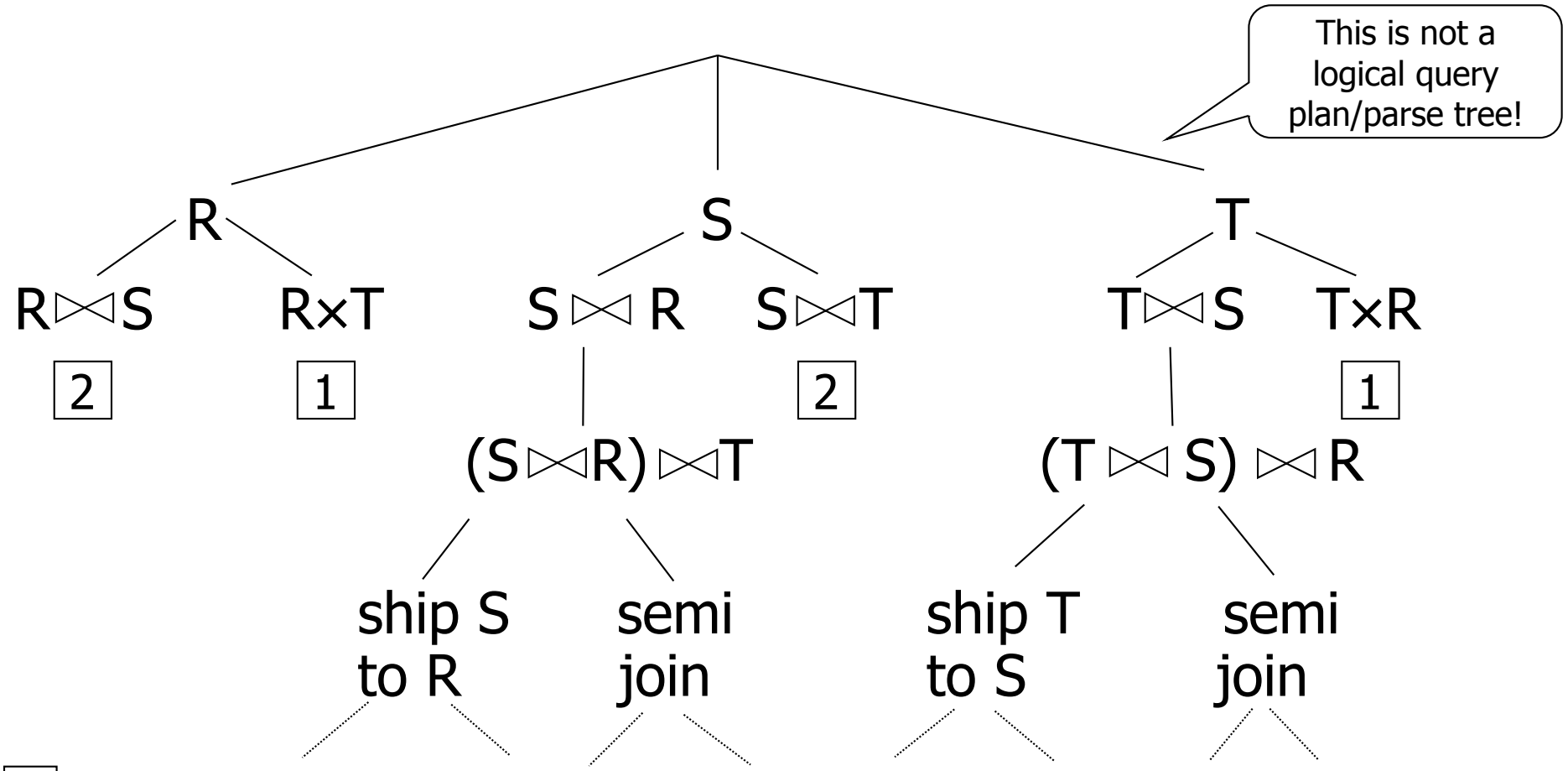
1) Exhaustive

- Consider "all" query plans with a set of techniques
- Prune some plans
- Heuristics

Example: join



|R|>|S|>|T|



1 Prune because cross-product not necessary

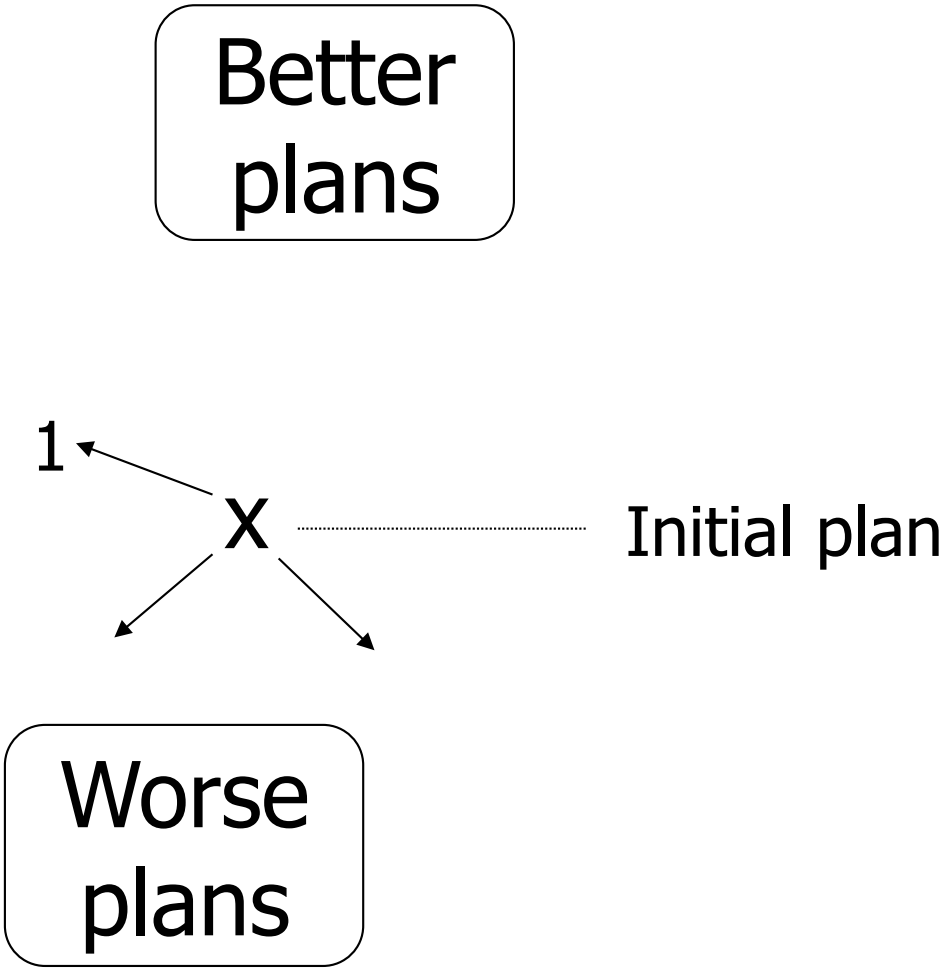
2 Prune because larger relation first

☞ In generating plans, keep goal in mind:

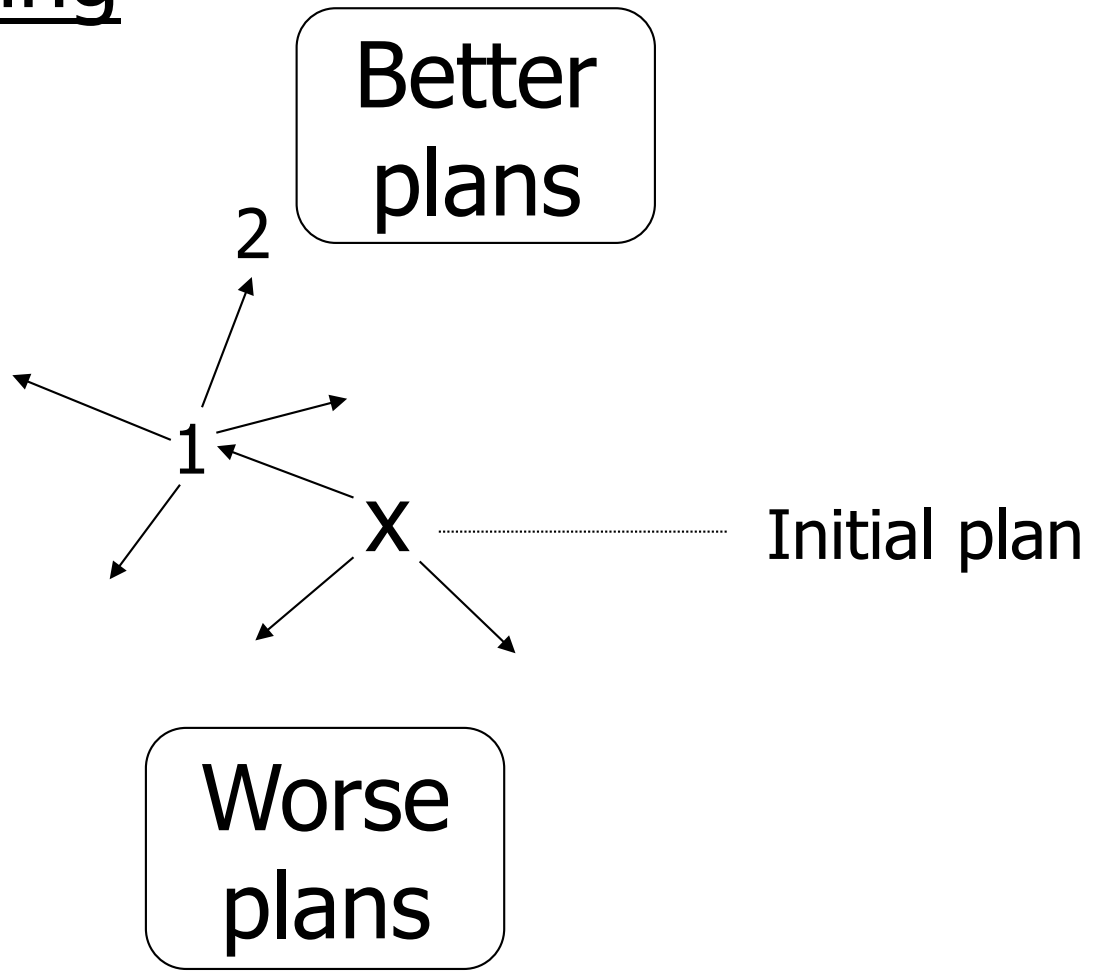
E.g., goal is parallelism in system with fast net → consider partitioning relations first

E.g., goal is reduction of net traffic → consider semi-joins

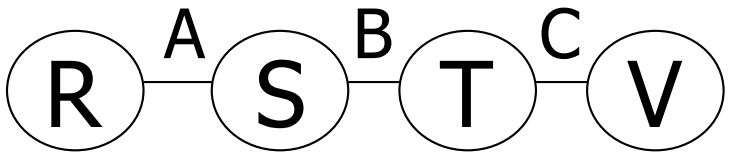
2) Hill climbing



2) Hill climbing



Example $R \bowtie S \bowtie T \bowtie V$



Rel	Site	Size
R	1	10
S	2	20
T	3	30
V	4	40

tuple size = 1

Goal: minimize data transmission

Initial plan: send relations to one site

What site do we send all relations to?

To site 1: cost=20+30+40=90

To site 2: cost=10+30+40=80

To site 3: cost=10+20+40=70

To site 4: cost=10+20+30=60 ✔

P_0: R (1 $\rightarrow$ 4)

S (2 $\rightarrow$ 4)

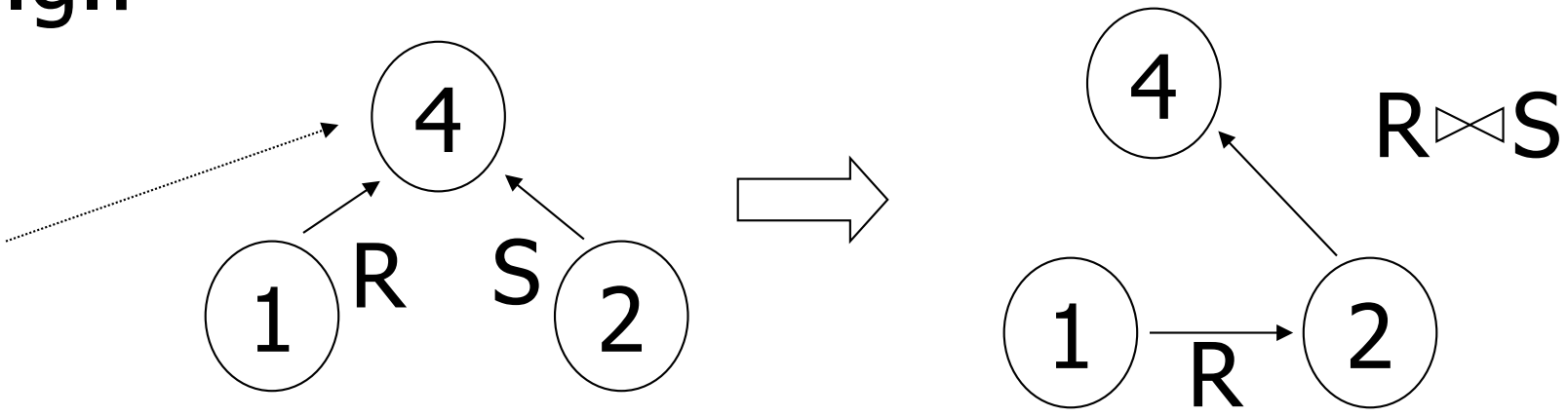
T (3 $\rightarrow$ 4)

Compute R $\bowtie$ S $\bowtie$ T $\bowtie$ V at site 4

Local search

- Consider sending each relation to neighbor:

E.g.:

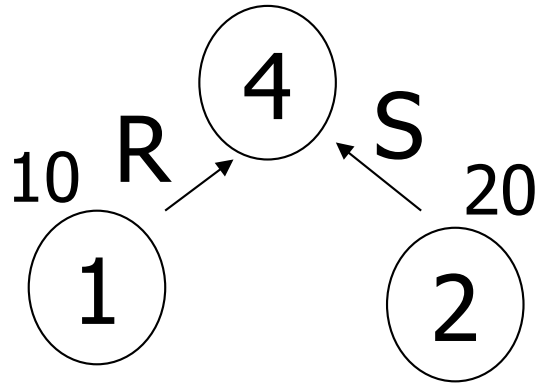


Assume: size $R \bowtie S = 20$

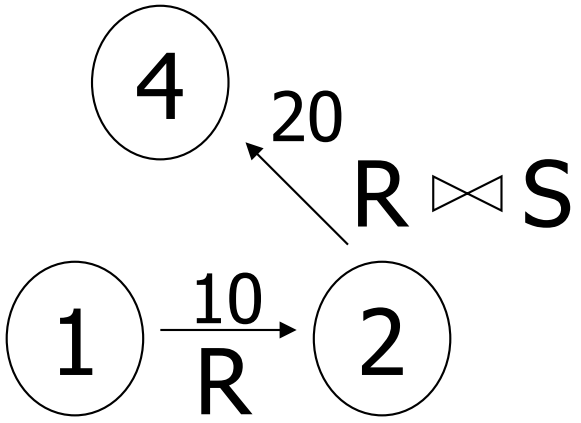
$S \bowtie T = 5$

$T \bowtie V = 1$

Option (a)



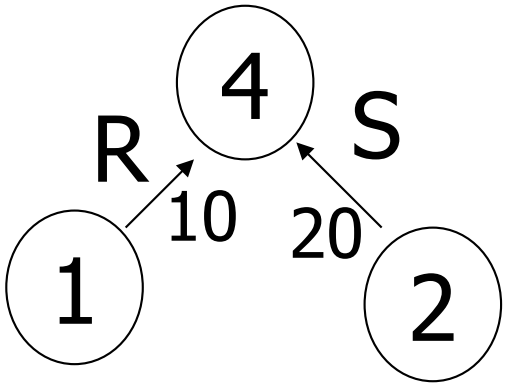
cost = 30

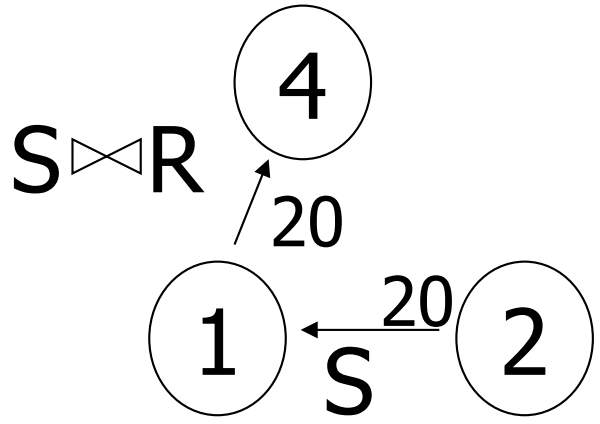


cost = 30

No savings

Option (b)



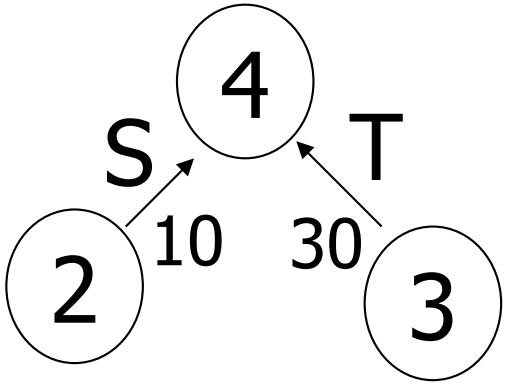


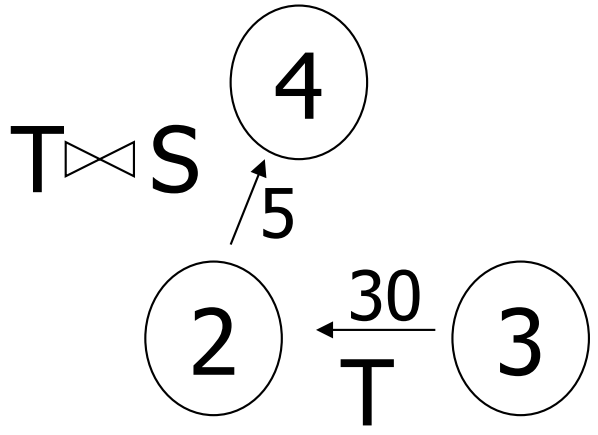
cost = 30

cost = 40

Worse off!

Option (c)



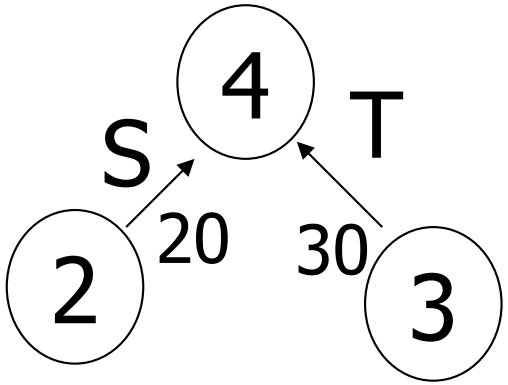


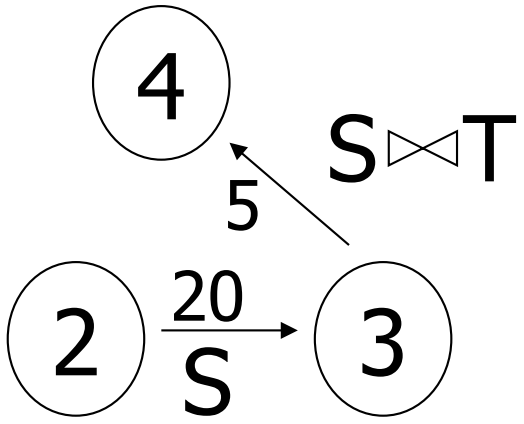
cost = 50

cost = 35

A win!

Option (d)





cost = 50

cost = 25

A bigger win!

P_1: P_1a: $S\ (2 \rightarrow 3)$

$\alpha = S \bowtie T$

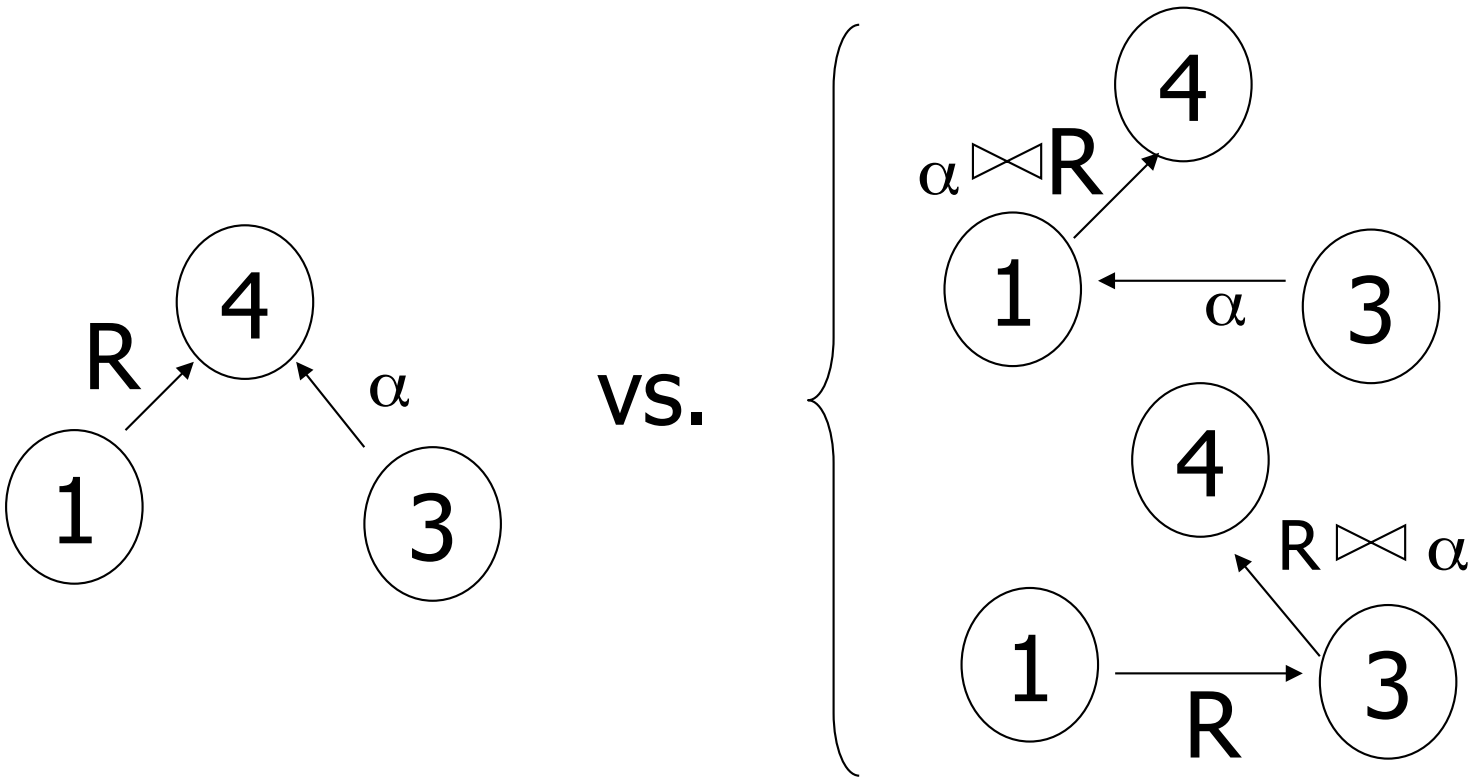
P_1b: $R\ (1 \rightarrow 4)$

$\alpha\ (3 \rightarrow 4)$

compute answer at site 4

Repeat local search

Treat $\alpha = S \bowtie T$ as relation



Hill climbing may miss best plan!

Example: best plan could be

P_B: T $(3 \rightarrow 4)$

$\beta = T \bowtie V$

β $(4 \rightarrow 2)$

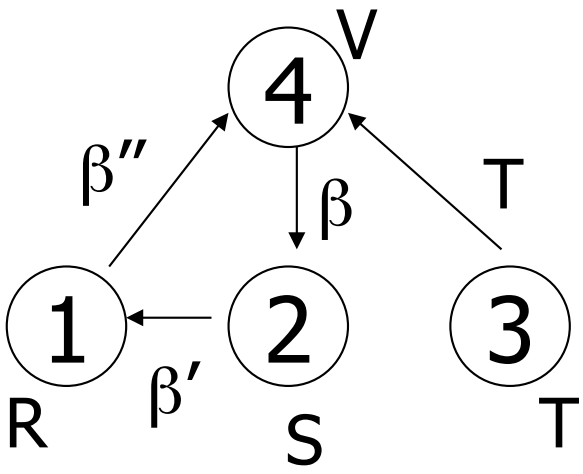
$\beta' = \beta \bowtie S$

β' $(2 \rightarrow 1)$

$\beta'' = \beta' \bowtie R$

[optional] β'' $(1 \rightarrow 4)$

Compute answer



Hill climbing may miss best plan!

Example: best plan could be

P_B: T $(3 \rightarrow 4)$ 30

$\beta = T \bowtie V$

β $(4 \rightarrow 2)$ 1

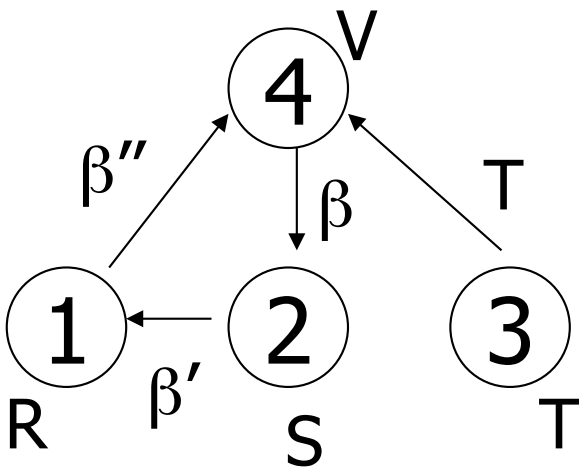
$\beta' = \beta \bowtie S$

β' $(2 \rightarrow 1)$ 1

$\beta'' = \beta' \bowtie R$

[optional] β'' $(1 \rightarrow 4)$ 1

Compute answer 33 = total



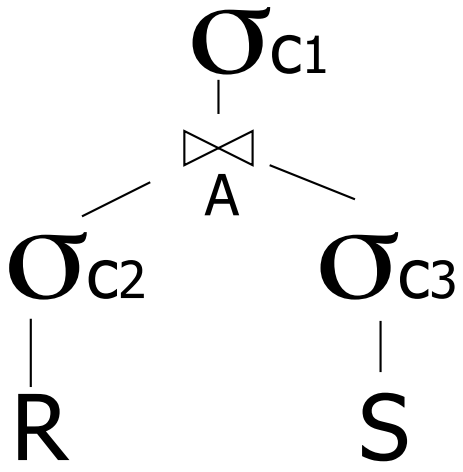
Costs could be low because β is very selective

3) Query separation

- Separate query into 2 or more steps
- Optimize each step independently

Example: simple queries

E.g.:



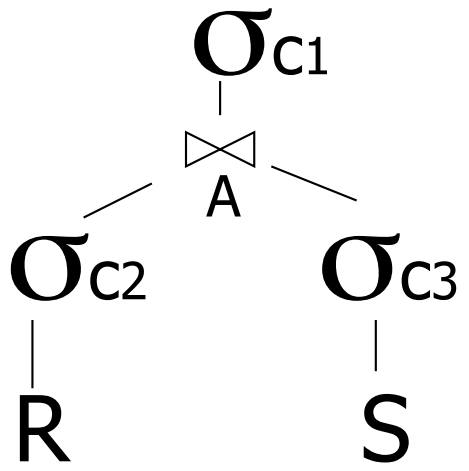
1. Compute $R' = \Pi_A[\sigma_{C2} R]$

   $S' = \Pi_A[\sigma_{C3} S]$

2. Compute $J = R' \bowtie S'$

1. Compute $R' = \Pi_A[\sigma_{C2} R]$

   $S' = \Pi_A[\sigma_{C3} S]$



2. Compute $J = R' \bowtie S'$

3. Compute

$$Ans = \sigma_{C1}\{[J \bowtie \sigma_{C2} R] \bowtie [J \bowtie \sigma_{C3} S]\}$$

In other words:

- Compute "A" values in answer (steps 1 and 2)
- Get tuples from sites with matching "A" values and compute answer (step 3)

Simple query

- Relations have a single attribute
- Output has a single attribute
  E.g., $J \leftarrow R' \bowtie S'$

Idea

- Decompose query into
  - Local processing
  - Simple query (or queries)
  - Final processing
- Optimize simple query

Philosophy

- Hard part is distributed join
- Do this part with only keys;
  get rest of data later
- Simpler to optimize simple queries

Summary: Query optimization

- Cost estimation
- Strategies
  - Exhaustive
  - Hill climbing
  - Separation

Words of wisdom

"Optimization is like chess playing"

That is, may have to make sacrifices

(move data, partition relations, build indexes)

for later gains!