Parallel Cut Generation for Service Cost Allocation in Transmission Systems

Celso C. <u>Ribeiro</u>
Renata M. Aiex
M. Poggi de Aragão

Department of Computer Science
Catholic University of Rio de Janeiro

Summary

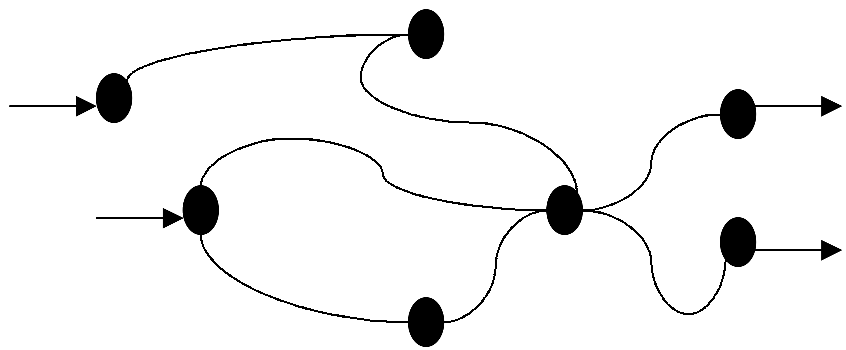
- LPs with exponential number of constraints
- Transmission Expansion Planning Problem
- Transmission Service Cost Allocation Problem
- Cut Generation
- Heuristic Separation
- Parallel Approach
- Computational Results

Linear Problems with Exponential Number of Constraints

- Many applications in practice
- Implicit representation: constraints do not have to be completely stored in memory
- Cut generation: solve a restricted LP, find a violated constraint (separation problem), append it to the restricted LP
- Polynomial procedure if separation problem can be solved in polynomial time
- Exact vs. separation procedure

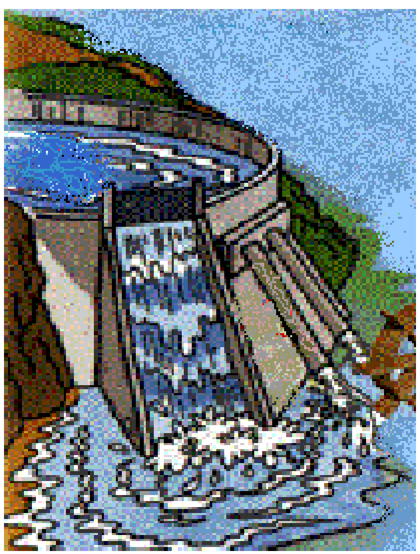
The Transmission Expansion Planning Problem (TEPP) (1)

undirected graph G ⟺ transmission network

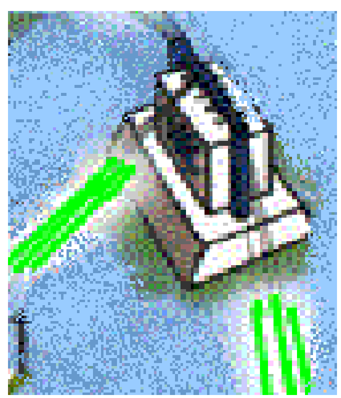


demand centers

power plants



substations



⇒ Edges are transmission lines

The Transmission Expansion Planning Problem (TEPP) (2)

Edges are transmission lines with:

- installed capacity
- maximum number of additional expansions
- incremental capacity
- cost per expansion

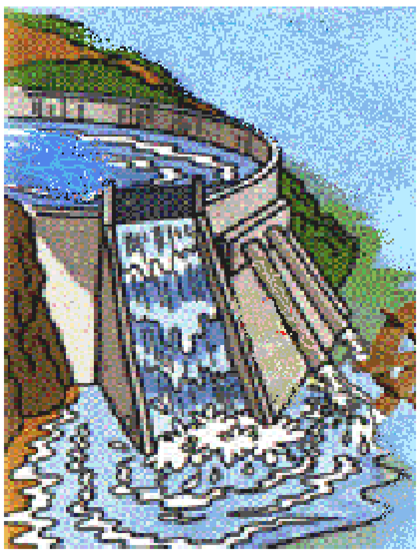
Minimize the expansion costs, supplying the demand centers from the power plants

The Transmission Expansion Planning Problem (TEPP) (3)

$$
\begin{cases}
\text{Minimize} \sum_{(i,j)\in E} d_{ij}\, x_{ij} \\
\sum_{j:(j,i)\in E} f_{ji} - \sum_{j:(i,j)\in E} f_{ij} = b_i^0 + \sum_{k=1}^{K} b_i^k & \forall i \in V \\
\left| f_{ij} \right| - c_{ij}\, x_{ij} \le \overline{c}_{ij} & \forall (i,j) \in E \\
x_{ij} \le u_{ij} & \forall (i,j) \in E \\
x_{ij} \in \mathrm{N} & \forall (i,j) \in E
\end{cases}
$$

The Transmission Service Cost Allocation Problem (TSCAP) (1)

- The transmission network is operated by a pool of *K* agents:



send power from power plants they operate

to demand centers they have as clients

The Transmission Service Cost Allocation Problem (TSCAP) (2)

Problem consists in assigning costs θ to the agents in the pool so that:

(1) the sum of costs θ assigned to agents is equal to the total service cost (network expansion);

(2) costs assigned to any subset of agents cannot exceed the cost they would incur if they decided to operate their own isolated system.

The Transmission Service Cost Allocation Problem (TSCAP) (3)

Master problem: Solution of a restricted linear problem

Separation problem: Identification of a violated constraint

$$
\begin{cases}
\text{Minimize } \Delta \\
\displaystyle\sum_{k=1}^{K} \theta_k + \Delta = Z^* & & (1) \\
\displaystyle\sum_{k \in S} \theta_k \leq Z_s & \forall S \subset \{1,\ldots,K\}, S \neq \varnothing & (2) \\
\theta_k \geq 0 & k = 1, \ldots, K \\
\Delta \geq 0
\end{cases}
$$

The Transmission Service Cost Allocation Problem (TSCAP) (4)

- Exponential number of type (2) constraints:

$$\sum_{k \in S} \theta_k \leq Z_S \Rightarrow 2^K - 1 \quad \text{constraints}$$

where K is the number of agents and
S is any subset of agents.

- Each Z_s is calculated solving a smaller Transmission Expansion Planning Problem (NP-hard) associated with subset S of agents

Cut Generation (1)

- Linear problems with an exponential number of constraints
- Implicit representation of the constraints
- Separation problem:
  - Subset of the constraints -> restricted LP
  - At each step, identify one (or the most) violated constraint
  - Separation problem is NP-hard (exact separation: branch-and-bound)

Solve the separation problem by a heuristic procedure.

Cut Generation (2)

$$
\begin{cases}
\text{Min} \sum_{(i,j)\in E} d_{ij} x_{ij} - \sum_{k=1}^{K} \overline{\theta_k} \lambda_k \\
\sum_{j:(j,i)\in E} f_{ji} - \sum_{j:(i,j)\in E} f_{ij} = b_i^0 + \sum_{k=1}^{K} b_i^k \lambda_k & \forall i \in V \\
\left| f_{ij} \right| - c_{ij} x_{ij} \le \overline{c_{ij}} & \forall (i,j) \in E \\
x_{ij} \le u_{ij} & \forall (i,j) \in E \\
x_{ij} \in \mathrm{N} & \forall (i,j) \in E \\
\lambda_k \in \{0,1\} & k = 1,\dots,K
\end{cases}
$$

- Exact solution: branch-and-bound
- Approximation (heuristic separation)

Cut Generation (3)

Procedure Solution-TSCAP

Find a feasible cost allocation θ to the restricted LP

Look for a **violated constraint** $\sum_{k \in S} \theta_k > Z_S$

If **a violated constraint was found,** then **append it** to the restricted linear problem

else **the current cost allocation is optimal** to the master problem

Cut Generation (4)

Separation problem with fixed agents:

$$
\begin{cases}
\text{Min} \sum_{(i,j)\in E} d_{ij} x_{ij} & \\
\sum_{j:(j,i)\in E} f_{ji} - \sum_{j:(i,j)\in E} f_{ij} = b_i^0 + \boxed{\text{constant}_i} & \forall i \in V \\
|f_{ij}| - c_{ij} x_{ij} \le \overline{c_{ij}} & \forall (i,j) \in E \\
x_{ij} \le u_{ij} & \forall (i,j) \in E \\
x_{ij} \in \mathrm{N} & \forall (i,j) \in E \\
\lambda_k \in \{0,1\} & k = 1,\ldots,K
\end{cases}
$$

Heuristic Separation (1)

Two main components:

- Local search in the space of subset of agents *S*
- Compute the expansion cost Z_s associated with subset S of agents.

Heuristic Separation (2)

Local search in the space of subset of agents *S*

- Initial solution: (i) no agents, (ii) all agents, (iii) randomly generated set of agents, or (iv) set of agents associated with the last cut found.

- Neighborhood: all subsets of agents that differ from the current subset by exactly one agent.

- Stopping criteria: (i) a cut is found, or (ii) MaxTrials solution neighborhoods are investigated.

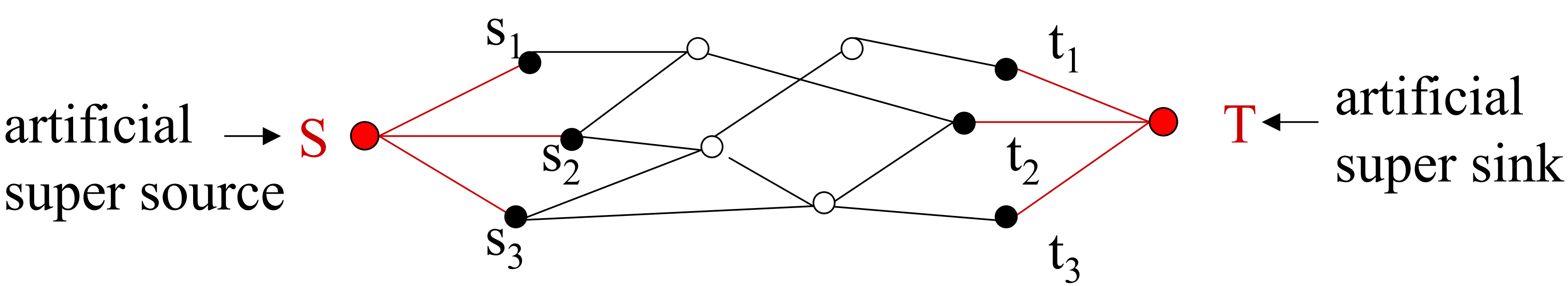
Heuristic Separation (3)

Heuristic construction of a feasible network

- For each subset of agents investigated in the local search step, heuristic construction of a feasible network solving TEPP(S) ⇨ approximation of the expansion cost Z_s associated with the subsystem formed by the agents in S
- Greedy construction: build a feasible network through the solution of a sequence of maximum flow problems (increase capacities of edges in minimum cut)
- Improvement procedure: decrease the number of expansions performed in each edge by rerouting its current flow

Heuristic Separation (4)

Heuristic construction of a feasible network

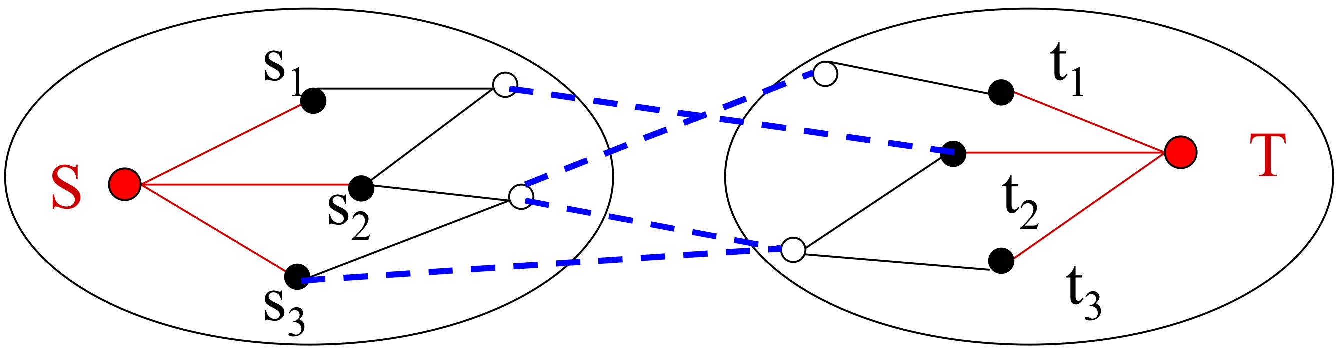


- Transform the network into a maximum flow problem
- Use a maximum flow algorithm (push-relabel) to check network feasibility ($F_{max} = F_{feasible} =$ demand)

Heuristic Separation (5)

Heuristic construction of a feasible network

- If $F_{max} < F_{feasible}$, then some arcs will be replicated.



- Find the arcs in the minimum cut closest to T.
- By successively replicating arcs, increase cut capacity from F_{max} to $F_{feasible}$.

Heuristic Separation (6)

Multi-item knapsack problem

- Replicated arcs should incur in minimum local cost.
- Strategies for the choice of edges:
  - increasing order of cost;
  - decreasing order of capacity;
  - increasing order of cost/capacity.

Heuristic Separation (7)

Procedure Increase-Cut-Capacity

do

Find the minimum cut closest to T

Sort the arcs in the cut

Replicate each arc (i,j) until $\geq$

(i) number of replications = max number of replications, or

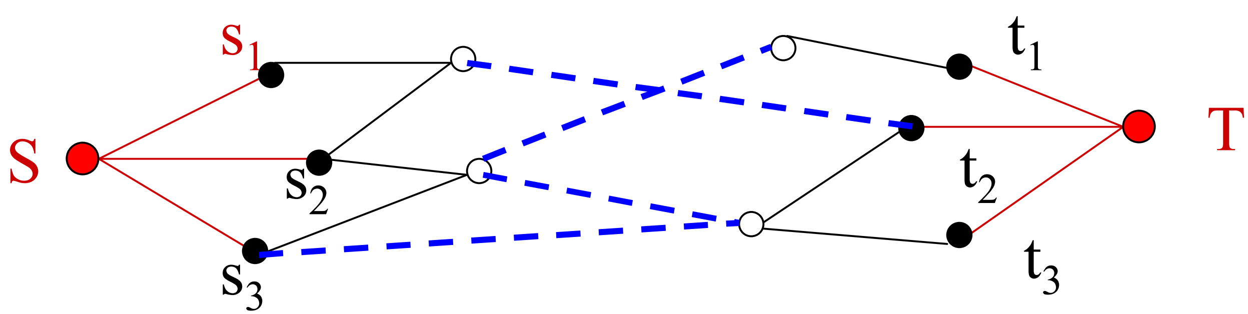
(ii) flow increase in the cut greater or equal than $F_{feasible} - F_{max}$

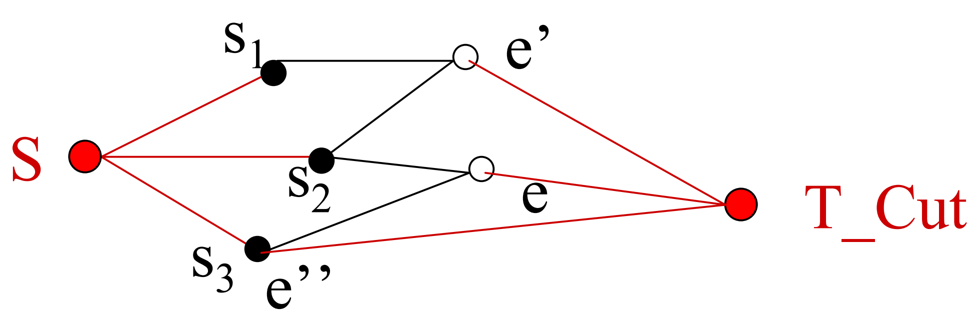
Solve max flow problem in the expanded network

until $F_{feasible} = F_{max}$

Heuristic Separation (8)

Heuristic construction with excess control





Heuristic Separation (9)

Heuristic construction with excess control

do ...

Replicate each arc (i,j) until

(i) number of replications = max number of replications, or

(ii) flow increase in the cut greater $\geq$ or equal than $F_{feasible} - F_{max}$, or

(iii) (arc capacity x number of replications) > excess in node *i*

...

until $F_{feasible} = F_{max}$

Heuristic Separation (10)

Improvement procedure

- For each arc replication
  - Decrease number of replications by one
  - Solve maximum flow problem for the new network
  - If $F_{max} < F_{feasible}$, then reinstall arc

Heuristic Separation (11)

Local Search in the replicated arcs

- Move: for each arc replication
    - Decrease replications by one
    - Reconstruct the network using same heuristics presented
    - Run the improvement procedure
- Steepest descent local search:

  Choose the most decreasing move

  Stop at the first locally optimal visited solution

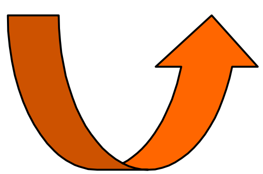
Parallel Approach (1)

Exact separation thread:

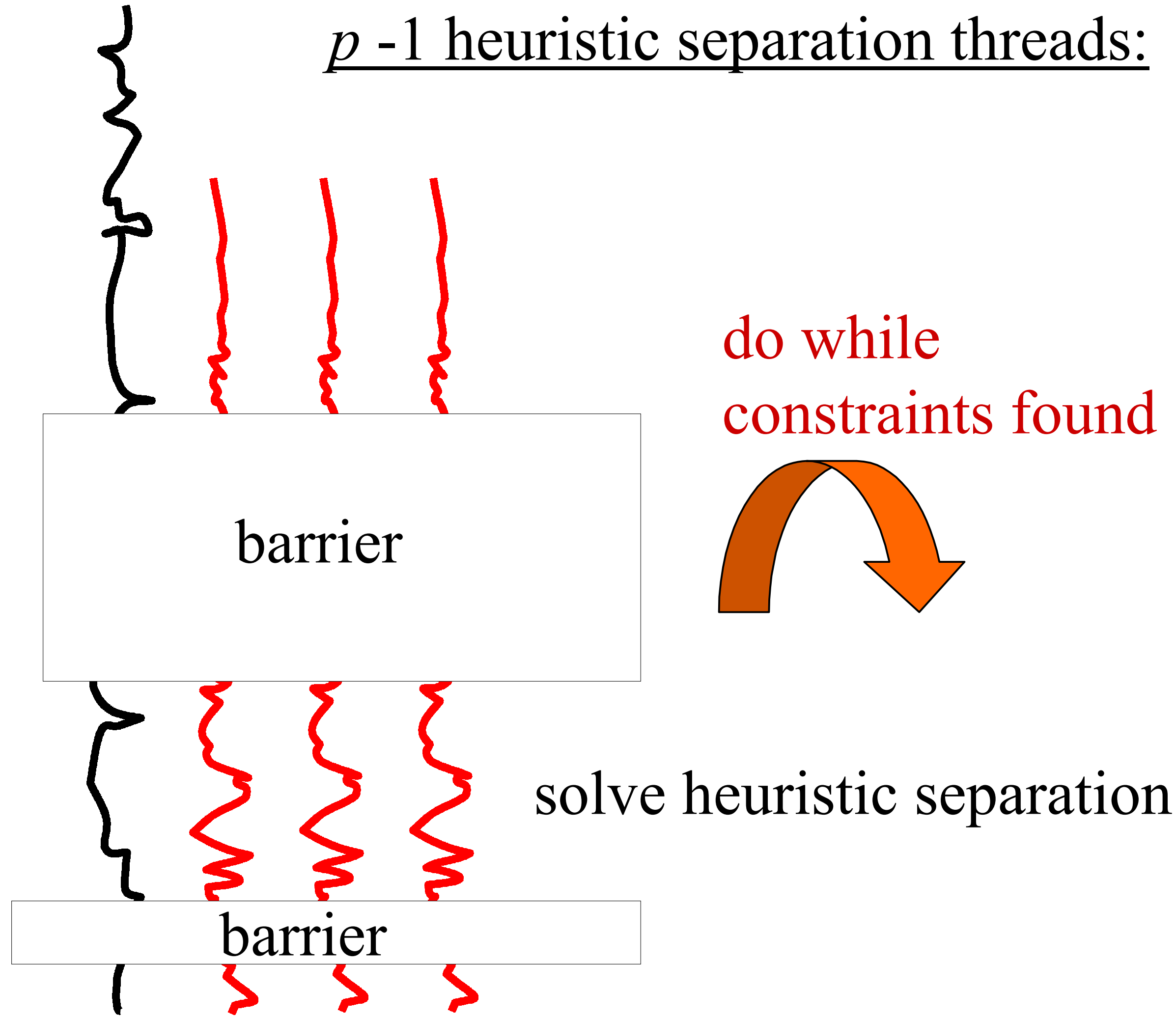
- starts heuristic threads
- initialize restricted LP

- appends constraints to restricted LP
- computes new costs θ

- solves exact separation



p -1 heuristic separation threads:



Parallel Approach (2)

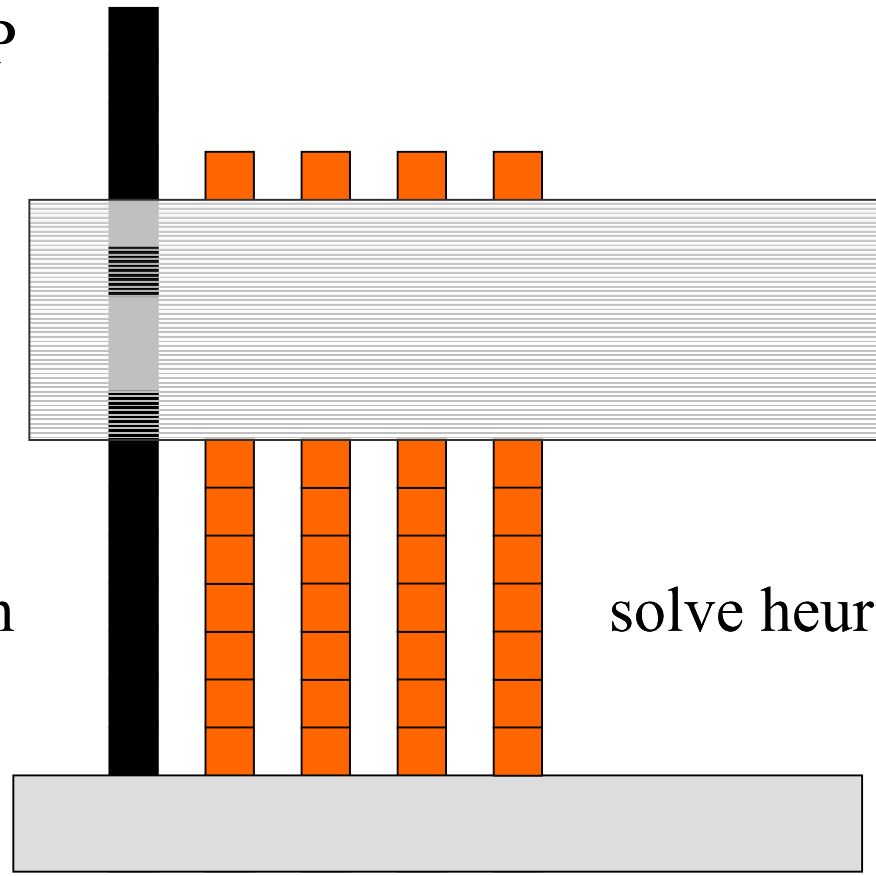
initializes restricted LP

starts heuristic threads

appends constraints
to restricted LP

computes new costs θ

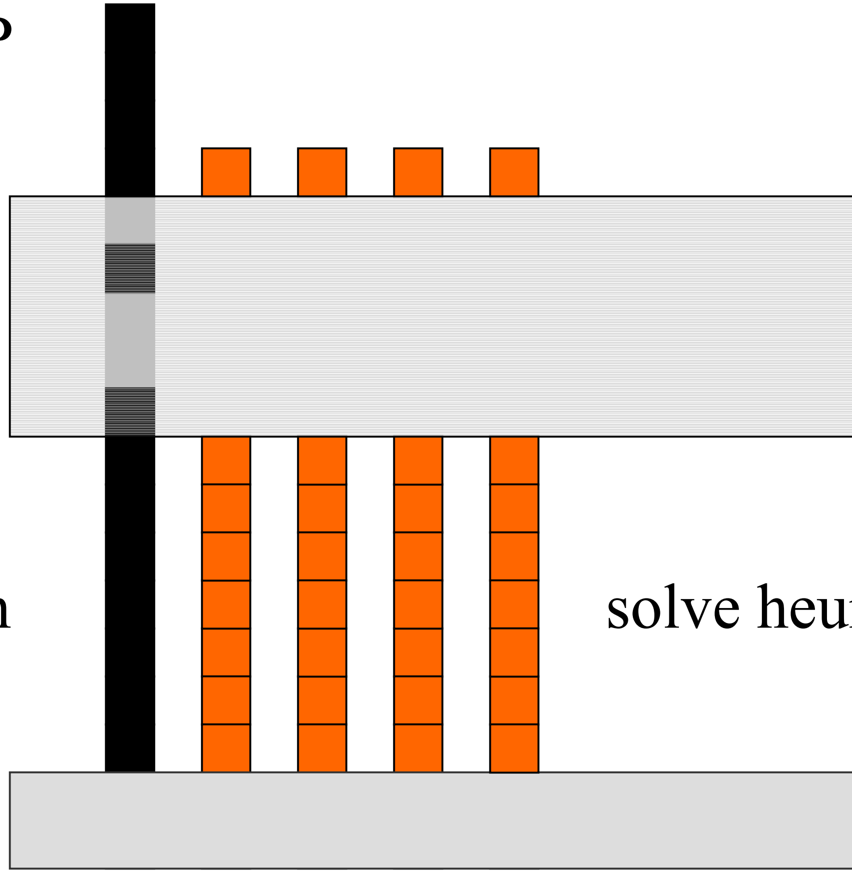
solves exact separation



solve heuristic separation

threads waiting

Parallel Approach (2)



Parallel Approach (3)

- Shared memory paradigm

- Multiple cuts per iteration:
  - CPLEX is very fast in generating new values for the costs θ (fast solution of the restricted LP)
  - Use multiple cuts, but do not wait too much for them

Parallel Approach (4)

- Global x local hashing tables
  - Used to store solutions visited during local search in the space of subset of agents *S*
  - Global hashing table:
    - Lock/unlock structures become a bottleneck when the number of processors is increased
    - Information is shared among processors
  - Improvement: use global hashing table for groups of processors

Parallel Approach (5)

- More precise x faster move evaluation at each iteration of the local search
  - faster evaluation: compute the cost of constructing a network for a subset of agents *S'* in the neighborhood of *S*, without applying local search to the arcs.
  - faster evaluation works much better then precise evaluation.

Computational Results (1)

- Sun Starfire ENT10000:
  - 32 Ultra Sparc 250 MHz processors
  - 8 Gbytes of RAM memory
  - 1 Mbyte of cache memory per processor
- Software:
  - c compiler
  - Posix threads
  - CPLEX 5.0

Computational Results (2)

- Agents in initial solutions: (i) same as in the previous cut, (ii) none, (iii) all, (other) random
- First iteration: 100 cuts
- Next, 70% of cuts found in previous iteration
- Each processor performing heuristic separation is ready-to-stop after investigating MaxTrials = 160 neighborhoods
- Stop heuristic separation: 70% of processors ready-to-stop and at least one cut found

Computational Results (3)

- Test problems derived from the Brazilian network
  - 16 and 19 agents
  - 79 nodes and 283 edges (134 can be replicated)

Computational Results (4)

	16 agents			19 agents		
Processors	1	5	9	1	5	9
Iterations	240	27	48	237	39	82
Total number of cuts	319	403	340	409	426	411
Cuts from exact separations	102	28	35	204	56	74
Cuts from heuristic separations	217	375	305	205	370	337
Exact separations required	23	3	3	32	3	5
Elapsed time (hh:mm)	8:40	4:38	1:41	20:25	5:26	5:16

Conclusions and Extensions

- Heuristic separation leads to faster computations even in sequential mode
- Effectiveness of parallel cut generation
- More systematic tests (other test instances; several runs or single user mode; criteria, parameters, and strategies used in the parallel implementation)
- Improvements in the local search heuristic
- Improvements in the network design heuristic

Parallel Cut Generation

END

Heuristic Separation (5)

Maximum Flow Problem (MFP)

$$
\text{MFP}\quad
\begin{cases}
F_{\max} = \quad \text{Max } F \\
\sum_{j:(j,i)\in E} f_{ji} - \sum_{j:(i,j)\in E} f_{ij} = 0 & \forall i \in V, i \neq s,t \\
\sum_{i:(s,i)\in E} f_{si} = F \\
\left| f_{ij} \right| \leq c'_{ij} & \forall (i,j) \in E \\
F \geq 0
\end{cases}
$$

Parallel Approach (6)

- Local search using patterns
  - Generate new values for the costs θ
  - Find the active cuts in the restricted LP
  - A pattern is a group of agents that appears together in many active cuts
  - Choose randomly (biased by the number of occurrences) a pattern to fix in the local search in the space of subset of agents of one of the processors.