Prize collecting Steiner tree problem

Heuristic & lower bounds

Maurício G. C. Resende

Algorithms & Optimization Research Dept.

AT&T Labs Research

Florham Park, New Jersey

mgcr@research.att.com

http://www.research.att.com/~mgcr

Joint work with S. Canuto, A. Lucena, & C.C. Ribeiro

May 2000

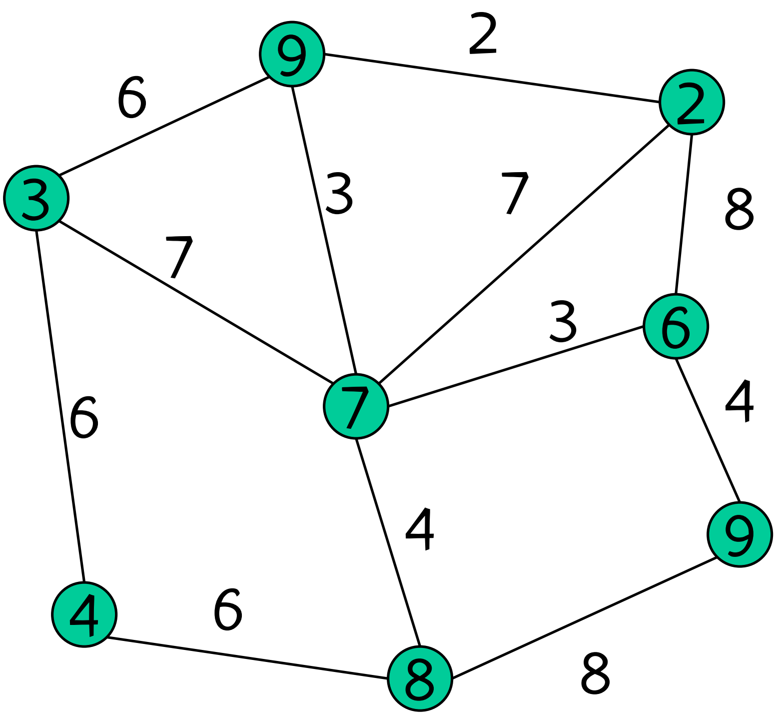
Outline

- Introduction
  - Problem definition
  - An application from telecommincations access network design
- Local search with perturbations: A heuristic
  - Local search with perturbations
  - Path relinking
  - Variable neighborhood search
- A cutting planes algorithm: Lower bounds
  - Integer programming formulation
  - Cutting planes algorithm
  - Preprocessing to reduce input graph size
- Computational results

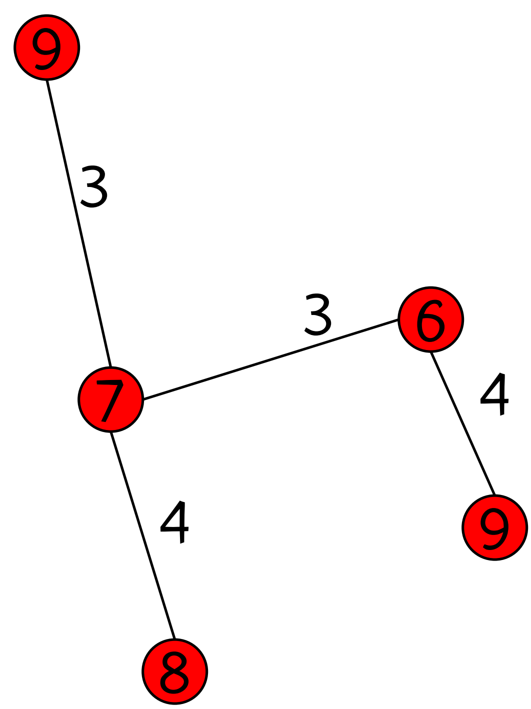
Prize-collecting Steiner tree (PCST) problem

- Given: graph $G = (V, E)$
  - Real-valued cost c_e is associated with edge e
  - Real-valued penalty d_v is associated with vertex v
- A tree is a connected acyclic subgraph of G and its weight is the sum of its edge costs plus the sum of the penalties of the vertices of G not spanned by the tree.
- PCST problem: Find tree of smallest weight.

Cost of tree



graph *G*



tree *T*

Cost (*T*) = (3+3+4+4) +
(2+3+4) = 23

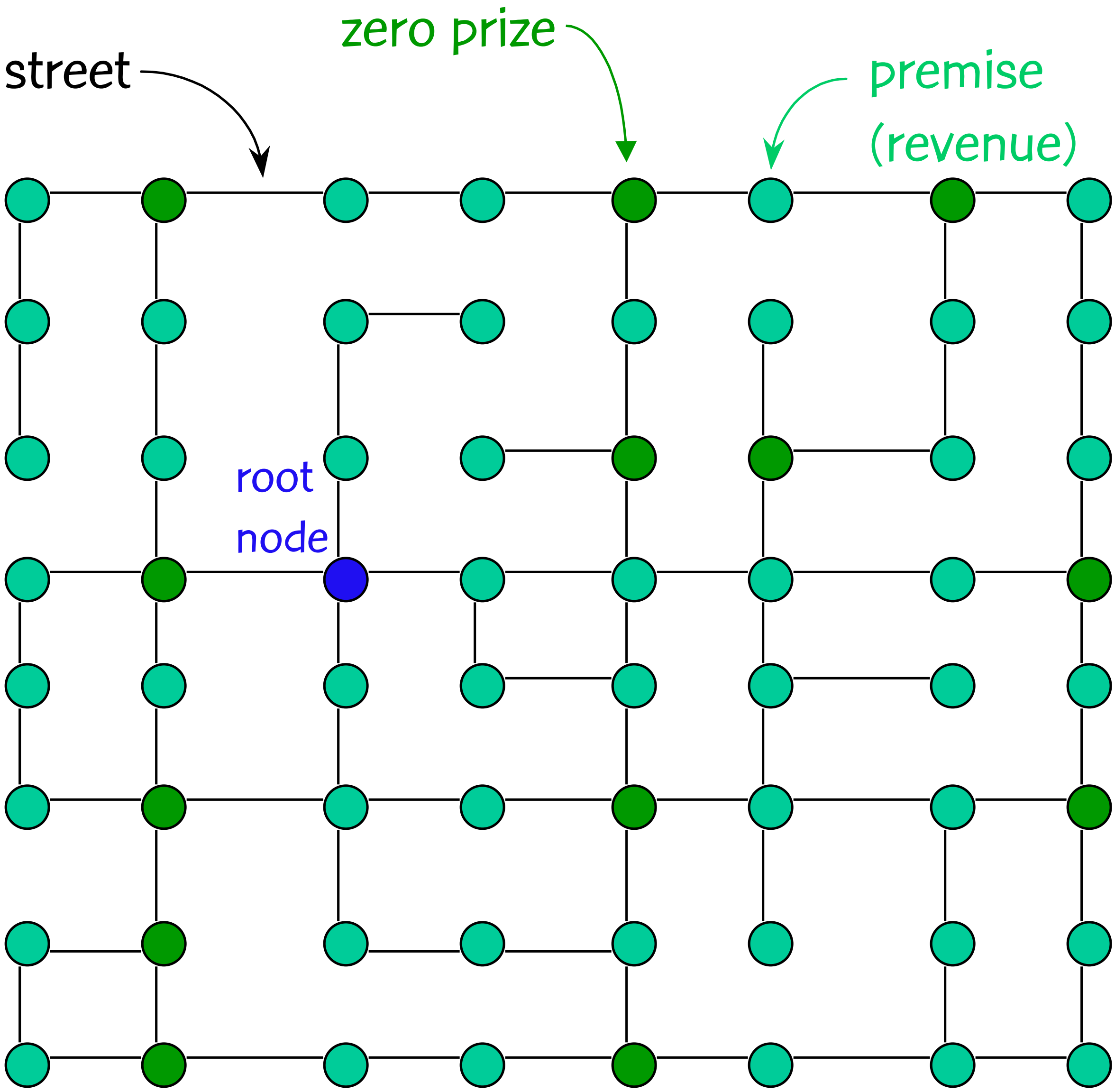
Design of local access telecommunications network

- Build a fiber-optic network for providing broadband connections to business and residential customers.
- Design a local access network taking into account tradeoff between:
  - cost of network
  - revenue potential of network

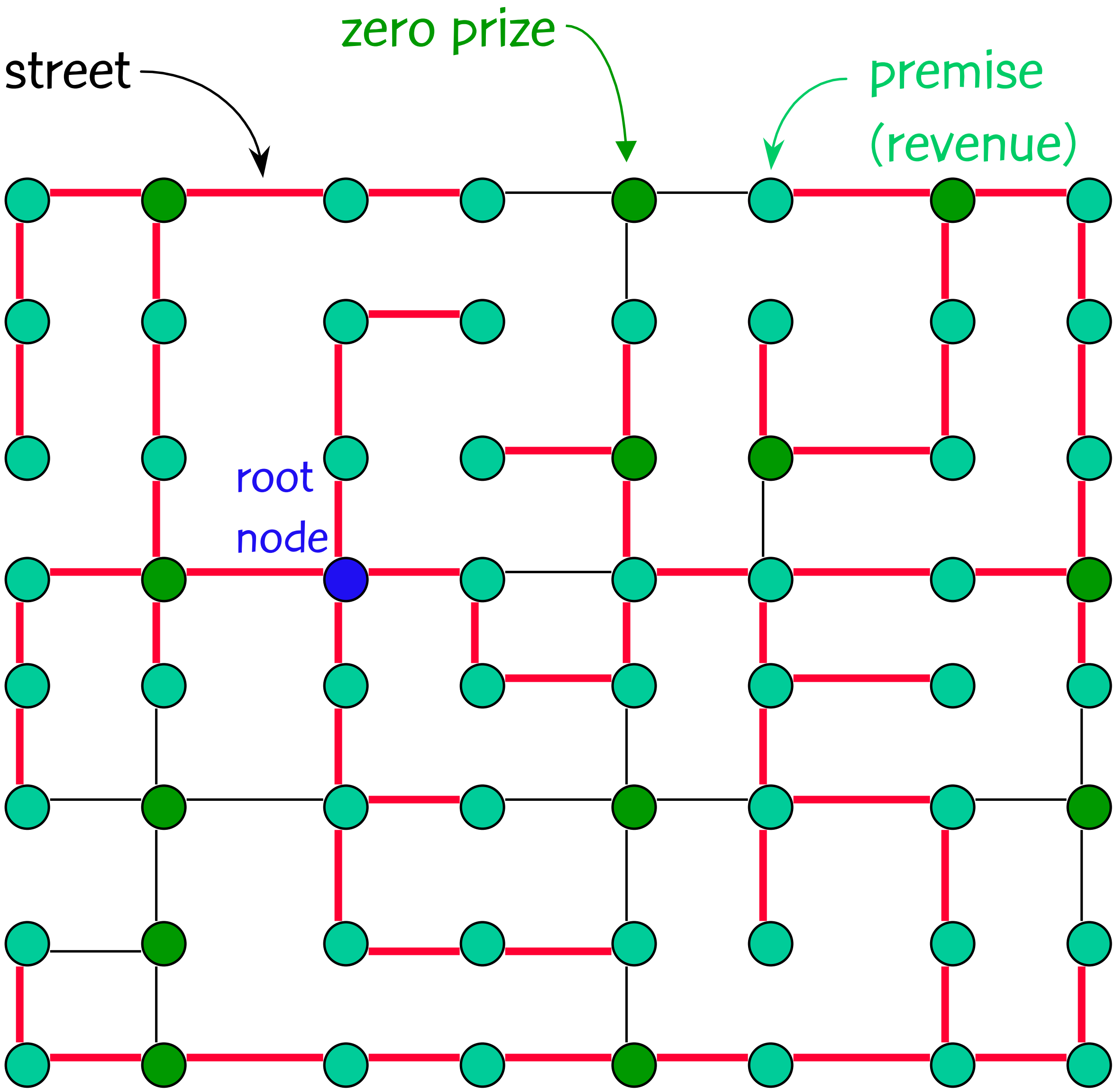
Design of local access telecommunications network

- Graph corresponds to local street map
  - Edges: street segments
    - Edge cost: cost of laying the fiber on the corresponding street segment
  - Vertices: street intersections and potential customer premises
    - Vertex penalty: estimate of potential loss of revenue if the customer were not to be serviced (intersection nodes have no penalty)

Local access network design

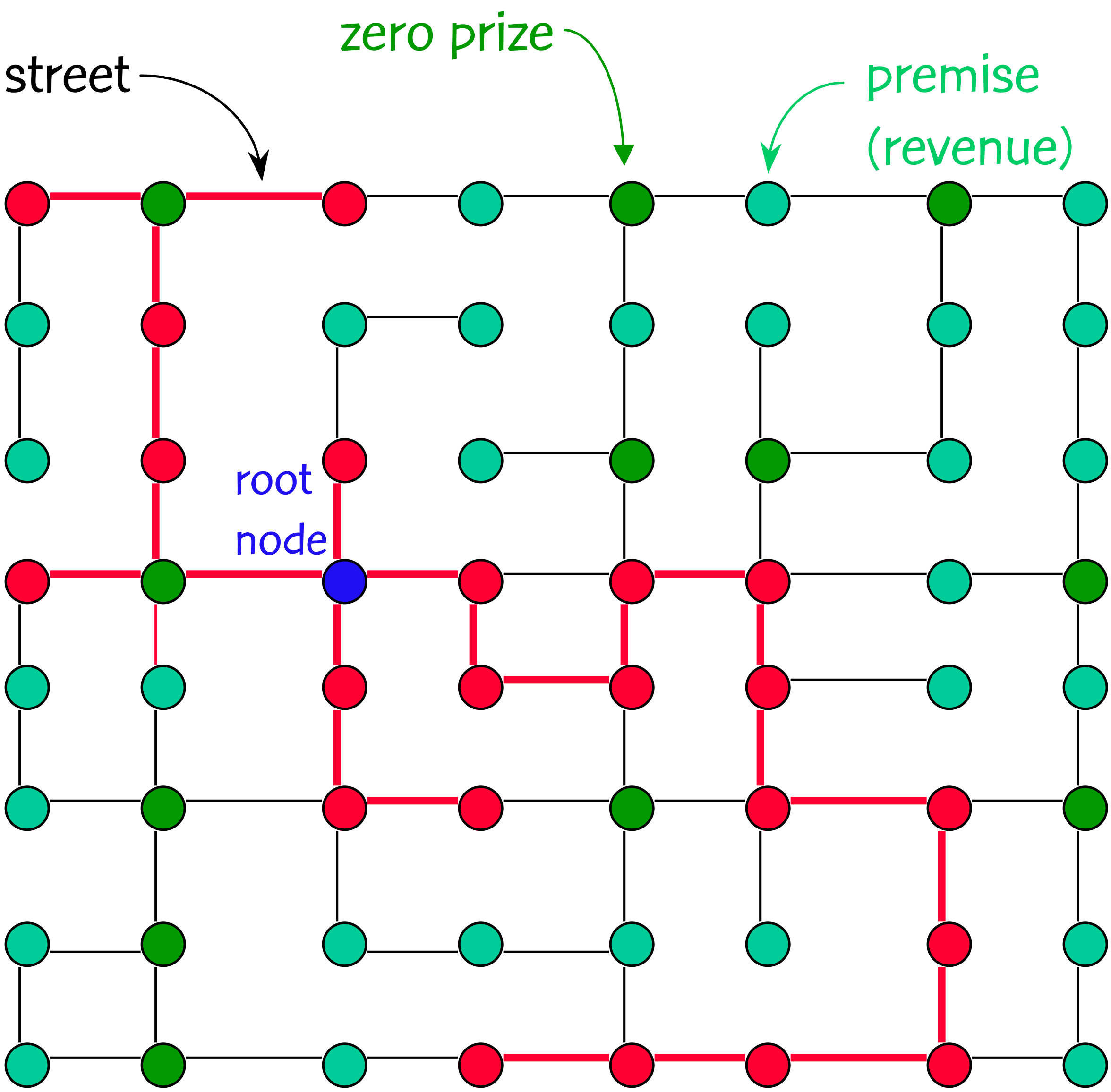


Collect all prizes
(Steiner problem in graphs)



Collect some prizes

(Prize-collecting Steiner Problem in Graphs)



Literature

- Introduced by Bienstock, Goemans, Simchi-Levi, & Williamson (1993)
- Goemans & Williamson (1993, 1996) describe 5/2 and 2 approximation algorithms
- Johnson, Minkoff, & Phillips (1999) describe an implementation of the 2-opt algorithm of Goemans & Williamson (GW)
- Canuto, R., & Ribeiro (1999) propose a multi-start heuristic that uses a randomized version of GW
- Lucena & R. (2000) propose a polyhedral cutting plane algorithm for computing lower bounds

Local search with perturbations: a heuristic

- Summary
  - Generation of initial solution
  - Local search
  - Multi-start strategy
  - Path-relinking associated with multi-start strategy
  - Variable neighborhood search

Generation of initial solution

- Select X, the set of collected nodes
- Connect node in X with minimum weight spanning tree $T(X)$
- Recursively remove from $T(X)$ all degree-1 nodes with prize smaller than its incident edge cost = $T_r(X)$
- Basic strategy:

for (i = 1 to MAXITR){

select X_i ← Goemans & Williamson 2-opt algorithm

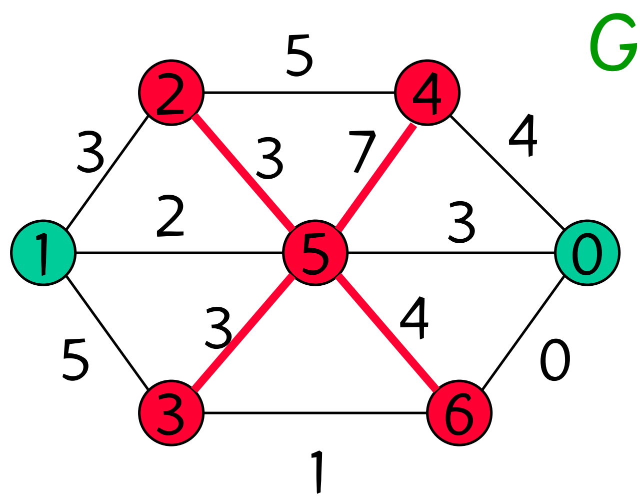
compute $T(X_i)$ and $T_r(X_i)$ ← Kruskal's algorithm

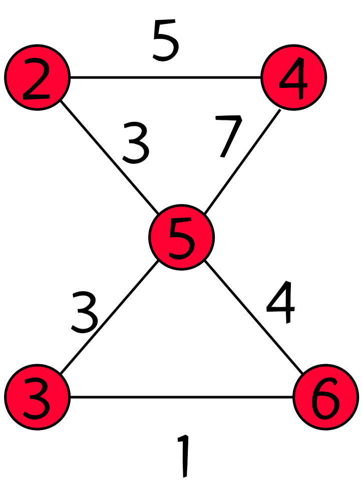
}

Generation of initial solution

Solution obtained by GW: $X = \{2,3,4,5,6\}$

Cost = 18

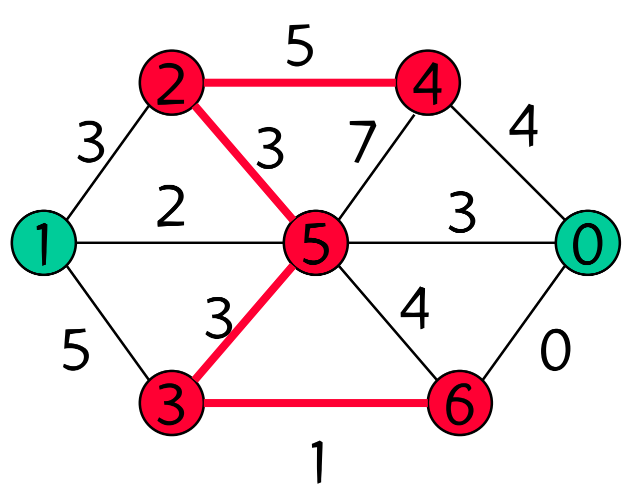




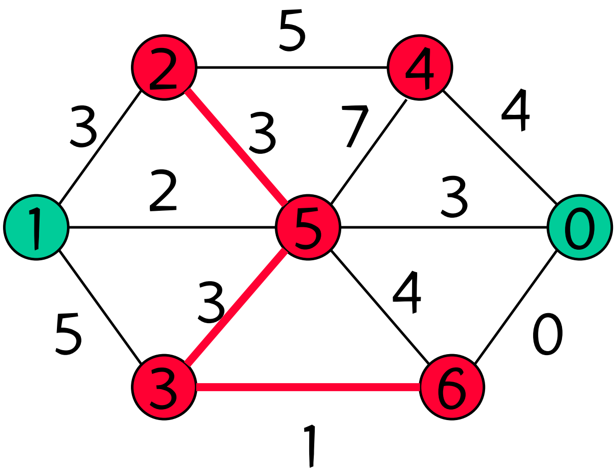
G' = subgraph induced on G by nodes in X

MST solution on G''

Cost = 13



Generation of initial solution

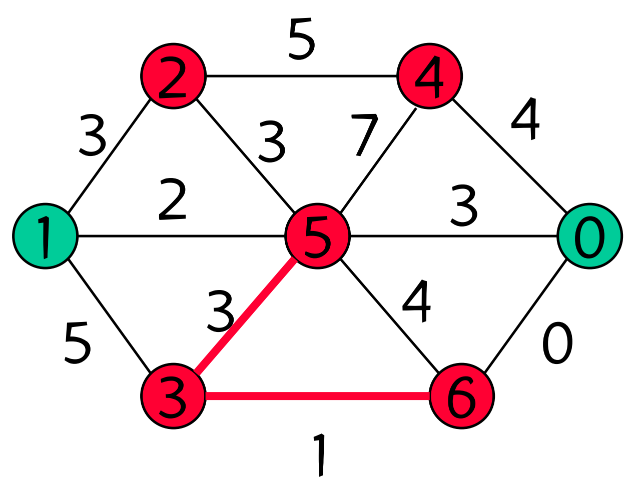


Solution obtained by pruning degree-1 node

Cost = 12

Final solution obtained by pruning another degree-1 node

Cost = 11



Local search

- Representation of solution: set X of vertices in tree $T(X)$
- Neighborhood:
  - $N(X) = \{X' : X$ and X' differ by single node$\}$
  - Moves: insertion & deletion of nodes
- Initial solution: nodes of tree obtained by GW
- Iterative improvement: make move as long as improvement is possible

Local search

```
improve = T
while ( improve){
    improve = F
    circfor i = 1, ..., |V| while .not. improve
    {       if (i ∈ X){ X' = X \ {i}}
            else {X' = X ∪ {i}}
            compute tree T(X') & cost(X')
            if (cost(X') < cost(X)){
                    X = X'
                    improve = T
            }
    }
}
```

Multi-start strategy

- Force GW to construct different initial solutions for local search
  - Use original prizes in first iteration
  - Use modified prizes after that
- Modify prizes (two strategies)
  - Introduce noise into prizes

    for $i = 1, \ldots, |V|$ {

    generate $\beta \in [1 - a, 1 + a]$, for $a > 0$

    $d'(i) = d(i) \times \beta$

    }
  - Node elimination
    - Set to zero the prizes of α% of the nodes in nodes(GW) $\cap$ nodes(local search)

Local search with perturbations

```
best = HUGE
d' = d
for ( i = 1, ..., MAXITR ){
    X = GW ( V, E, c, d' )
    X' = LOCALSEARCH(V, E, c, d, X )
    if ( cost(X' ) < best ){
        X* = X'
    }
    compute perturbations & update d'
}
return X*
```

Path relinking

- Integrates intensification & diversification
- Explores the path connecting good solutions
- In local search with perturbations let
  - X' be the local optimum found by LOCALSEARCH
  - Y be a solution chosen randomly from a POOL of elite solutions
  - $\Delta = \{i \in V : (i \in X'$ and $i \notin Y)$ or $(i \notin X'$ and $i \in Y)\}$
- Construct path between X' (start) and Y (guide):
  - Apply best movement in Δ
  - Verify quality of solution after move
  - Update Δ

Path relinking

- Criteria for inclusion of solution X into POOL of elite solutions
  - If cost(X) is less than smallest cost of POOL solutions
  - If cost(X) is less than largest cost of POOL solutions and X is sufficiently different from all POOL solutions
    - X_1 and X_2 are sufficiently different if they differ by at least β nodes, where β is a fraction of $|V|$

Local search with perturbations & path relinking

```
POOL = φ
d' = d
for ( i = 1, ..., MAXITR ){
    X = GW ( V, E, c, d' )
    if ( X is new){
        X' = LOCALSEARCH(V, E, c, d, X )
        attempt insert X' into POOL
        X'' ∈ RAND(POOL)
        X_PR = PATHRELINK(X' , X'' )
        attemp to insert X_PR into POOL
        }
    }
    compute perturbations & update d'
}
return best solution in POOL
```

Variable neighborhood search

- Can we gain something by going from a static neighborhood to one that is dynamic?
- Consider K neighborhoods:
  - $N^1, N^2, \ldots, N^K$
  - $N^k(X) = \{ X' : X$ and X' differ by k nodes$\}$
- Basic scheme (repeated MAXTRY times):
  - Start with initial solution X and $k = 1$
  - while ($k \leq K$){

    choose $X' \in N^k(X)$

    $k = k + 1$

    if cost(X') < cost(X) { $X = X'$; $k = 1$}

  }

Local search with perturbations & path relinking & VNS

```
POOL = ϕ
d' = d
for ( i = 1, ..., MAXITR ){
    X = GW ( V, E, c, d' )
    if ( X is new){
        X' = LOCALSEARCH(V, E, c, d, X )
        attempt insert X' into POOL
        X'' ∈ RAND(POOL)
        X_PR = PATHRELINK(X' , X'' )
        attemp to insert X_PR into POOL
        }
    }
    compute perturbations & update d'
}
X* = best solution in POOL
X* = VNS(V, E, c, d, X* )
return X*
```

A cutting planes algorithm: Lower bounds

- Integer programming formulation
- Cutting planes algorithm
- Preprocessing to reduce input graph size
- Implementation details

Integer programming formulation

- $x_e = 1$ iff edge $e \in T$ (real-valued)
- $y_v = 1$ iff vertex $v \in T$ (real-valued)
- Polyhedral region P
  - $x(E) = y(V) - 1$
  - $x(E(S)) \leq y(S \setminus \{s\}),\ s \in S,\ S \subseteq V$
  - $0 \leq x_e \leq 1,\ e \in E$
  - $0 \leq y_v \leq 1,\ v \in V$
- Integer programming formulation:

  minimize $\Sigma_{e \in E}\, c_e x_e + \Sigma_{v \in V}\, d_v (1 - y_v)$

  subject to: $(x_e, y_v) \in P \cap (R^{|E|}, Z^{|V|})$

$z(S) = \Sigma_{s \in S}\, z_s$

Integer programming formulation

- Region P: follows directly from SPG formulation of Goemans (1994), Lucena (1991), and Margot, Prodon, and Liebling (1994)
- $x(E) = y(V) - 1$: number of selected edges must equal required number of edges for spanning tree of implied subgraph
- $x(E(S)) \leq y(S \setminus \{s\}), s \in S, S \subseteq V$: generalized subtour elimination constraints (GSECs) $\Rightarrow$ solution is cycle-free
- Set of feasible solutions: all trees of G
- Lower bound to integer program can be computed by solving linear programming relaxation of integer program

Solving the linear programming relaxation

- LP relaxation:

  minimize $\sum_{e \in E} c_e x_e + \sum_{v \in V} d_v (1 - y_v)$

  subject to: $(x_e, y_v) \in P$

- Exponentially many GSECs:
  - initially exclude some or all of them from P: $P_1 \supseteq P$
  - optimize over P_1
  - adequate choice of P_1:
    - $x(E) = y(V) - 1$
    - $0 \leq x_e \leq 1,\ e \in E$
    - $0 \leq y_v \leq 1,\ v \in V$

Solving the linear programming relaxation

minimize $\Sigma_{e \in E} c_e x_e + \Sigma_{v \in V} d_v (1 - y_v)$

subject to: $(x_e, y_v) \in P_1$

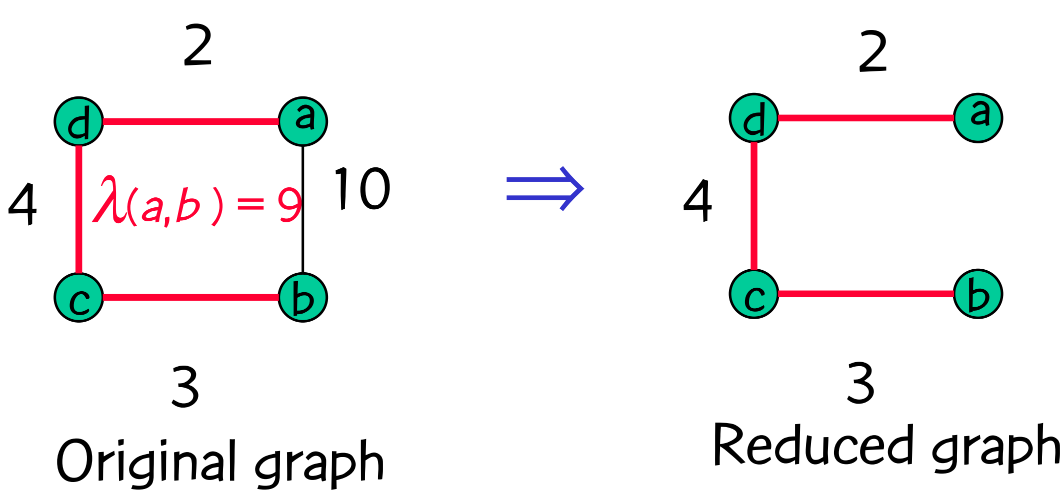
- Optimal (x^*, y^*): its cost is a valid lower bound for the prize-collecting Steiner problem
- Separation problem: Find one or more GSECs that are violated by (x^*, y^*) or determine that no such inequality exists
  - Solved as $|V|$ max-flow problems
  - Introduce violated GSECs as cutting planes
  - Re-optimize using dual simplex method

Preprocessing to reduce input graph size

- *A* reduction operator transforms G into a smaller graph G' such that the values of the optimal solutions of the integer programs defined on these two graphs are equal.
- Reduction tests: adapted from SPG tests of Duin (1994)
  - Shortest path test
  - Cardinality-1 test
  - Cardinality-2 test
  - Cardinality larger than 2 test

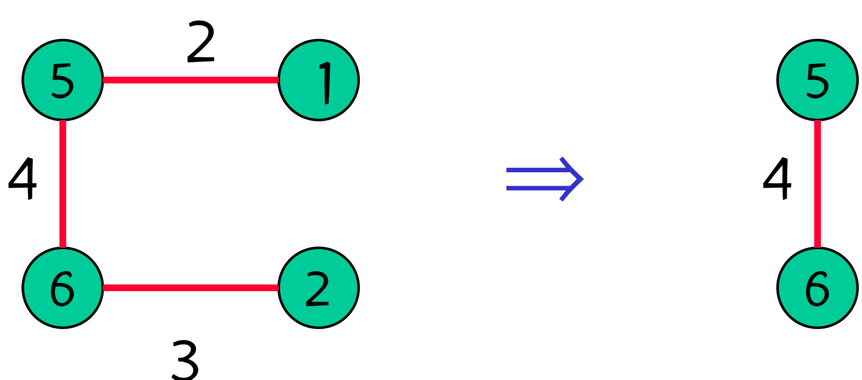
Preprocessing to reduce input graph size

- Shortest path test: Let $\lambda(u,v)$ be the length of the shortest path between vertices u and v.
- If $\lambda(u,v) < c_{uv}$, then edge (u,v) can be eliminated from G



Preprocessing to reduce input graph size

- Cardinality-1 test: Let vertex $v \in V$ have edge cardinality 1 (edge e is the only edge incident to v).
- If $c_e > d_v$, then vertex v can be eliminated from G

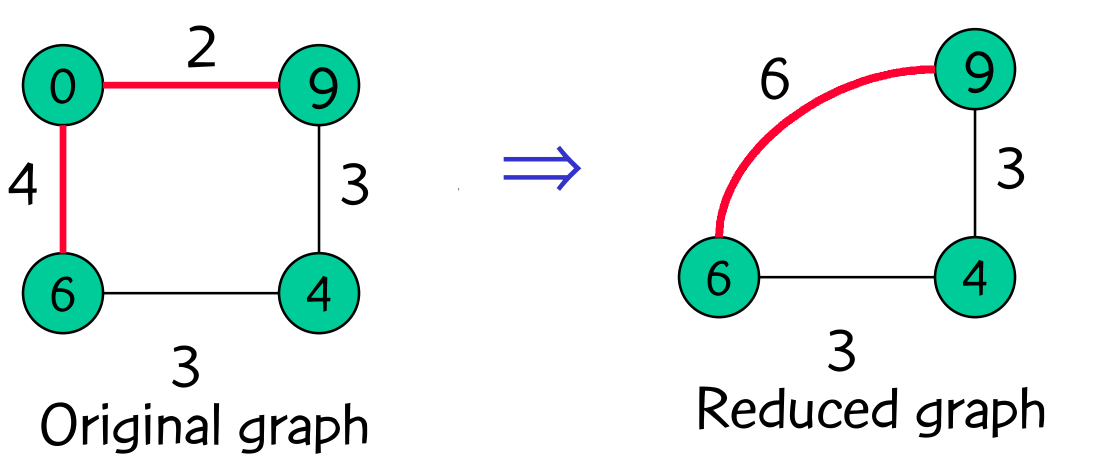


Original graph

Reduced graph

Preprocessing to reduce input graph size

- Cardinality-2 test: Let vertex $v \in V$ have edge cardinality 2 (edges incident to v are $e_1 = (v, v_1)$ and $e_2 = (v, v_2)$)
- If $d_v = 0$, either these two edges appear together in an optimal solution or neither does.
- Pseudo-eliminate v: replace v, e_1, and e_2 with edge (v_1, v_2) with weight $c(v, v_1) + c(v, v_2)$



Implementation details

- Most basic form of PCSPG solution is a single, isolated, positive penalty vertex
  - Easy to compute: max $\{d_v : v \in V\}$
  - We can set aside single vertex solutions and deal only with solutions of one or more edges

Restrict P with constraints

$$x(E(\delta(v))) \geq y_v \quad \text{if } d_v > 0$$

$$x(E(\delta(v))) \geq 2y_v \quad \text{if } d_v = 0$$

Computational results

- 114 test problems
  - From 100 nodes & 284 edges
  - To 1000 nodes & 25,000 edges
  - Three classes:
    - Johnson, Minkoff, & Phillips (1999) P & K problems
    - Steiner C problems (derived from SPG Steiner C test problems in OR-Library)
    - Steiner D problems (derived from SPG Steiner D test problems in OR-Library)

Computational results

- Lower bounding
  - Runs were done on an SGI (with 28 196 MHz MIPS R10000 processors and 7.6Gb of main memory)
  - Each run done on a single processor
  - Fortran
    - Cutting planes algorithm
    - Rather outdated XMP package of Marsten (1981) for solving the LPs
    - Package of Goldfarb & Grigoriadis (1988) to solve the separation max flow problems

Computational results

- Heuristic
  - Runs were done on a 400 MHz Pentium II with 32 Mb of main memory under Linux
  - C programming language (gcc)
    - Goemans & Williamson implementation of Johnson, Minkoff, and Phillips (1999)
    - Iterative improvement, path relinking, & VNS
  - Parameters
    - 500 multi-start iterations
    - Perturbation: $\alpha = 20$ and $a = 1.0$
    - VNS: MAXTRY = 10
    - Path relinking: $\beta = 0.04\ |V|$ and pool size = 10
    - Alternate between perturbation schemes

Computational results

lower bounds

- Cutting planes algorithm
  - Found optimal LP solutions in 97 of the 114 test problems (85%)
  - Found tight lower bounds (equal to best known upper bounds) in 104 instances (91%)
  - Of the 97 optimal LP solutions, 94 were integral. Each of the 3 fractional solutions was off of the best known upper bound by less than ½
  - On the 12 instances for which tight lower bounds were not produced, the bounds produced had at most a 1.3% deviation from the best known upper bounds
  - In 13 of the 114 instances, single vertex optima were found
  - In 7 instances the algorithm took over 100,000 seconds to converge to a lower bound. The longest run took over 10 CPU days.

Computational results

heuristic upper bounds

- Heuristic found
  - 89 of 104 known optimal values (86%)
  - solution within 1% of lower bound for 104 of 114 problems

Number of optima found with each additional heuristic

type	num	GW	+LS	+PR	+VNS	tot
C	38	6	2	25	3	36
D	32	5	6	10	4	25
JMP	34	8	6	12	2	28
	104					89

Computational results

heuristic upper bounds

Number of instances with given relative error

heuristic	< 1%	< 5%	<10%	max (%)
GW	7	22	29	36.4
+LS	17	34	37	11.1
+PR	35	38	40	9.1
+VNS	38	40	40	1.1

Problem type Steiner C

Computational results

heuristic upper bounds

Number of instances with given relative error

heuristic	< 1%	< 5%	<10%	max (%)
GW	7	21	31	38.5
+LS	22	33	36	30.8
+PR	34	38	39	10.5
+VNS	34	40	40	4.5

Problem type Steiner D

Computational results

heuristic upper bounds

Number of instances with given relative error

heuristic	< 1%	< 5%	<10%	max (%)
GW	15	31	34	6.6
+LS	24	34	34	3.7
+PR	32	34	34	3.4
+VNS	32	34	34	3.4

Problem type JMP

Concluding remarks

- Cutting planes algorithm produced tight lower bounds and feasible upper bounds for most instances.
  - Running times were high for most difficult instances
  - May be improved using a more up-to-date LP solver
- With substantially less computational effort, the heuristic produced optimal and nearly optimal solutions.
  - Running times for most difficult instances averaged about 10,000 seconds
  - Over 90% of solutions were within 1% of lower bound

Concluding remarks

- Online at my web site:
  - These slides:

http://www.research.att.com/~mgcr/talks/pcstp.pdf

  - A. Lucena & M.G.C. Resende, "Strong lower bounds for the prize-collecting Steiner tree problem in graphs," 2000

http://www.research.att.com/~mgcr/doc/pcspflp.pdf

  - S.A. Canuto, M.G.C. Resende, & C.C. Ribeiro, "Local search with perturbations for the prize-collecting Steiner tree problem in graphs," 1999

http://www.research.att.com/~mgcr/doc/pcstpls.pdf