Tutorial: Heuristic Search for Network Design

INFORMS Denver October 24, 2004

Ioannis Gamvros, Bruce Golden,
S. Raghavan, Daliborka Stanojević
University of Maryland

Complete paper in INFORMS Denver tutorial book

Heuristic Search for Network Design

- Wide variety of applications domains
  - Telecommunications, logistics, transportation, supply chain management.
- In practice problems are frequently NP-complete. Problems are large scale, and require real-time solutions.
- Motivates the need for good heuristics.

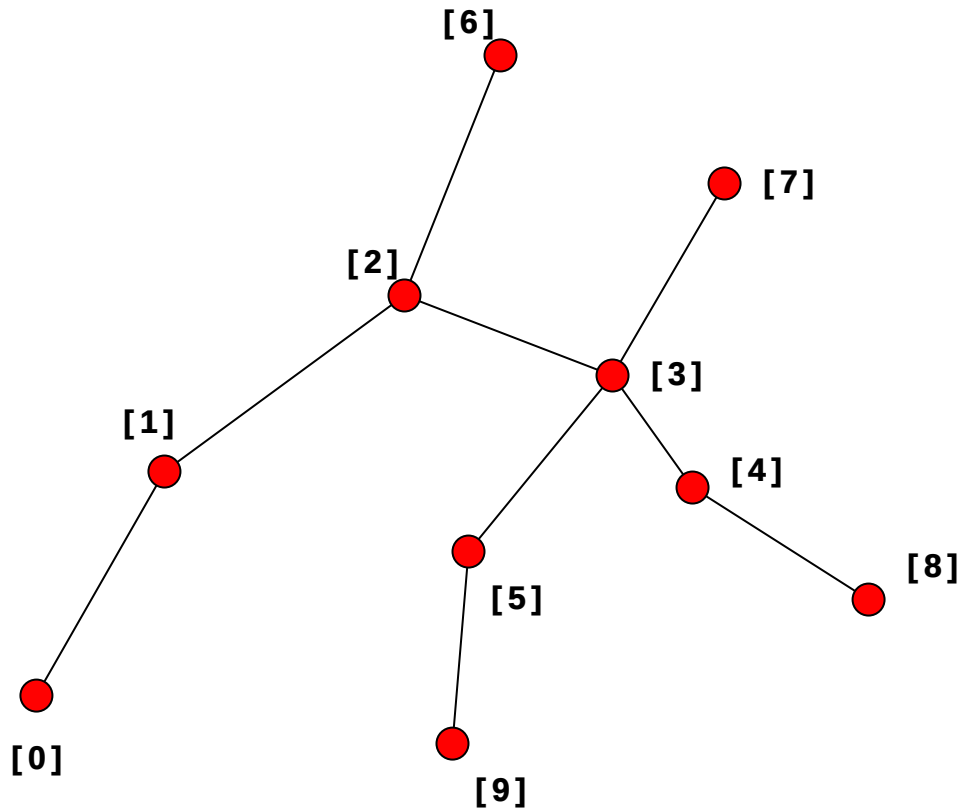
Goal of this Tutorial

- To provide the practitioner
  - an overview of variety of heuristic search techniques available
  - Some guidelines
  - EXAMPLES OF SUCCESSFUL APPLICATIONS OF HEURISTIC SEARCH TO NETWORK DESIGN.

Overview of Heuristic Search Techniques

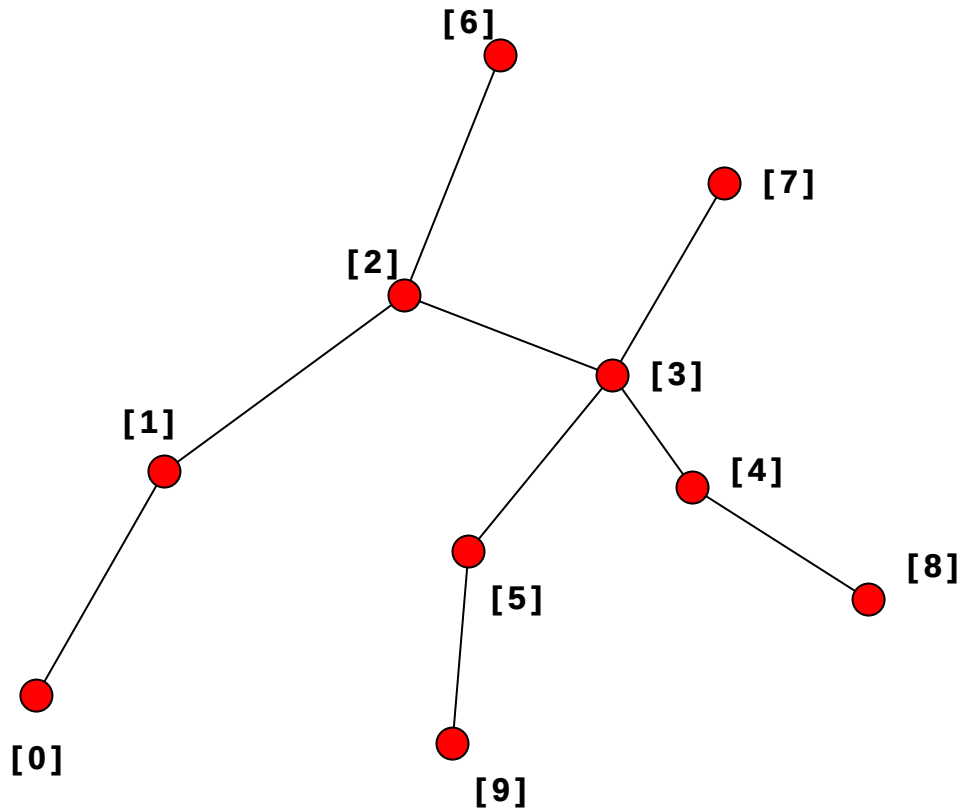
- Simple Heuristics
  - Construction
  - Deletion
  - Savings

Construction Heuristic



MST

Construction Heuristic



MST

Overview of Heuristic Search Techniques

- Simple Heuristics
  - Construction
  - Deletion
  - Savings
- Local Search

Local Search

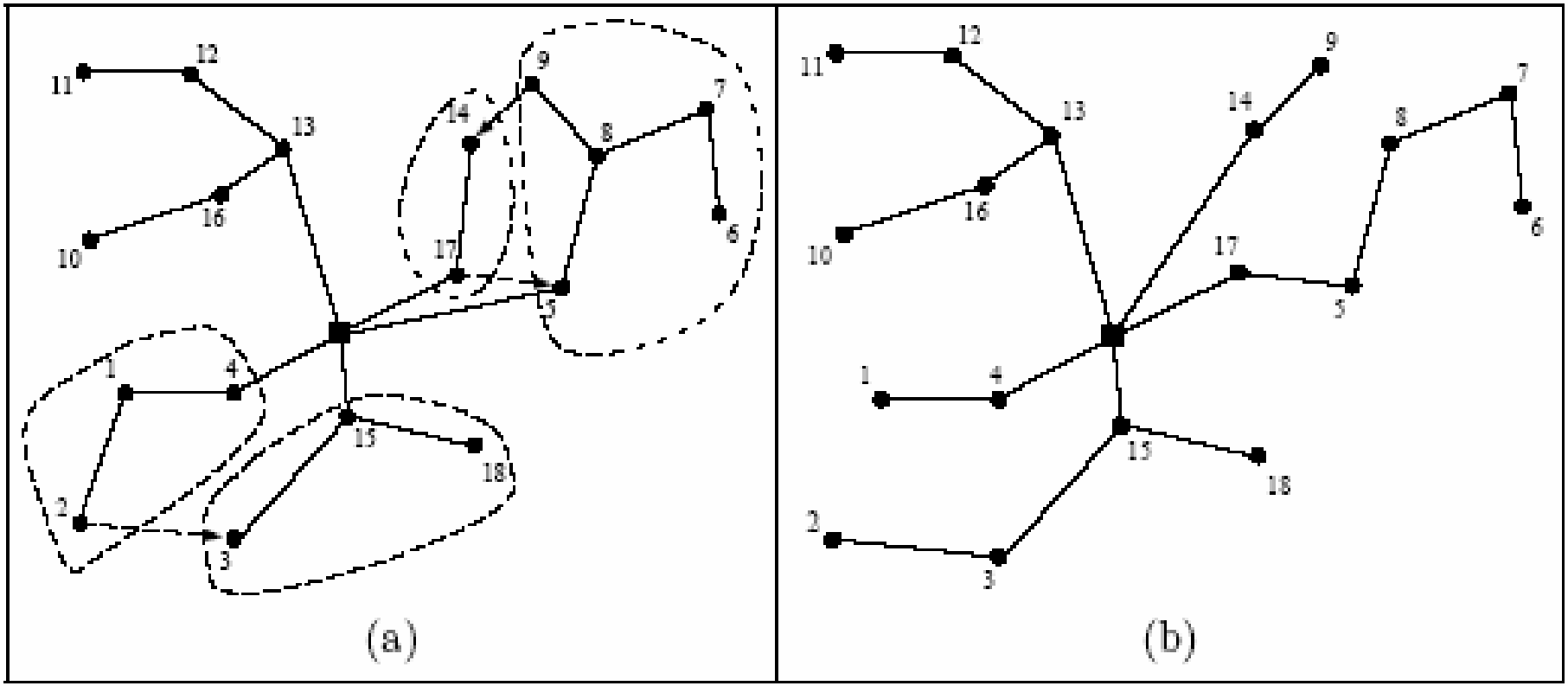


Figure 1.1. Two-exchange example for the CMST problem. (a) Original solution with the proposed exchanges. (b) Final solution after the exchanges.

Local Search

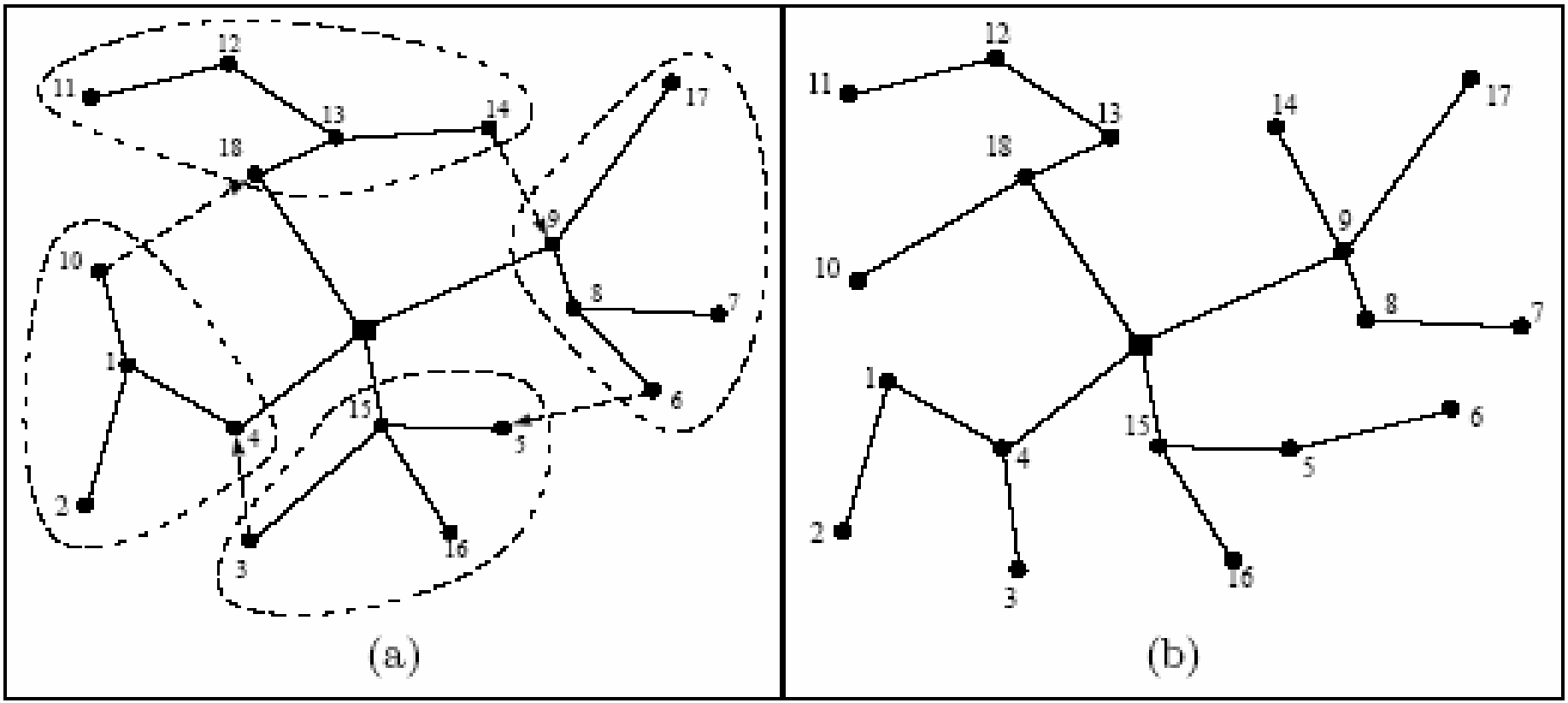


Figure 1.2. Multi-exchange example for the CMST. (a) Original solution with proposed cyclic exchange. (b) Final solution after exchanges.

Local Search

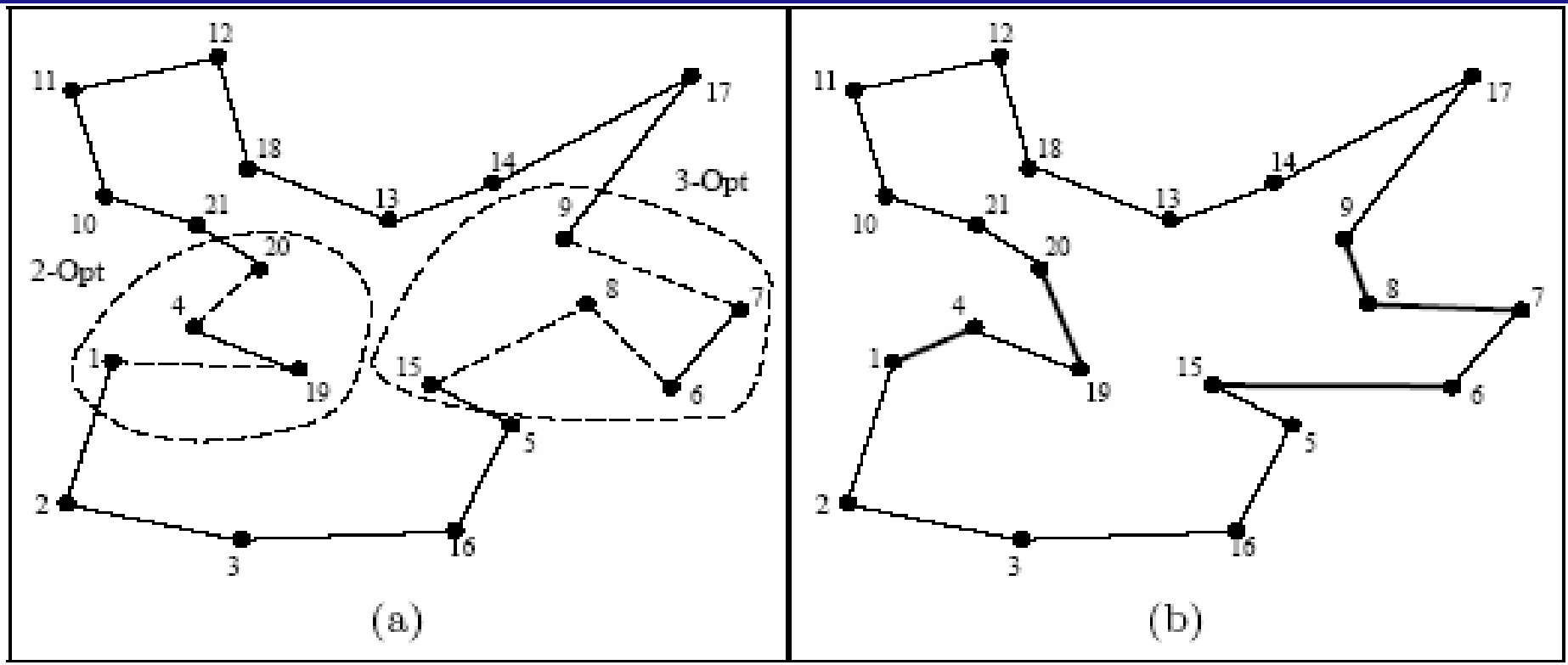


Figure 1.3. 2-opt and 3-opt example for the TSP. (a) Original solution (dashed edges will participate in the exchanges). (b) Final solution after the exchanges (the edges that have been added after the exchanges are noted in bold).

Local Search – Simulated Annealing

Begin
randomly generate an initial solution f_s
compute the cost of f_s, $C(f_s)$
set the effective temperature, T, to the initial value, T_0
while $T \neq T_N$ **do**
randomly select a neighbor, f_n, of the current solution, f_s
compute the cost $C(f_n)$ and $\Delta = C(f_n) - C(f_s)$
if $\Delta \leq 0$ **then**
$f_s \longleftarrow f_n$
else
with probability $e^{-\Delta/T}$, $f_s \longleftarrow f_n$
end if
update the temperature T
end while
end

Overview of Heuristic Search Techniques

- Simple Heuristics
  - Construction
  - Deletion
  - Savings
- Local Search
- Tabu Search
- Genetic Algorithms

Genetic Algorithms

```
Begin
    t ⟵ 0
    initialize P(t)
    evaluate P(t)
    while (not termination-condition) do
        t ⟵ t + 1
        select r solutions from P(t − 1)
        apply genetic operators on the r solutions
        insert the new r solutions to P(t)
        evaluate P(t)
    end while
end
```

Guidelines for Applying Heuristic Search Techniques

- Start simple! Develop simple heuristics to build intuition about the problem.
- Use knowledge gained in developing simple heuristics to develop search neighborhoods for use in a local search procedure. (strong and complex neighborhoods usually need simple search techniques. Simple neighborhoods benefit from the use of more sophisticated search techniques like simulated annealing, GRASP, and tabu search).
- For more challenging problems apply genetic algorithms. Suggest the use of local search within genetic algorithms (usually as mutation operators).

Heuristic Search for the Multi-Level Capacitated Minimum Spanning Tree (MLCMST) Problem

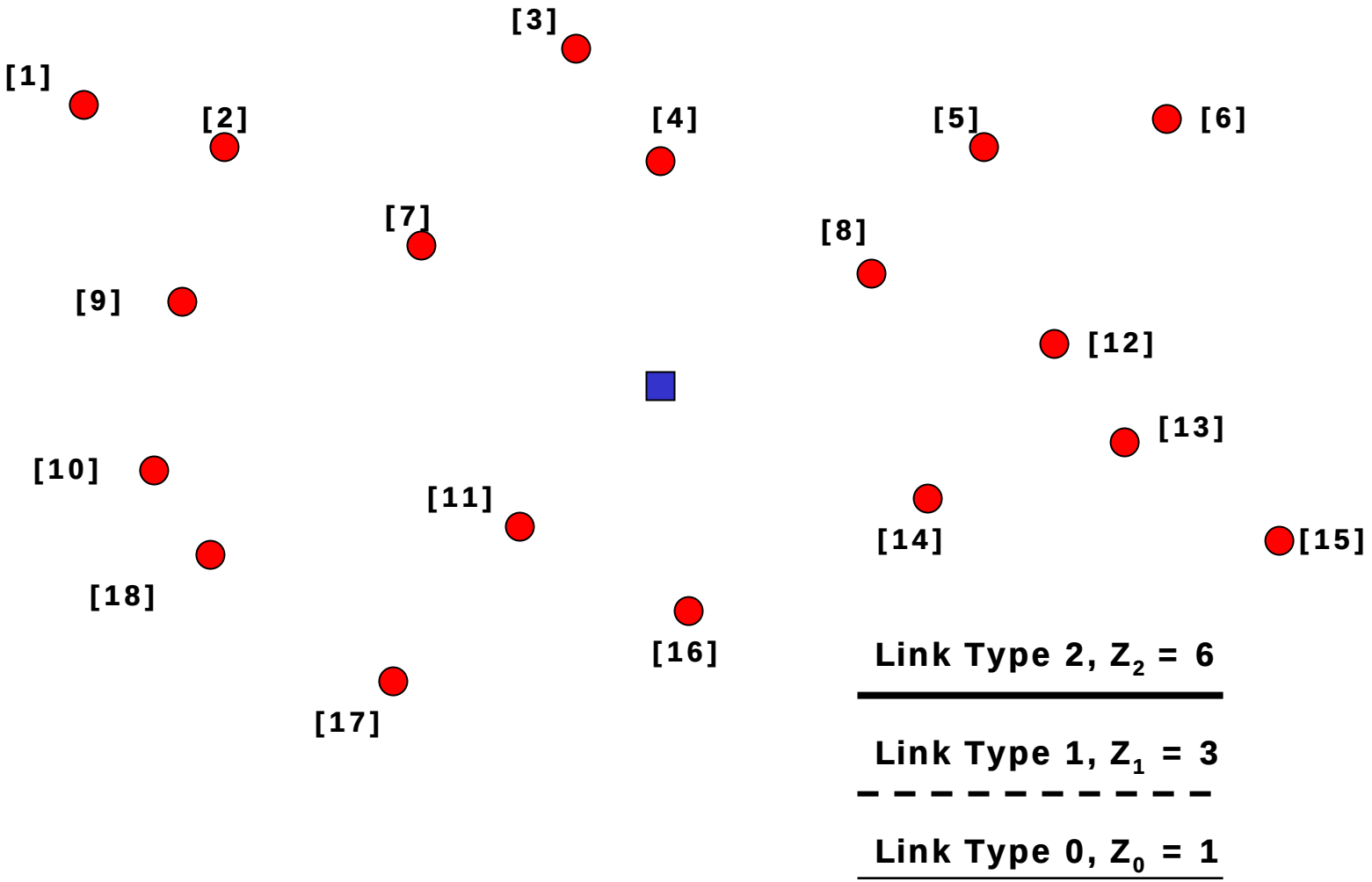
I. Gamvros
B. Golden
S. Raghavan

Robert H. Smith School of Business
University of Maryland, College Park

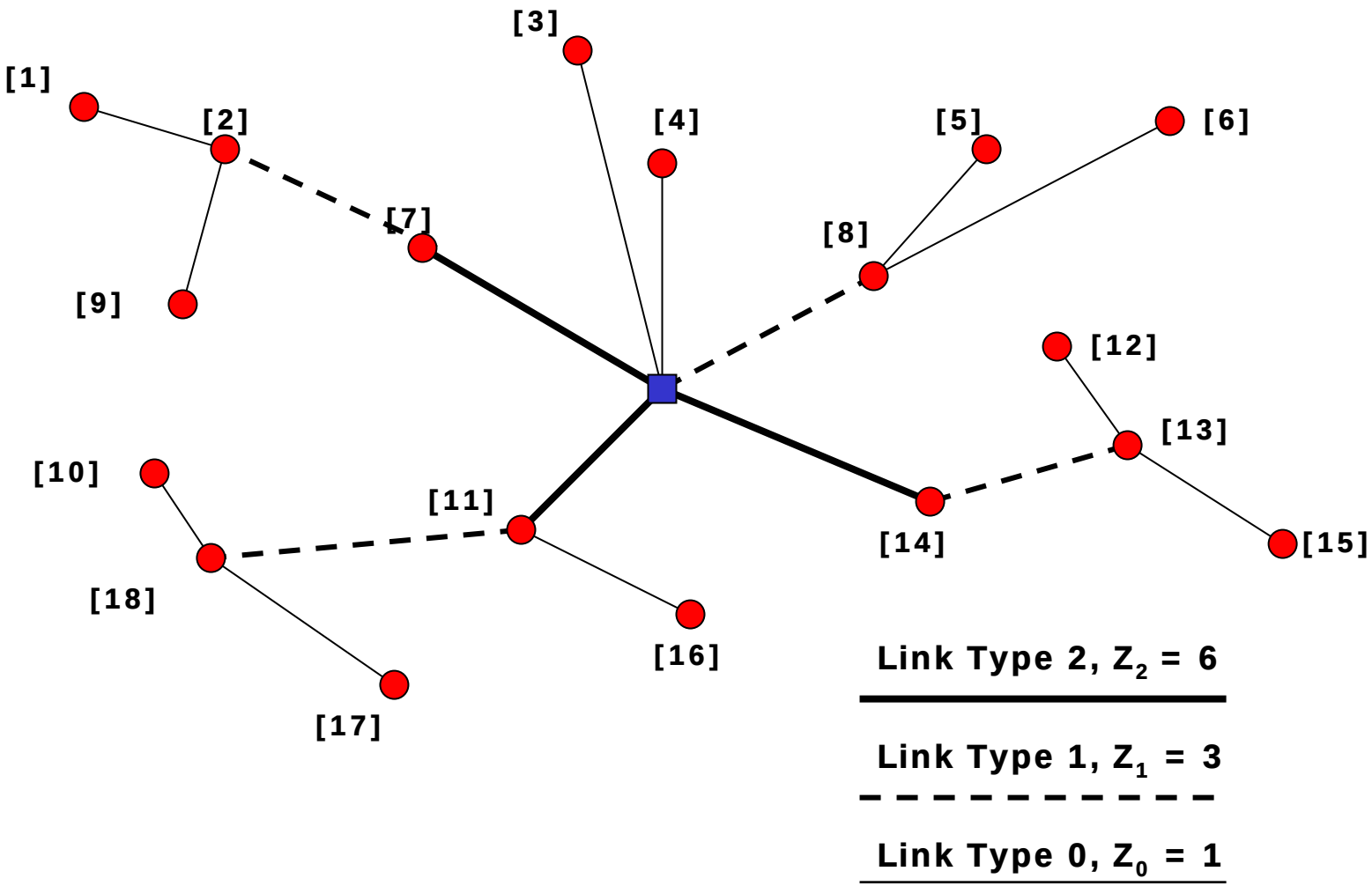
Agenda

- Problem Definition
- Savings Heuristic
- Local Search
- Genetic Algorithm
- Computational Results

Problem Definition



Problem Definition



Problem Definition

- Given a complete undirected graph G=(N,E), where N = {0,1,…n} is the node set and node 0 represents the central facility
- C_{ij}^{ℓ} the cost of connecting each pair of nodes i,j with link type ℓ.
- W_i weight of node i,
- Link Types 0,1,…L with capacities $Z_0 < Z_1 < \ldots < Z_L$,
- Find a minimal cost tree network that satisfies the traffic requirements of all nodes
- NP-hard since CMST is a special case

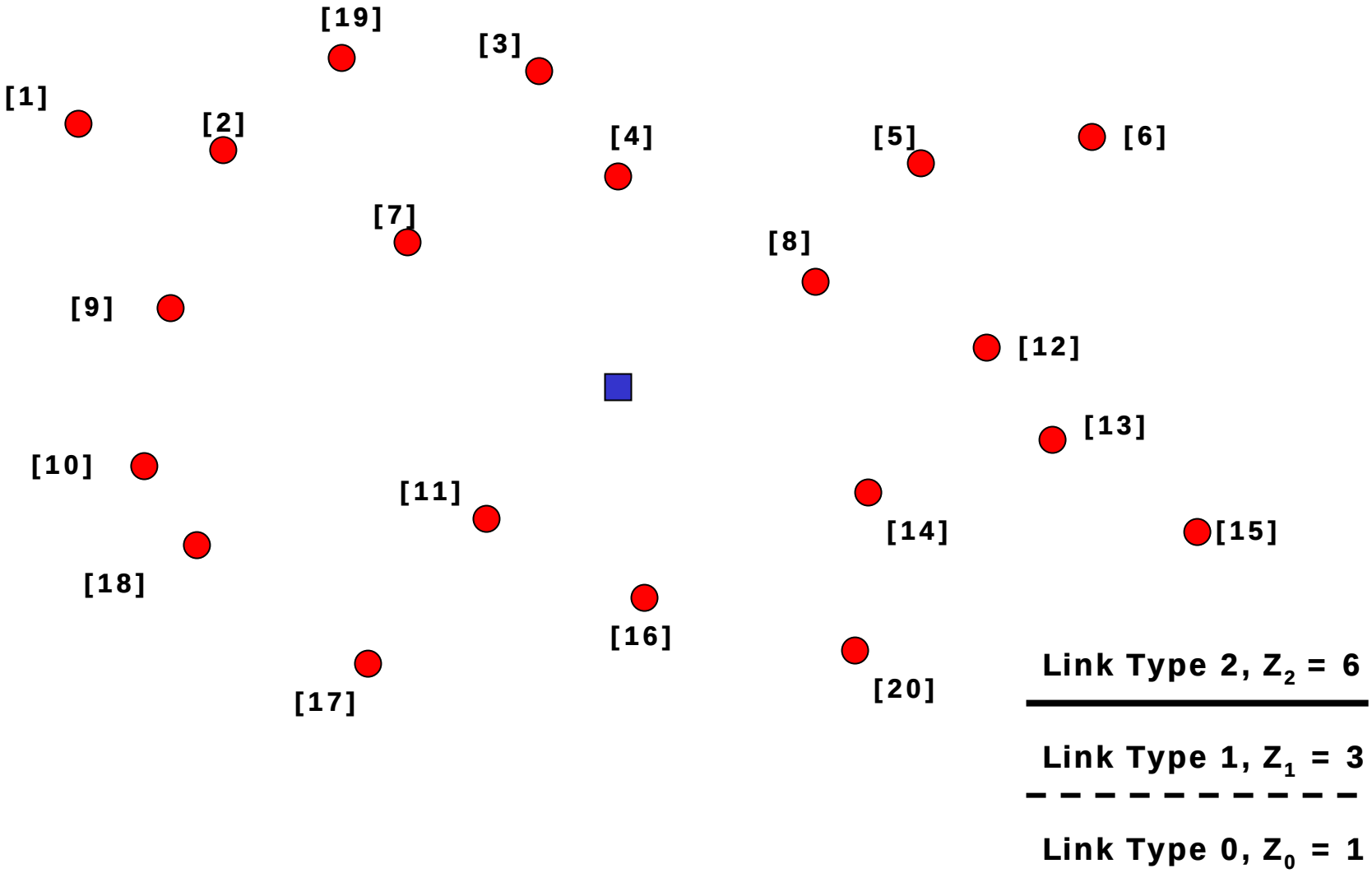
Problem Definition - Assumptions

- Assume only one link type between a pair of nodes,
- Homogeneous case, $W_i = k$ for every node i,
- We assume realistic cost functions that account for the economies of scale that usually exist. Specifically, we assume that:

$$C_{ij}^{y} \leq \alpha_{yx} C_{ij}^{x}$$

- where: $\alpha_{yx} = Z_y/Z_x$ and $x < y$.

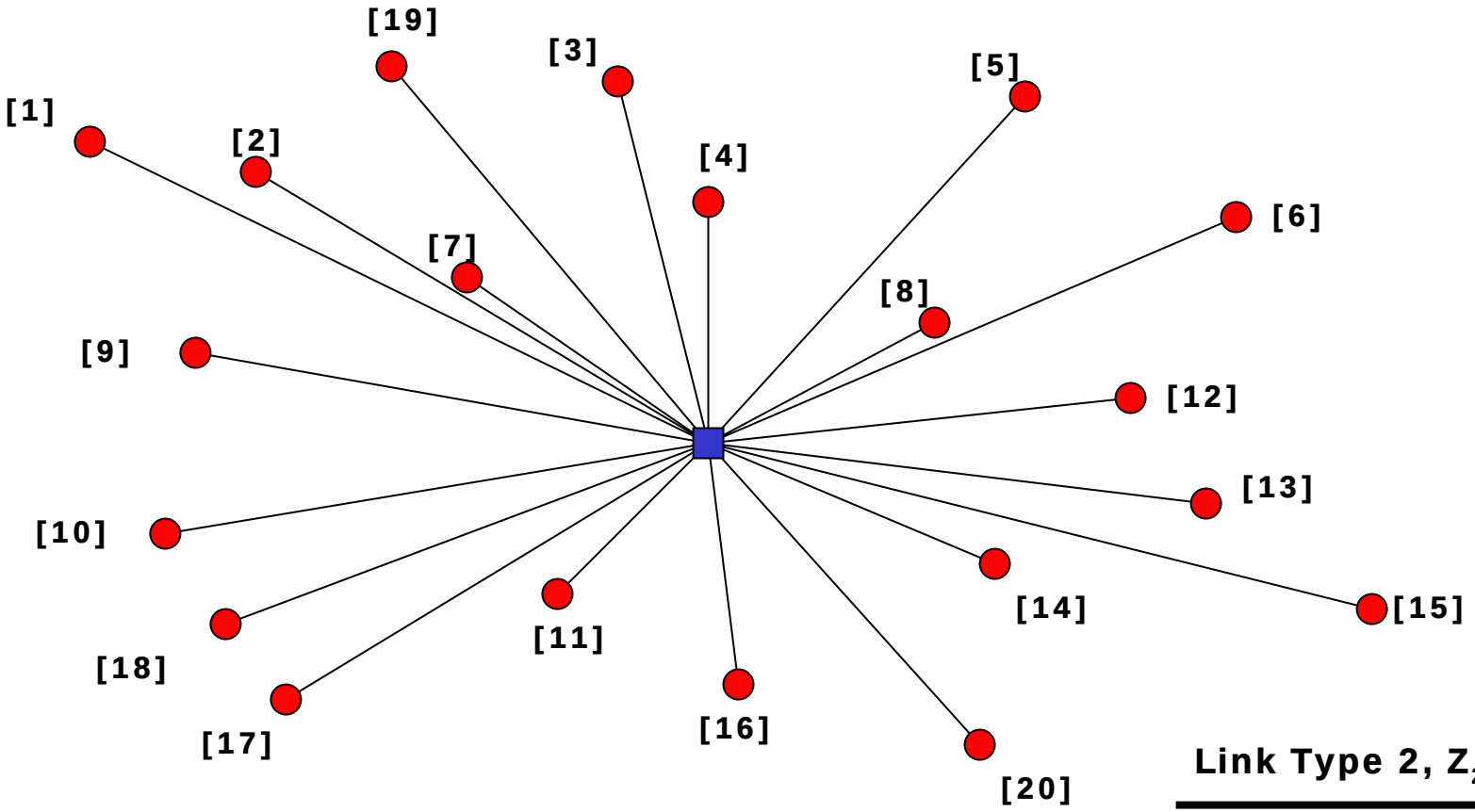
Savings Heuristic



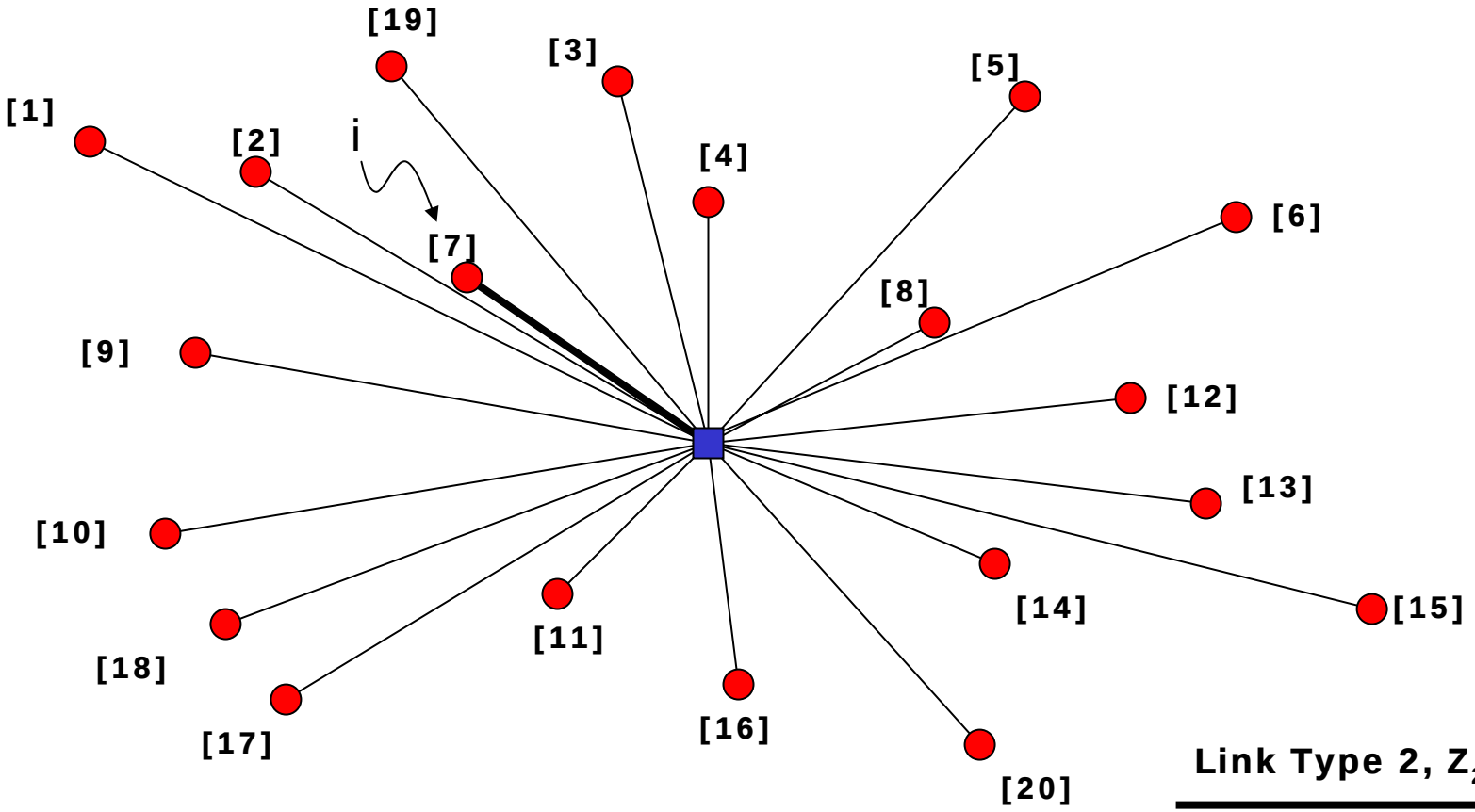
Savings Heuristic

- Start with a feasible star solution in which all nodes are connected to the central node with the lowest capacity link.
- For each node calculate the savings generated while upgrading its connection to the highest capacity link and connecting other nodes to it.

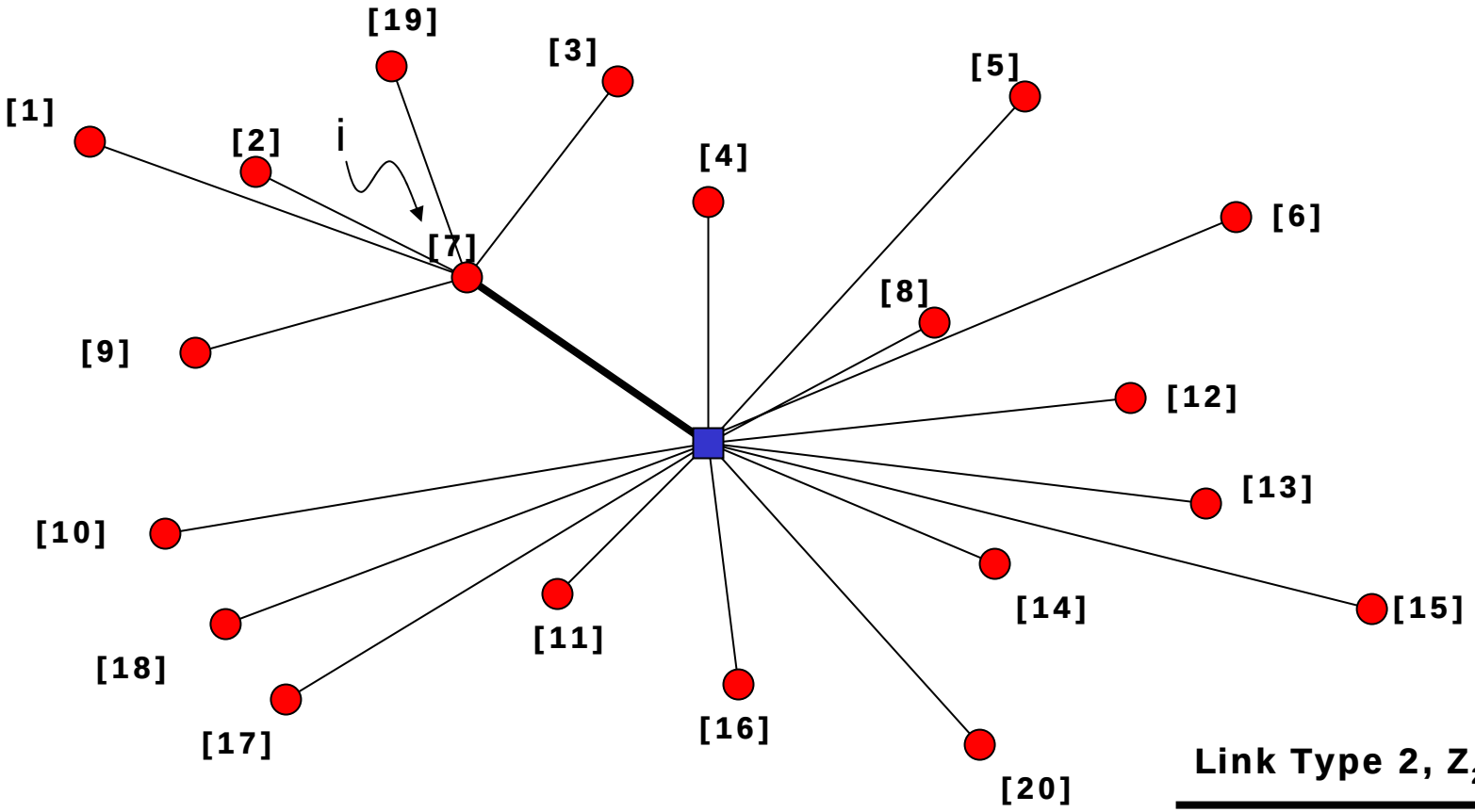
Savings Heuristic



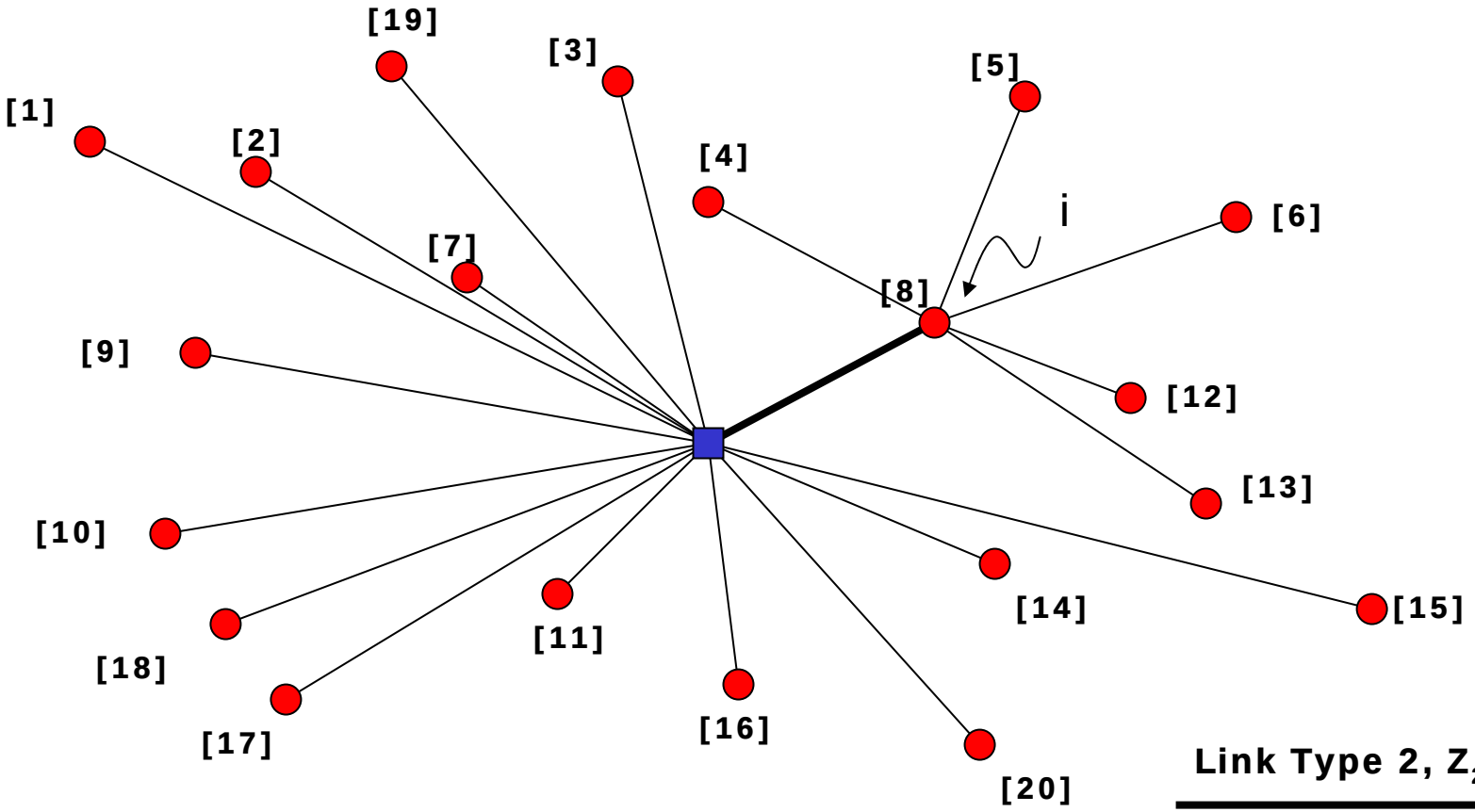
Savings Heuristic



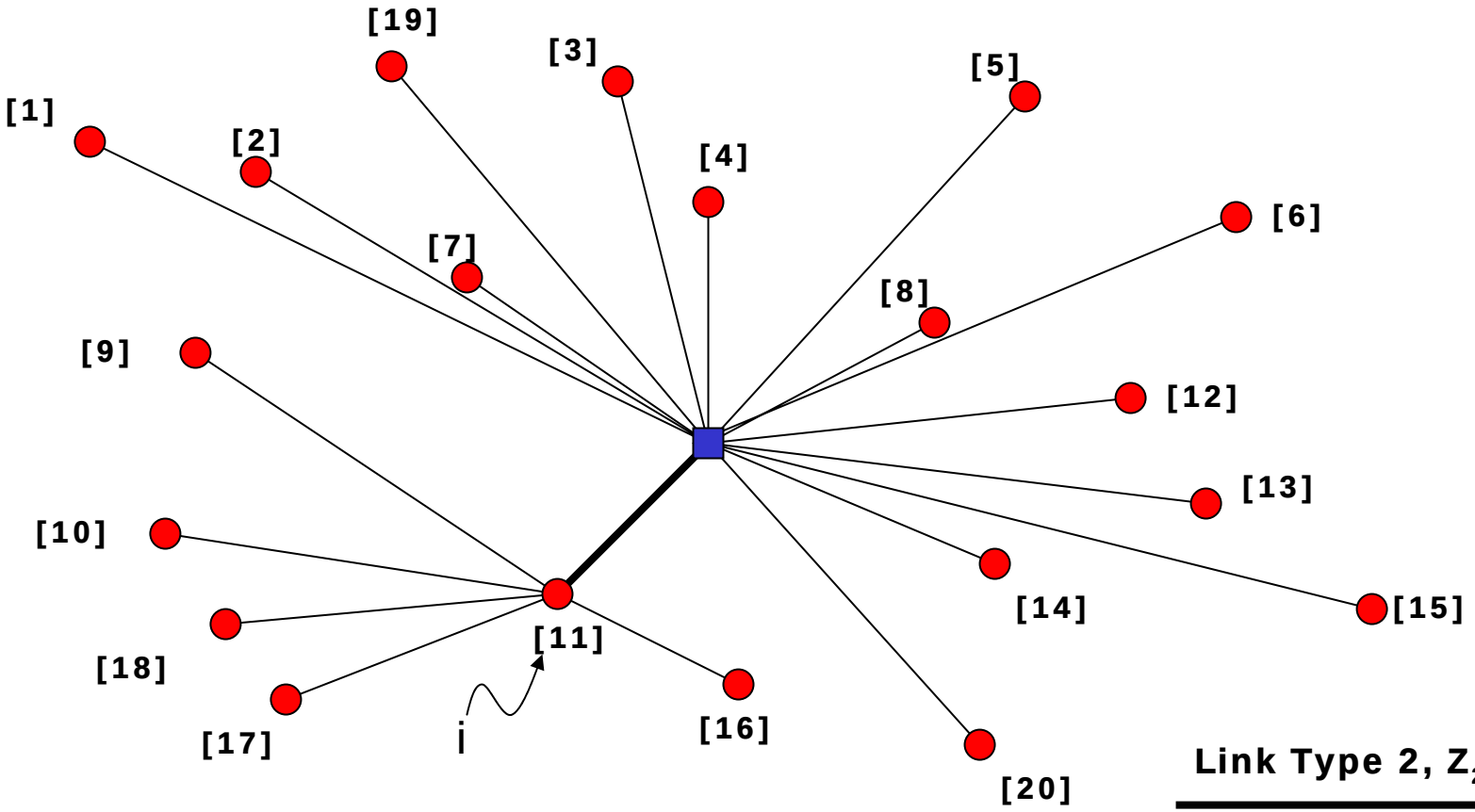
Savings Heuristic



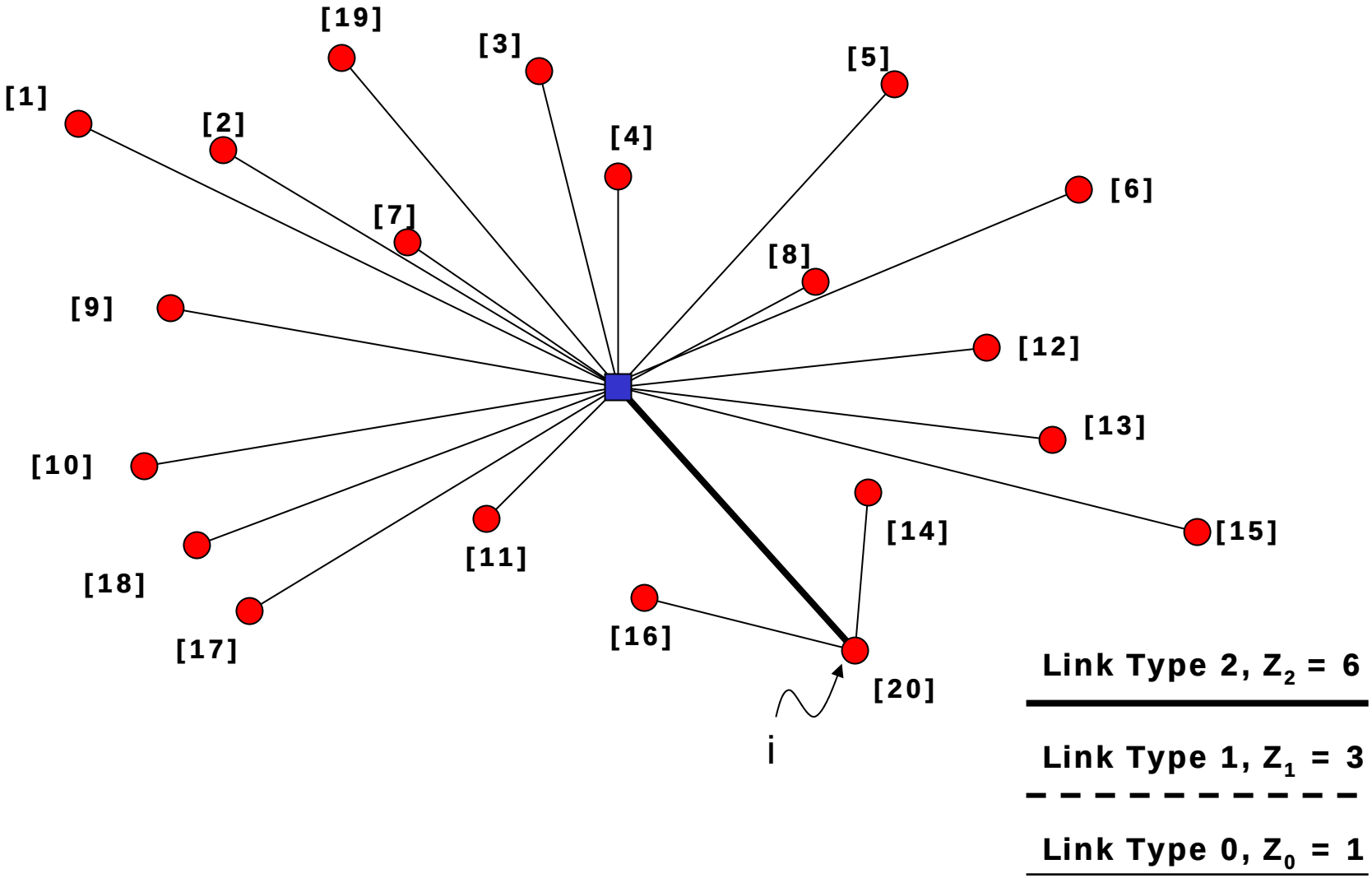
Savings Heuristic



Savings Heuristic



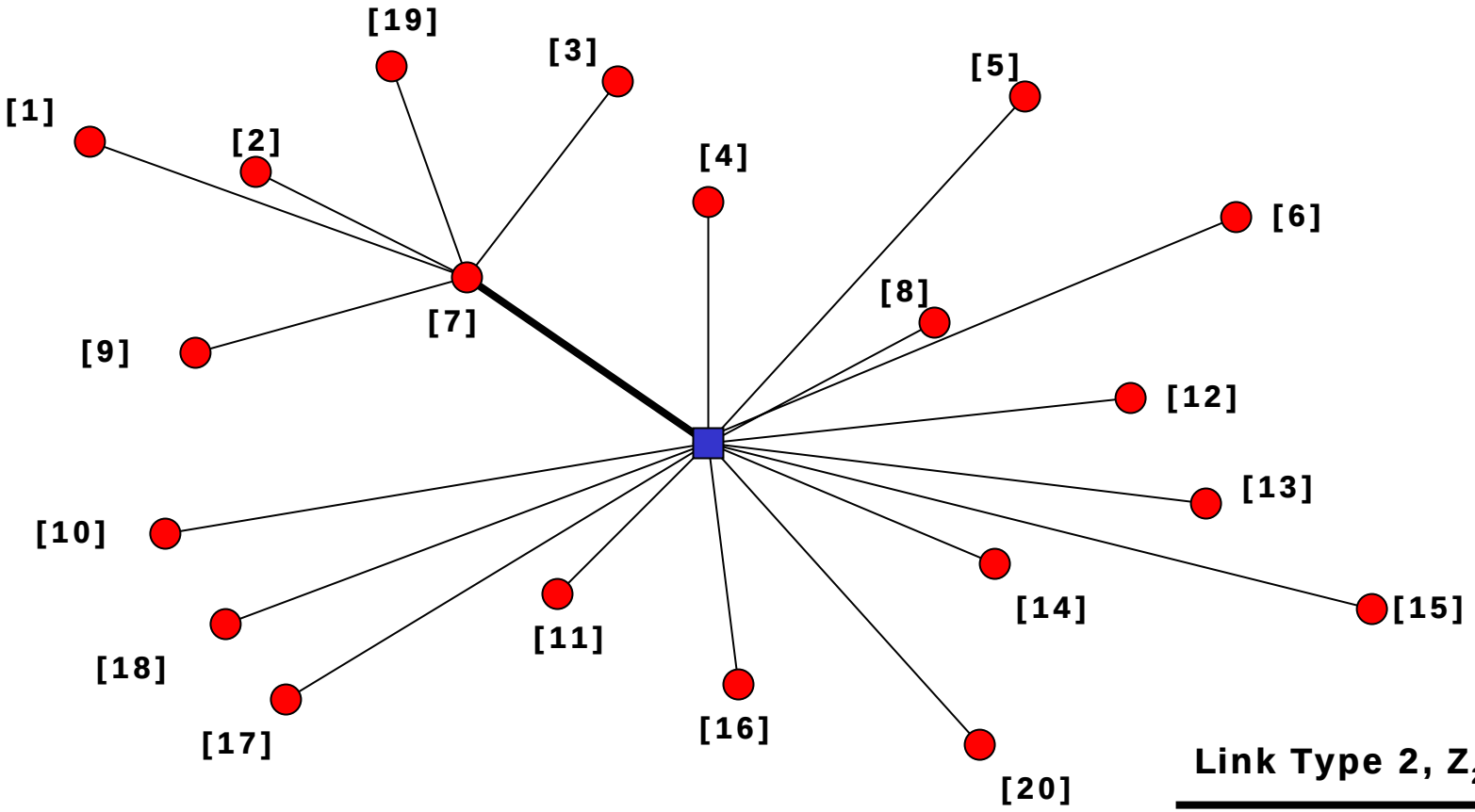
Savings Heuristic



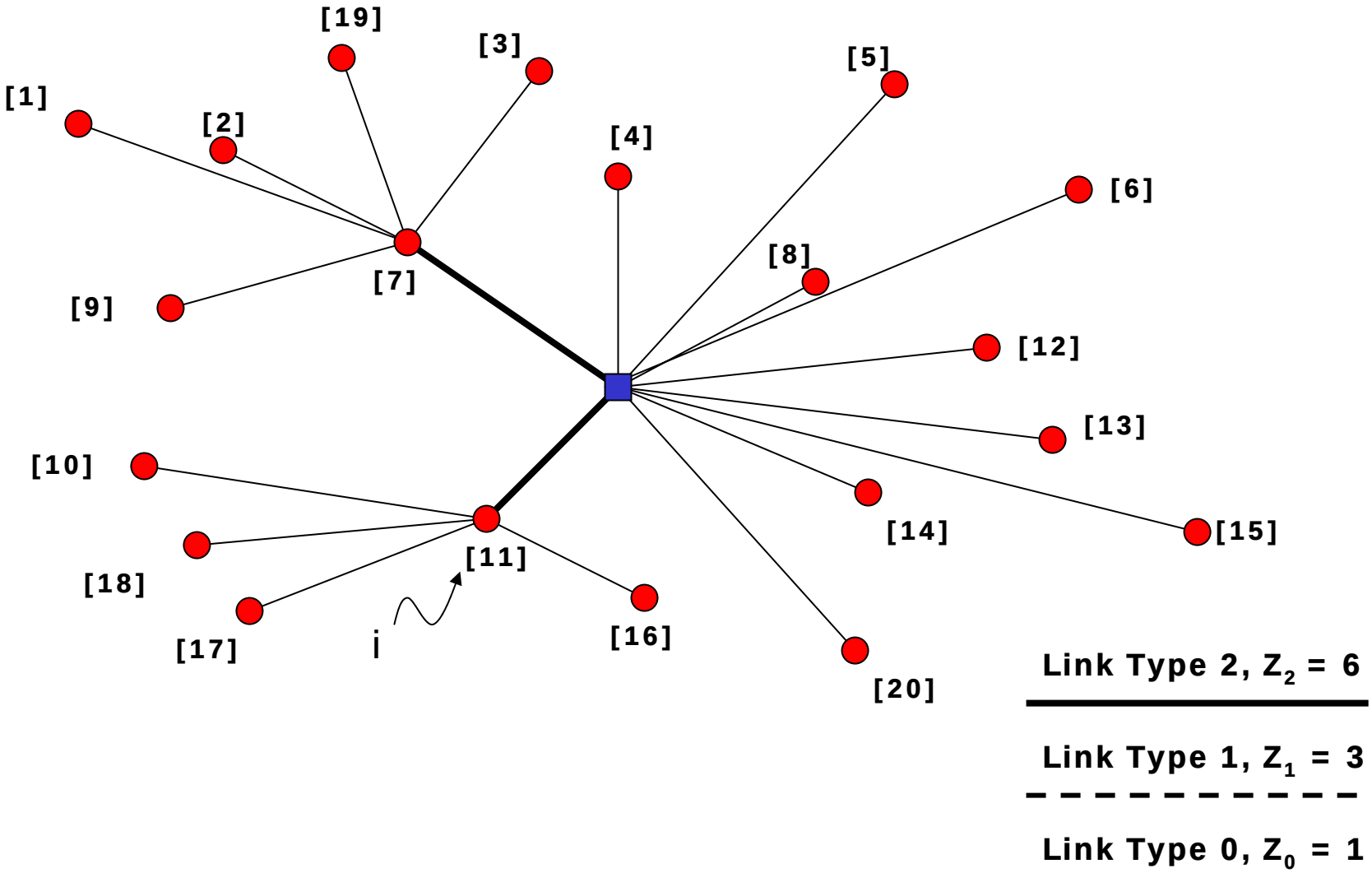
Savings Heuristic

- Select the node with the highest savings and implement the upgrade and reconnections.
- Repeat until there are no more savings by upgrading to the highest capacity link.

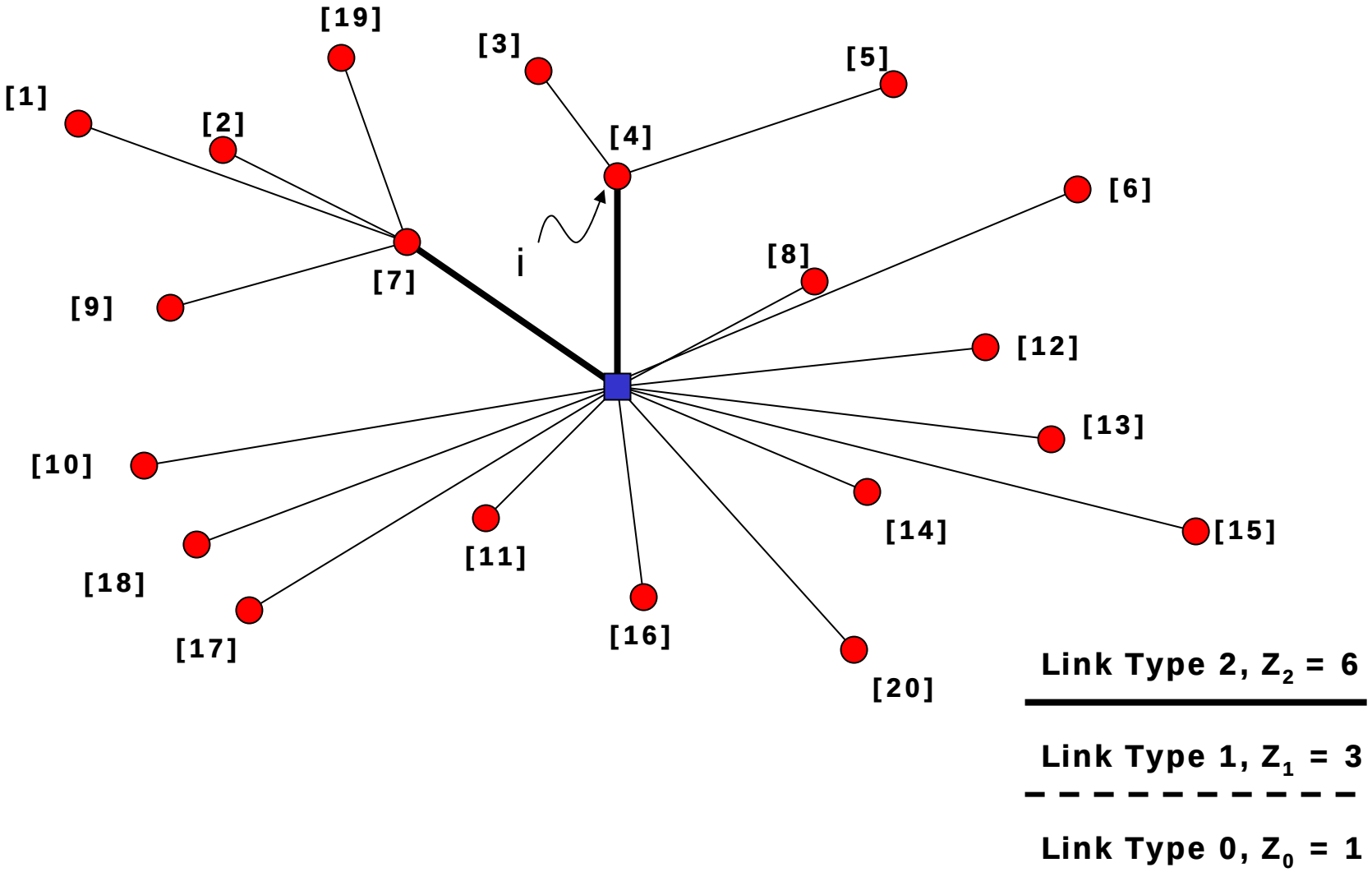
Savings Heuristic



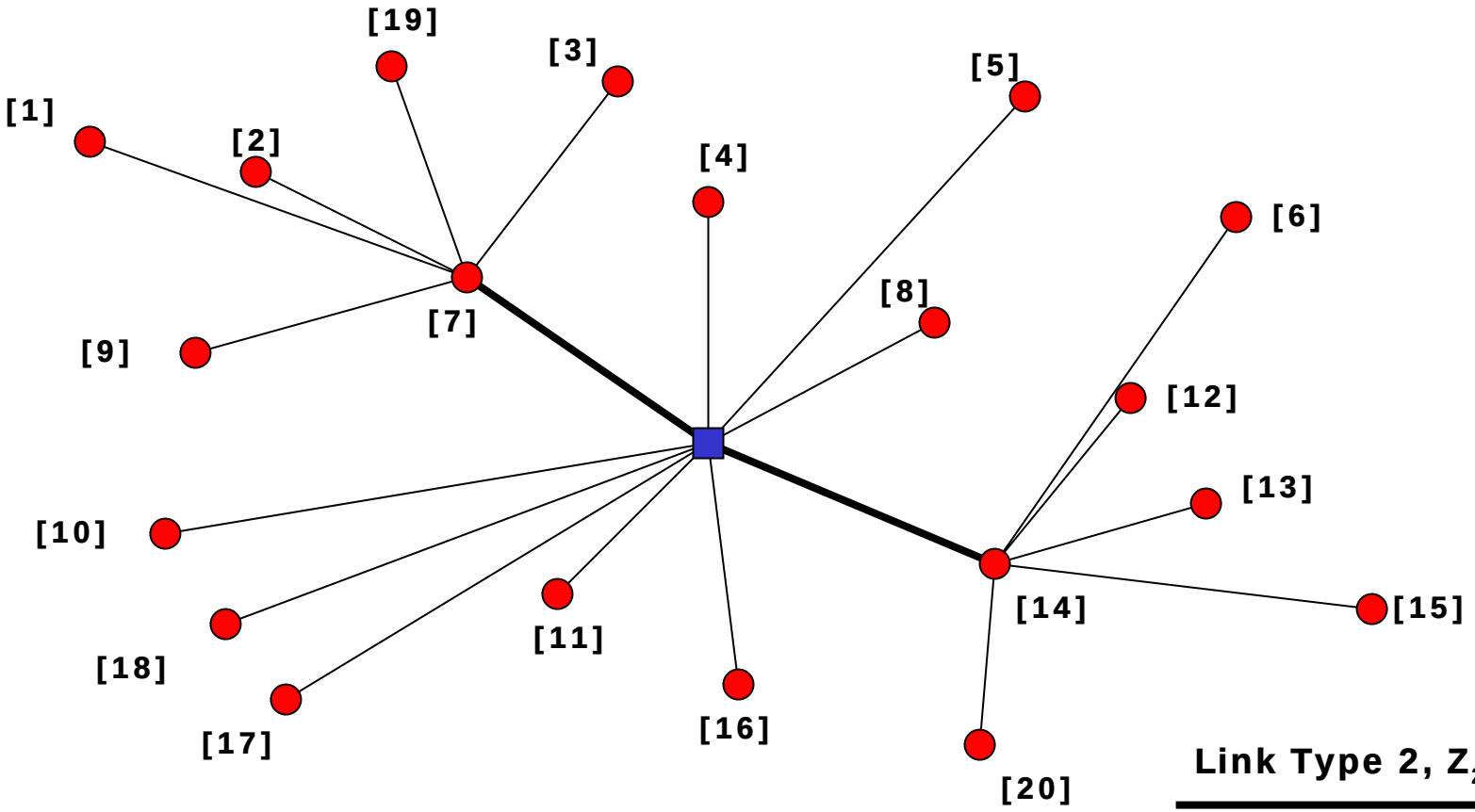
Savings Heuristic



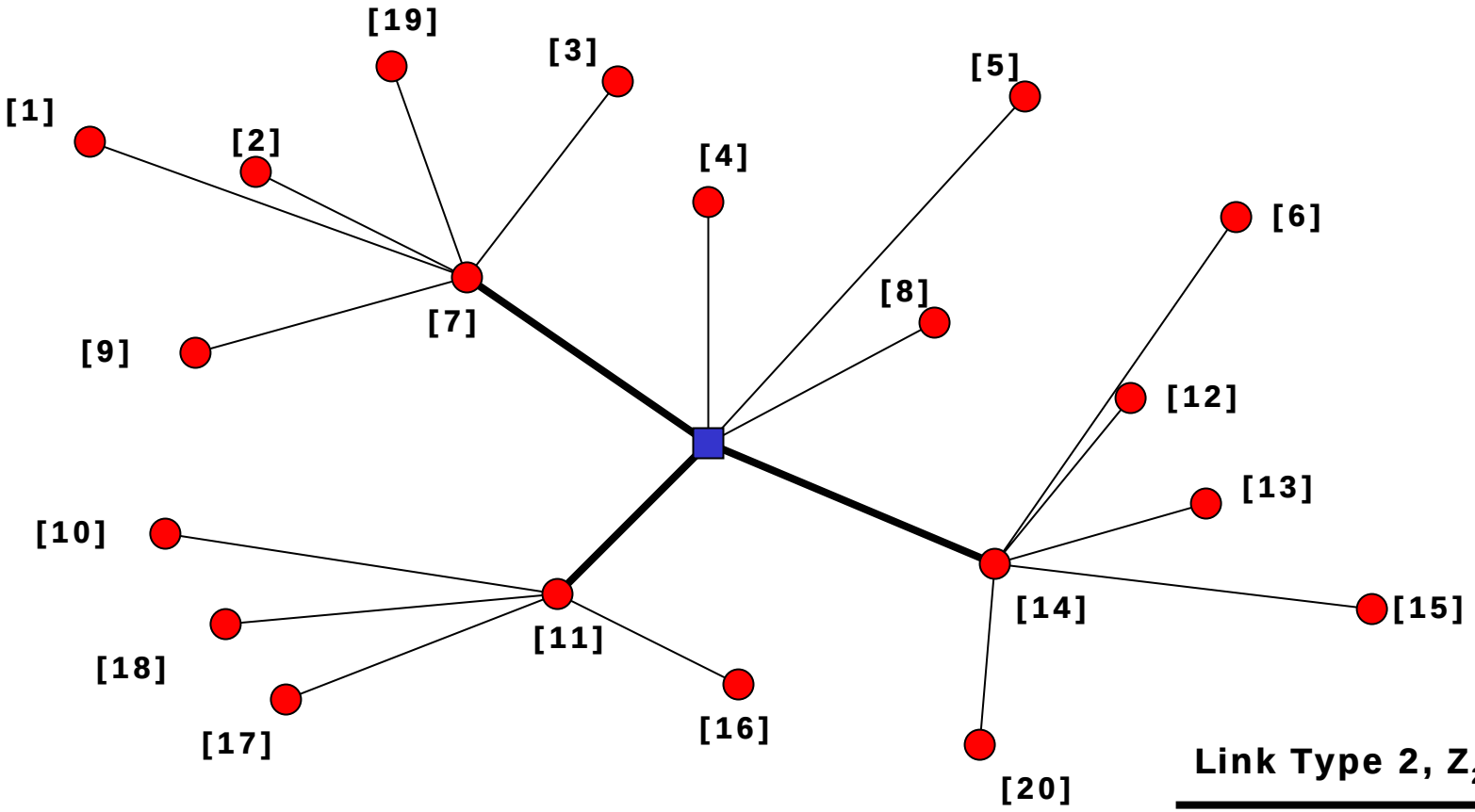
Savings Heuristic



Savings Heuristic



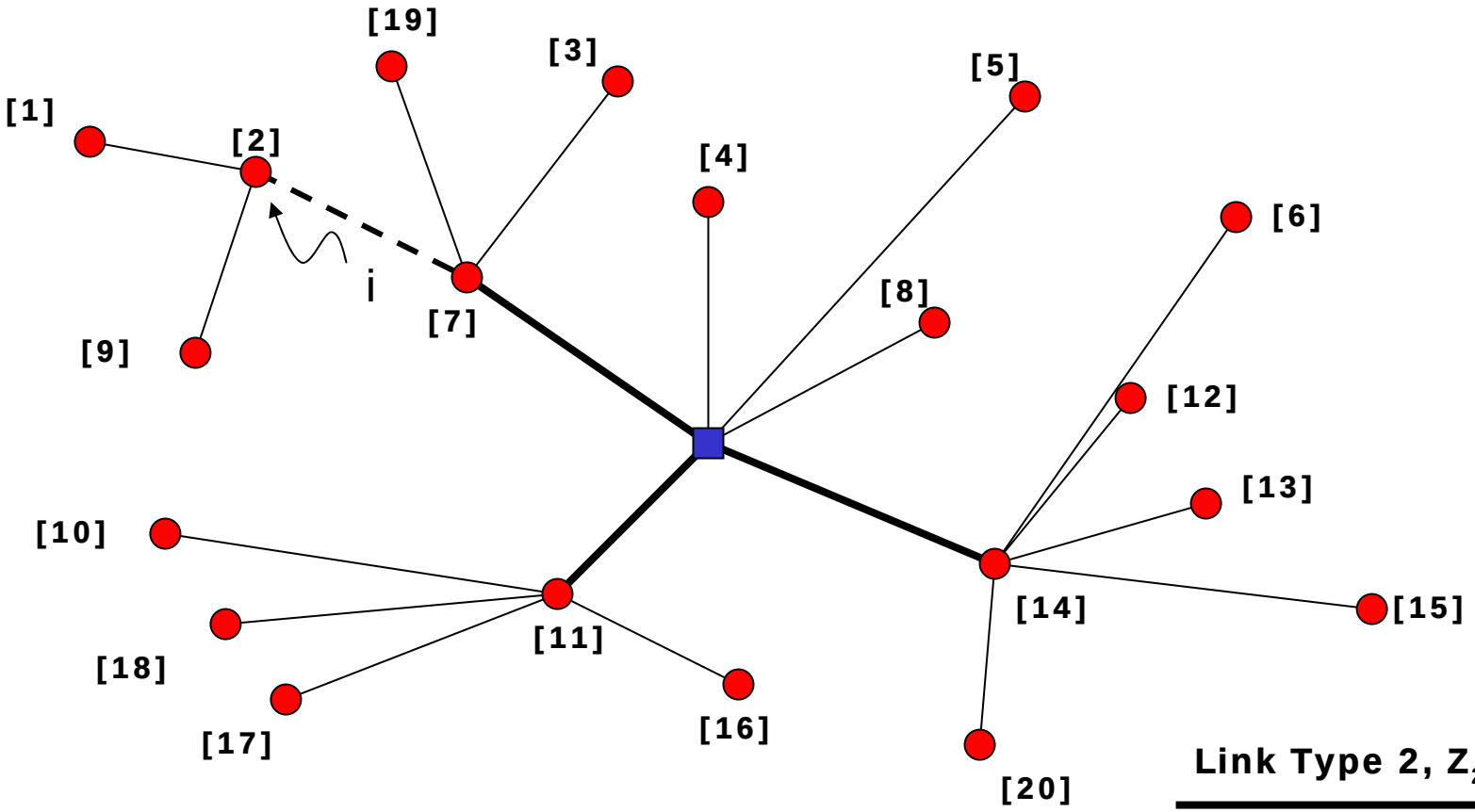
Savings Heuristic



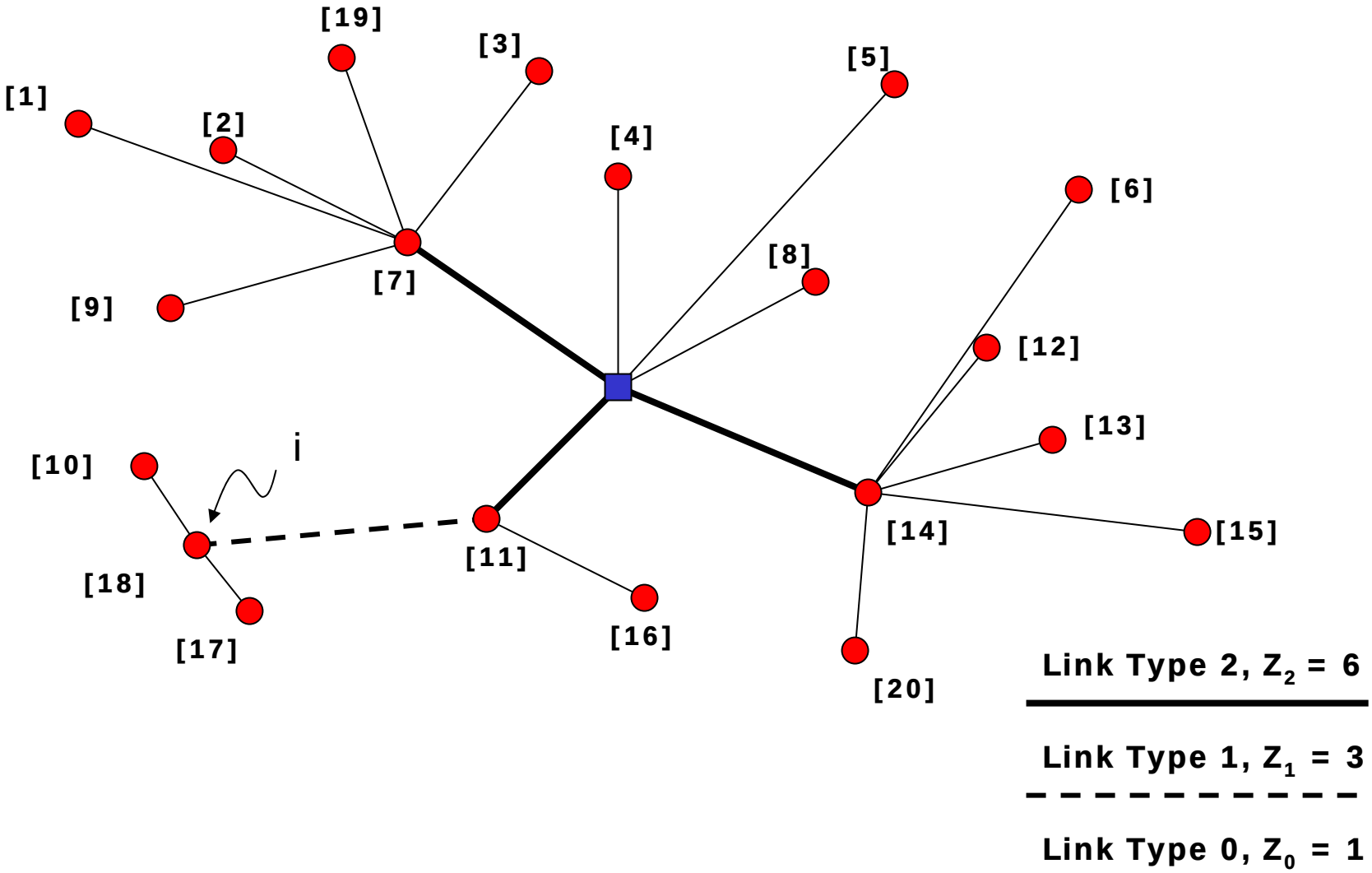
Savings Heuristic

- Determine savings generated while upgrading to the second highest capacity link and repeat for all link types.

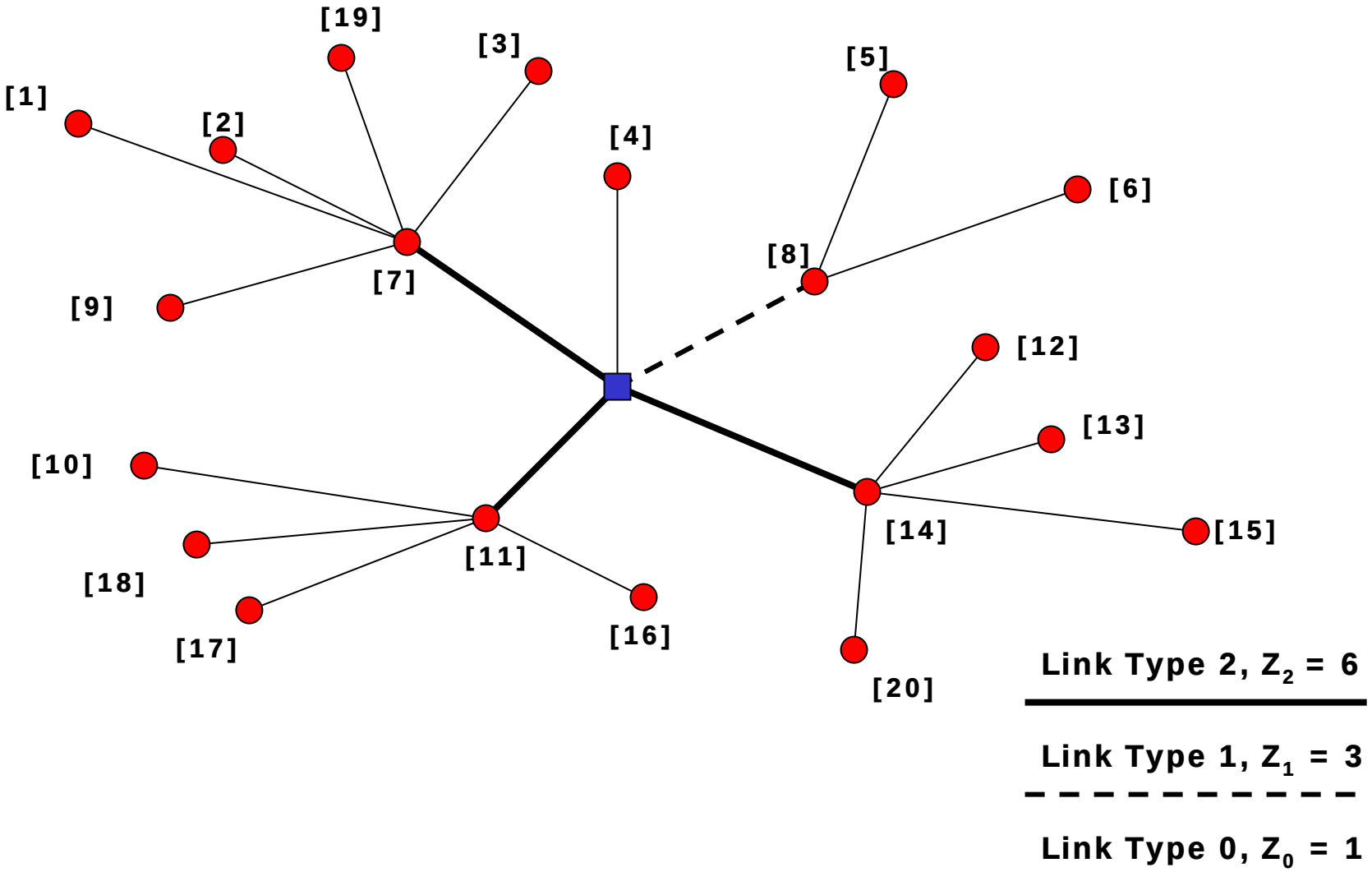
Savings Heuristic



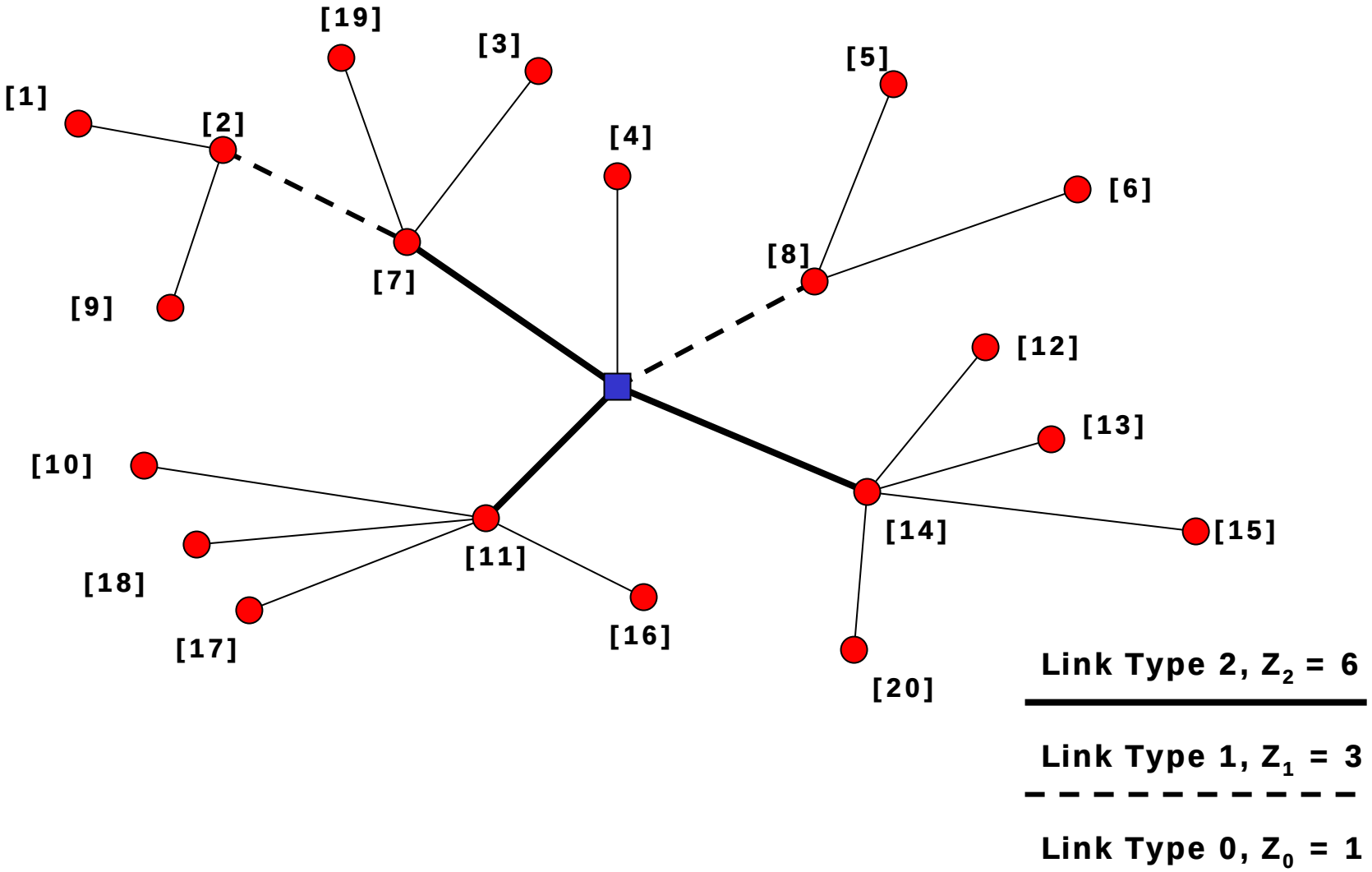
Savings Heuristic



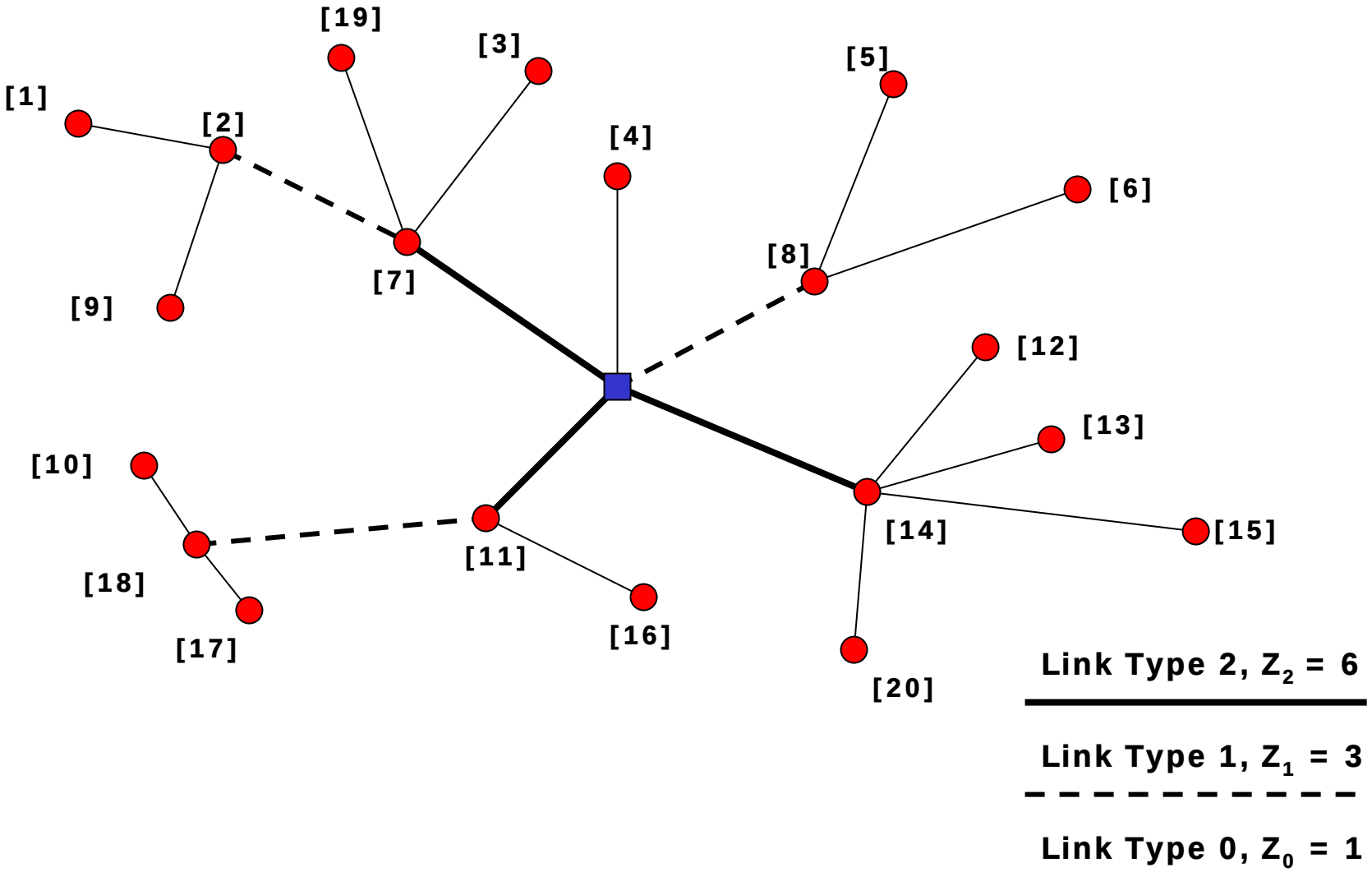
Savings Heuristic



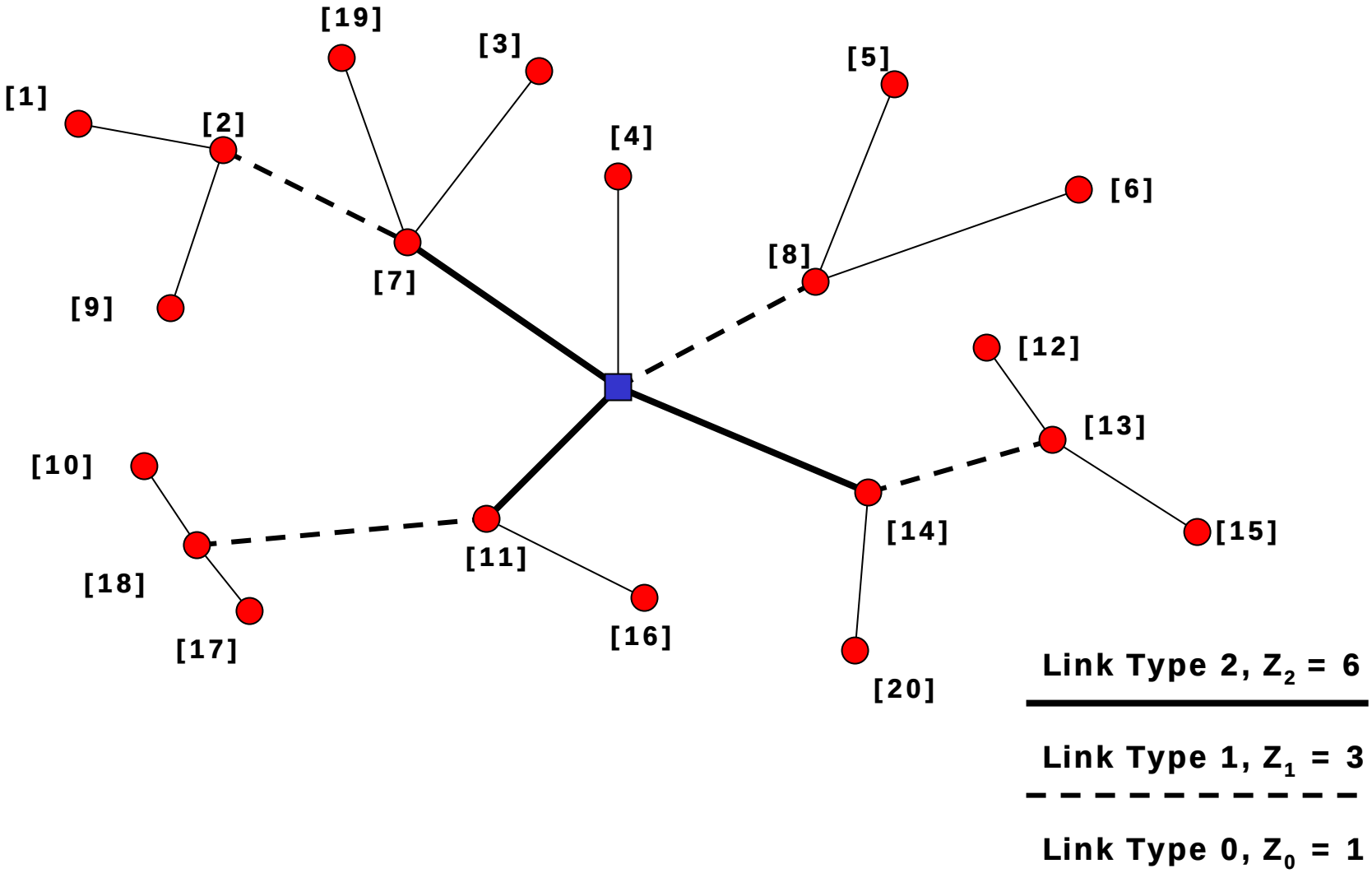
Savings Heuristic



Savings Heuristic



Savings Heuristic



Savings Heuristic

```
Begin
    Generate initial star solution
    l = L
    while (l > 0) do
        exit = false
        while (exit = false) do
            for every node i not in the final solution
                calculate savings
            if (savings > 0)
                select node i with largest savings
                implement the upgrade and reconnections
                add node i in the final solution
            else
                exit = true
        end while
        l = l – 1
    end while
end
```

Running Time:

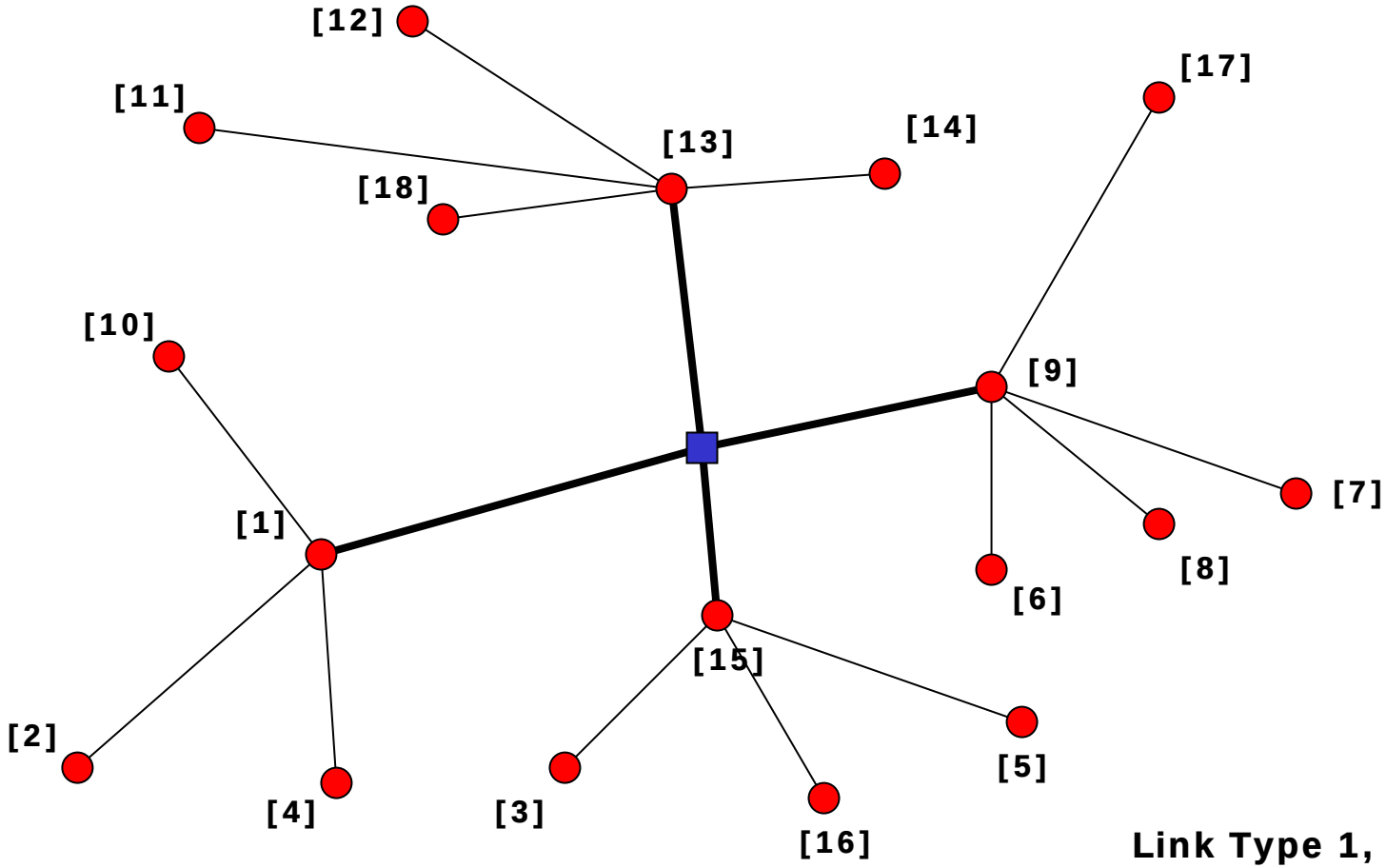
$$O\left(|N|^2\left(\log|N| + Z_L\right)\left(|N| + |\Lambda|\right)\right)$$

Local Search

- Neighborhood Structure defined by Ahuja, Orlin and Sharma in *"Multi-Exchange Neighborhood Structures for the CMST"*.

- The neighborhood is constructed while considering two types of node moves (exchanges): i) Cyclic exchanges, ii) Path exchanges.

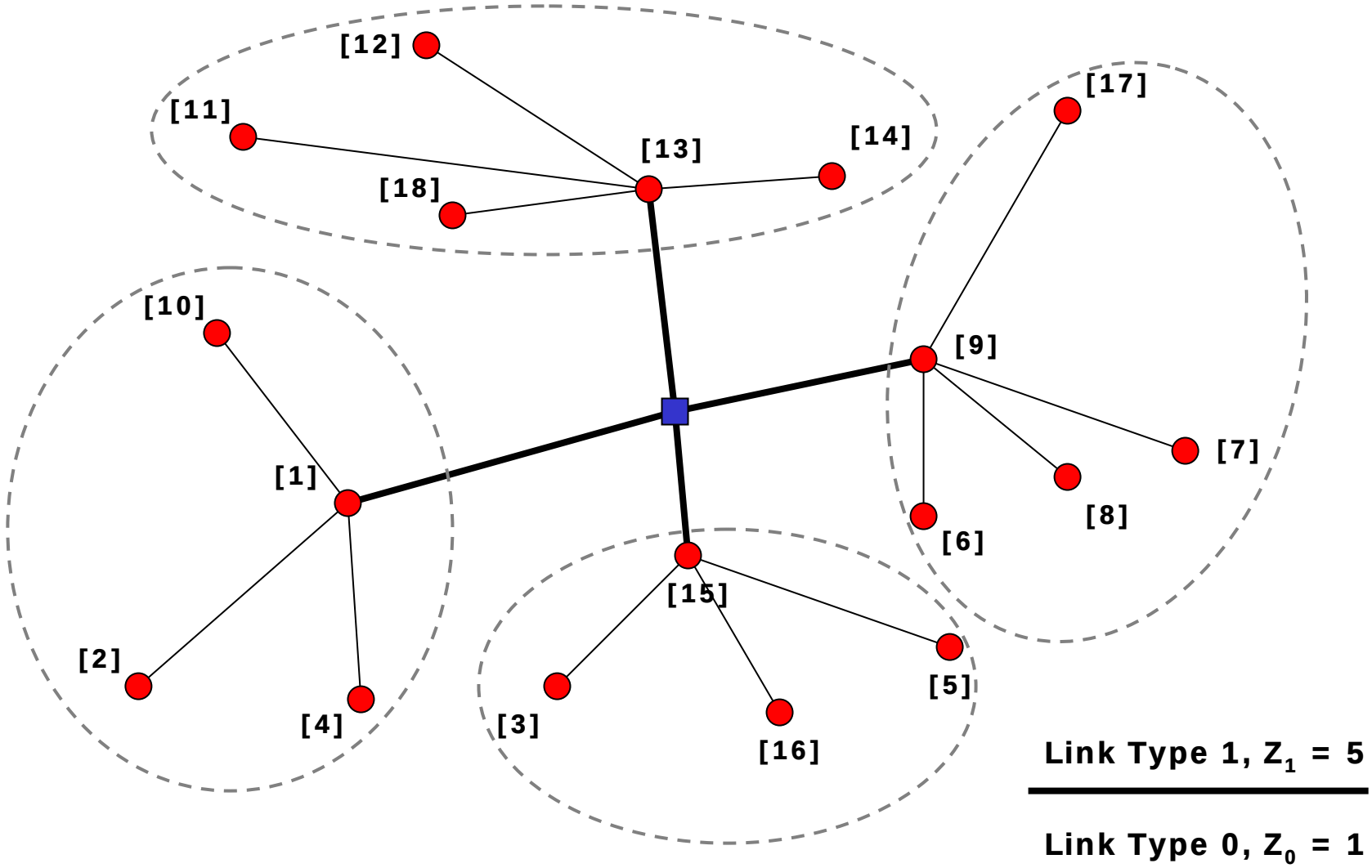
Local Search



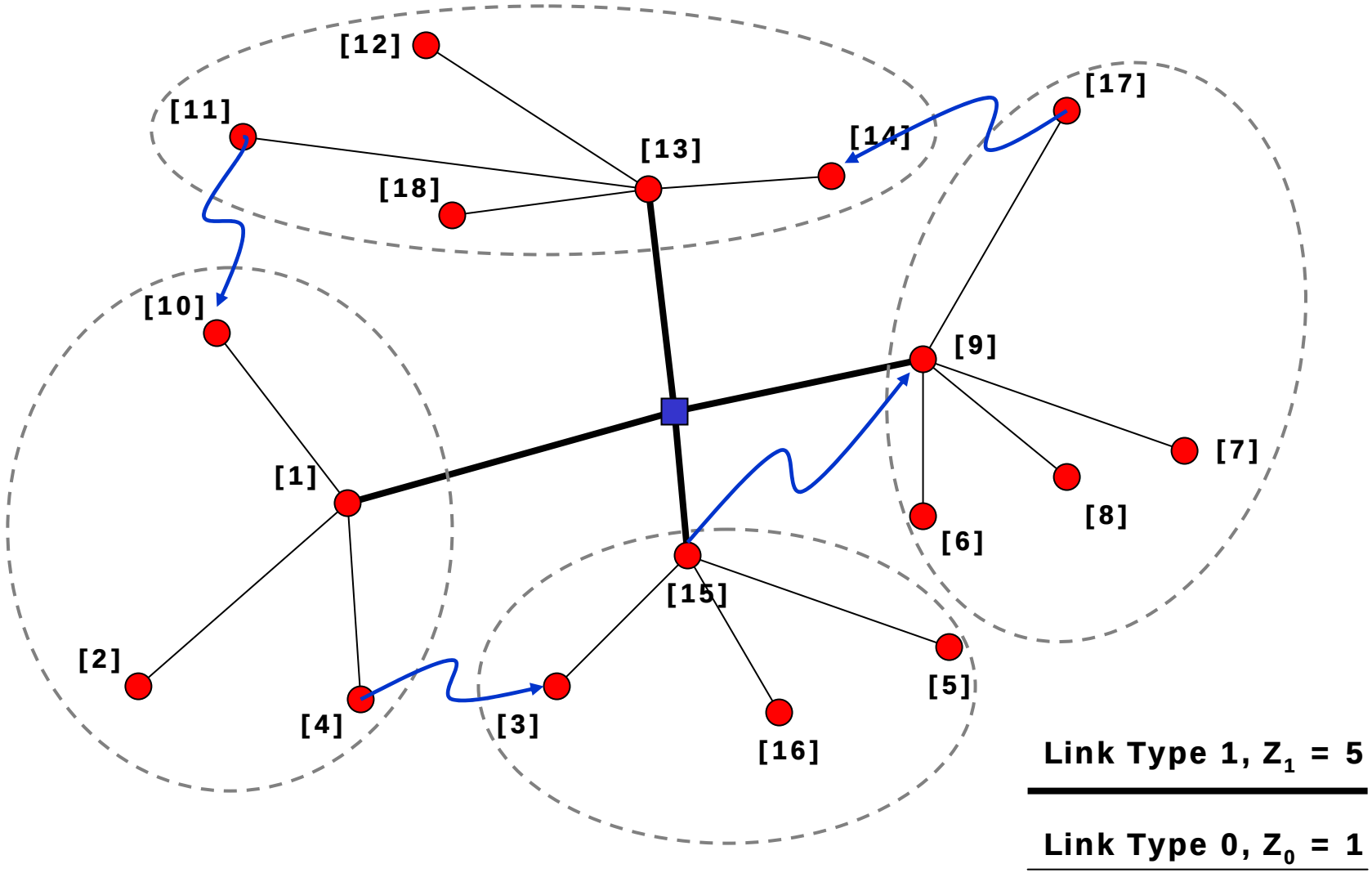
Link Type 1, $Z_1 = 5$

Link Type 0, $Z_0 = 1$

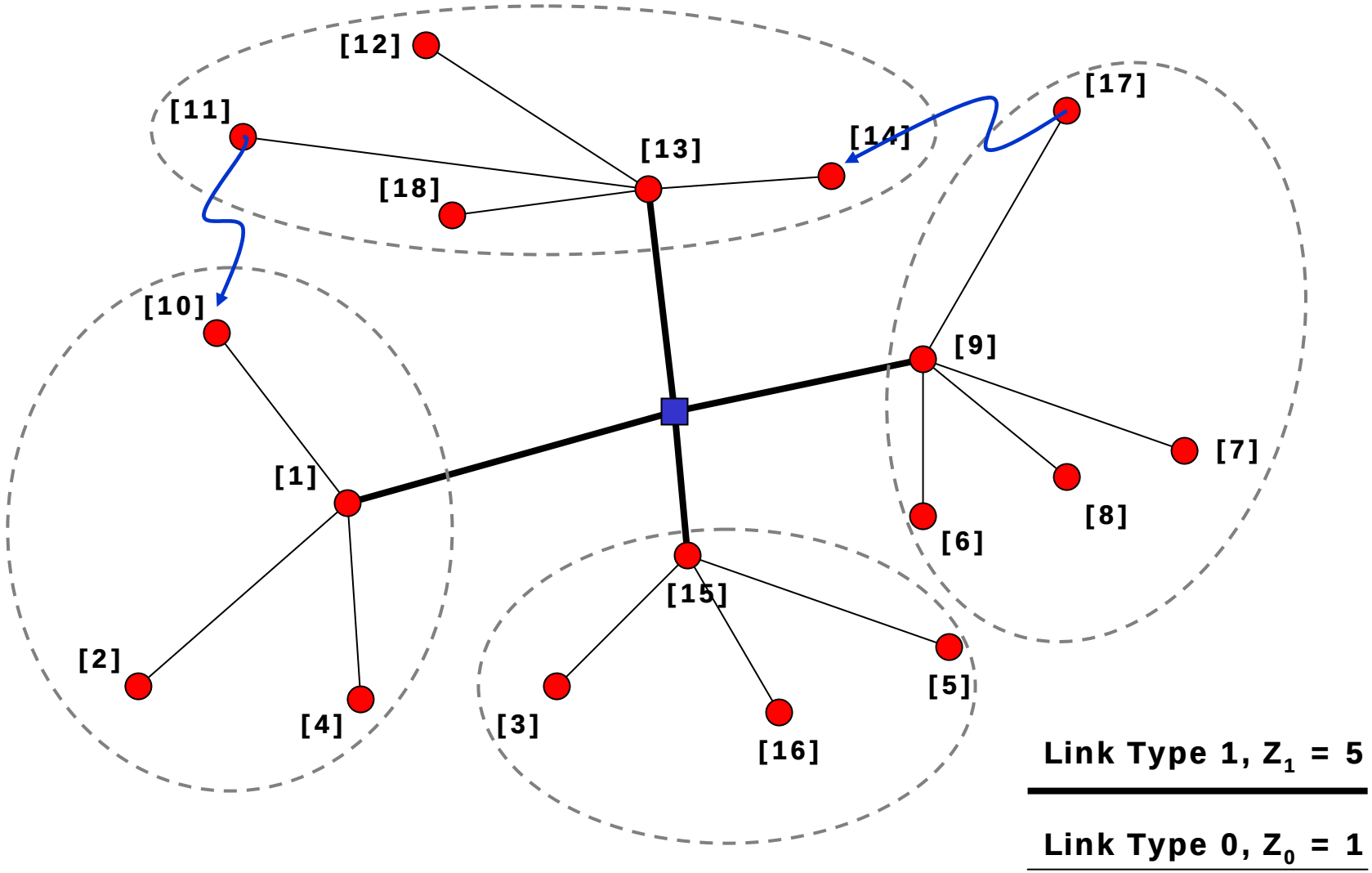
Local Search



Local Search – Cyclic Exchanges



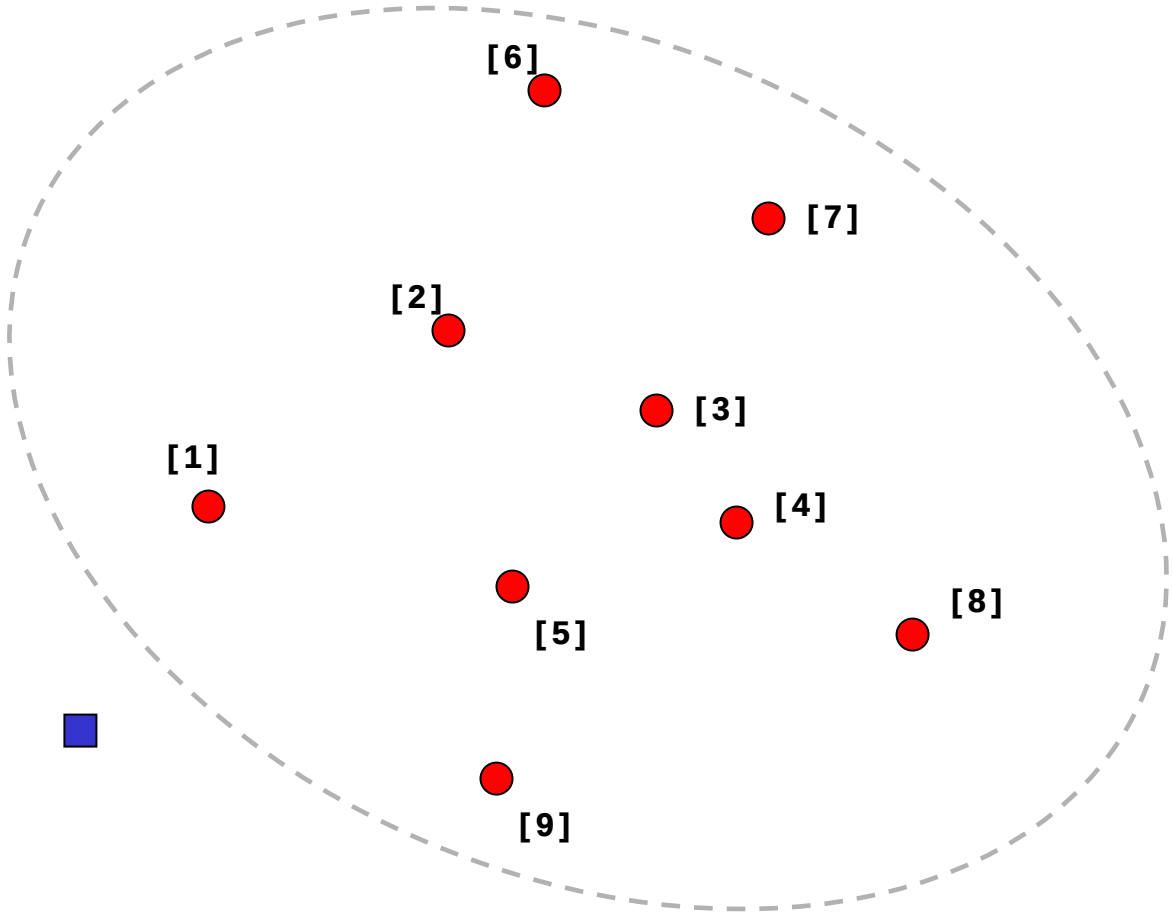
Local Search – Path Exchanges



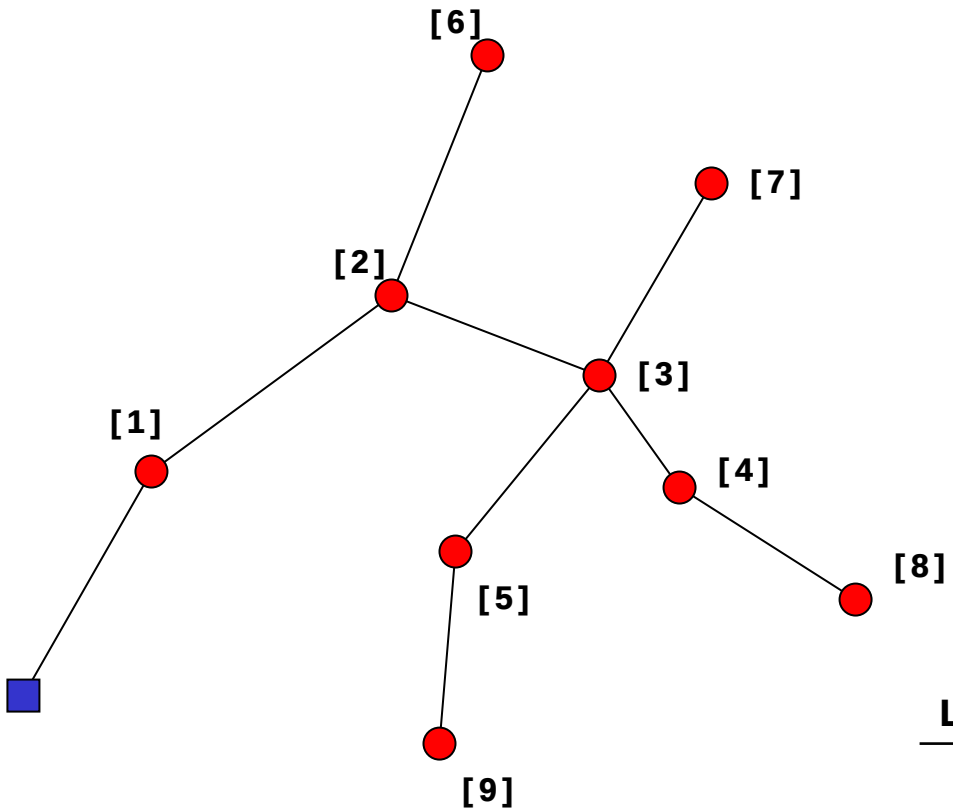
Local Search

- The main problem with the implementation of this procedure for the MLCMST problems is the interconnection of nodes that belong to the same subtree.
- In the CMST problem this is done easily with a Minimum Spanning Tree algorithm. In the MLCMST case the problem is still a multi-level problem.

Local Search



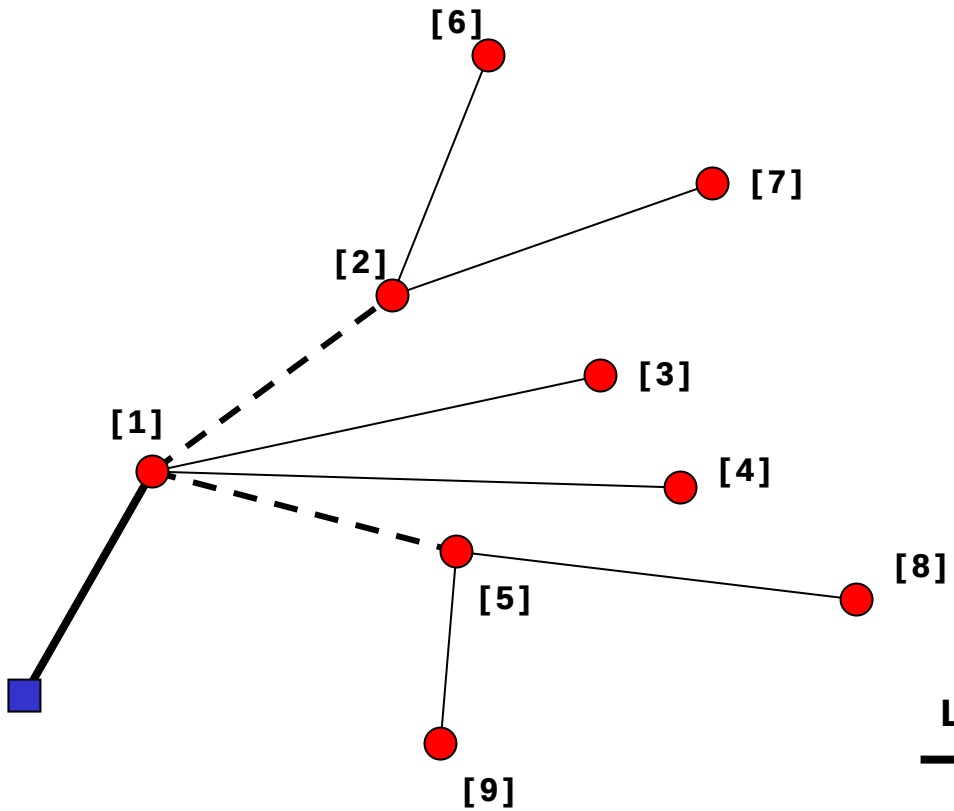
Local Search



CMST

Link Type 0, Z_0 = 9

Local Search



MLCMST

Link Type 2, $Z_2 = 9$

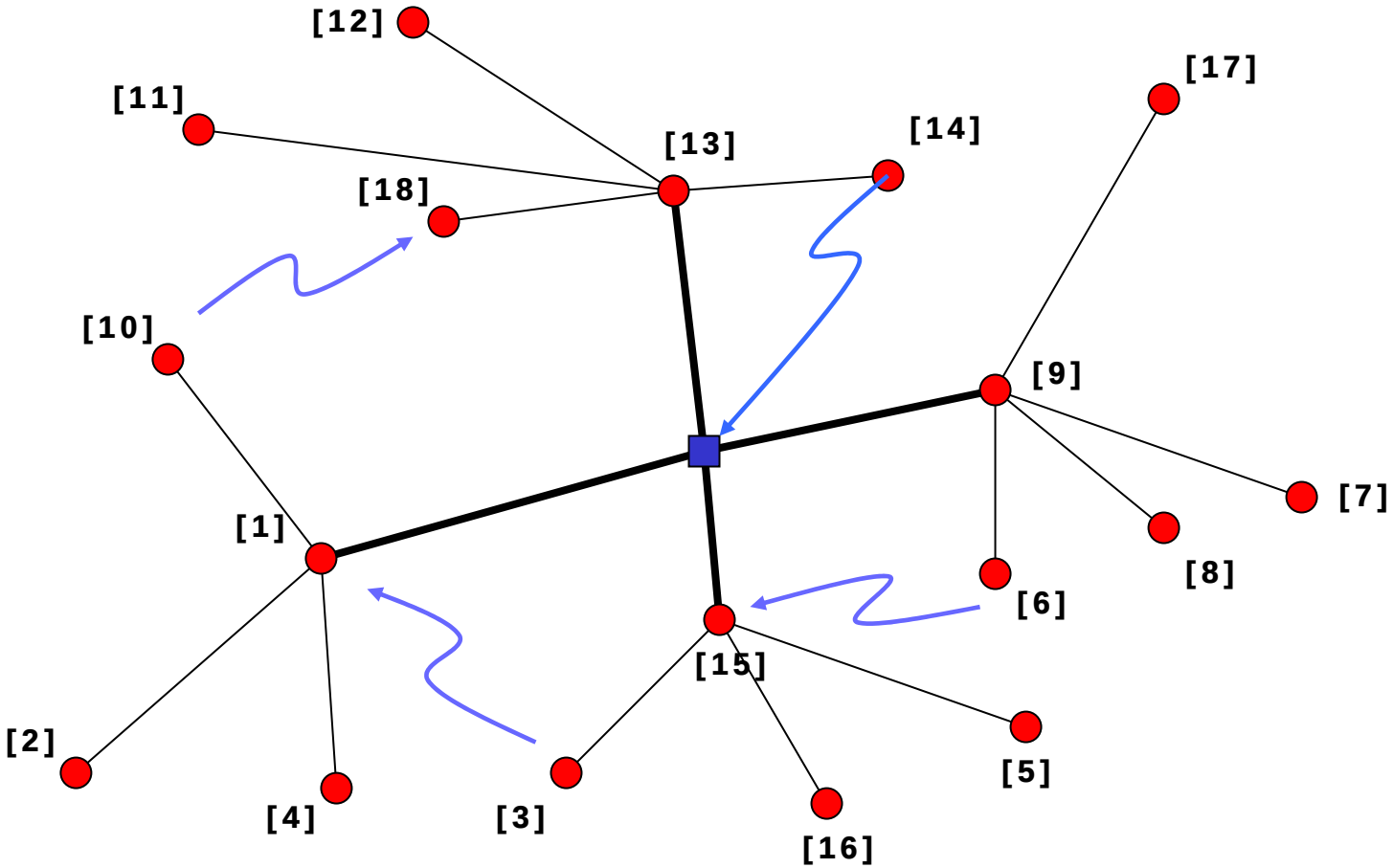
Link Type 1, $Z_1 = 3$

Link Type 0, $Z_0 = 1$

Local Search

- We use the MLCMST heuristic for the interconnection of nodes in the same subtreee.
  - Interconnection is subotpimal (solution quality).
  - Adding or deleting a node from a subtree requires that we solve the interconnection problem from scratch (running time).

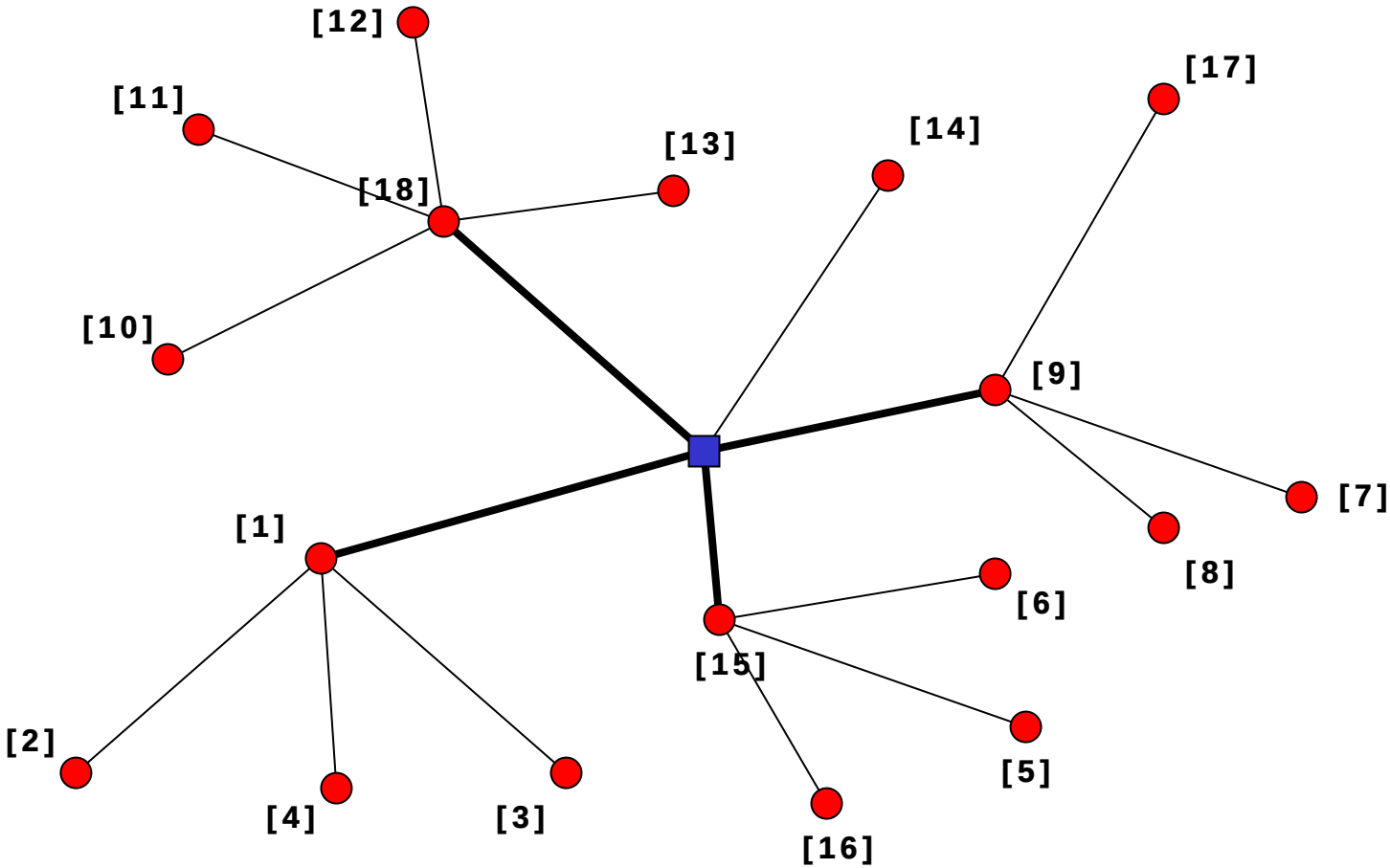
Local Search



Link Type 1, $Z_1 = 5$

Link Type 0, $Z_0 = 1$

Local Search



Link Type 1, $Z_1 = 5$

Link Type 0, $Z_0 = 1$

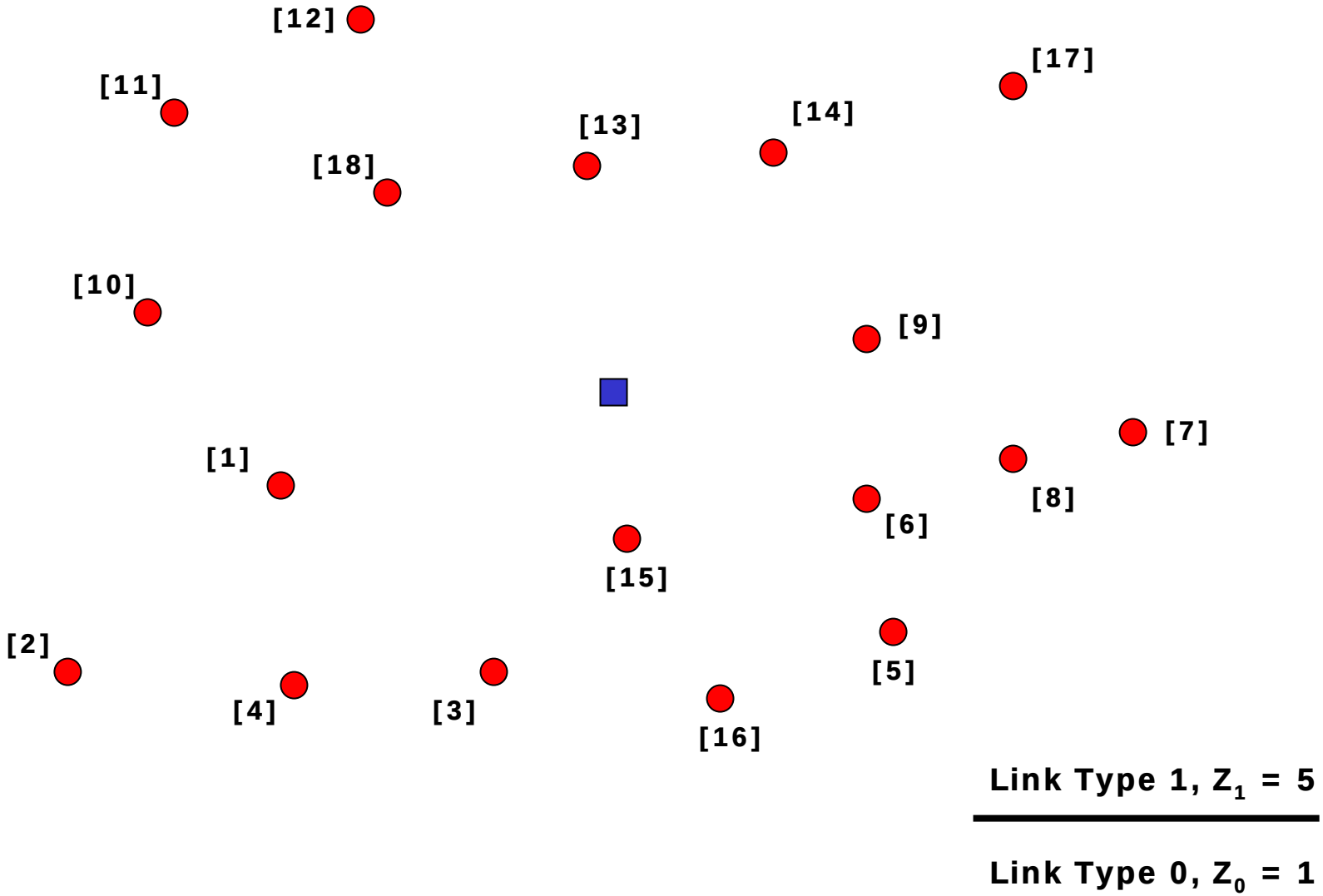
Local Search

- Our procedure starts from a feasible solution provided by the Savings heuristic.
- It looks for improvements to the initial solution by applying searching through the neighborhood.
  - Size of neighborhood exponential.
  - Search requires identification of negative cost cycles on a “neighborhood graph”.
- It stops when no more improvements can be found.

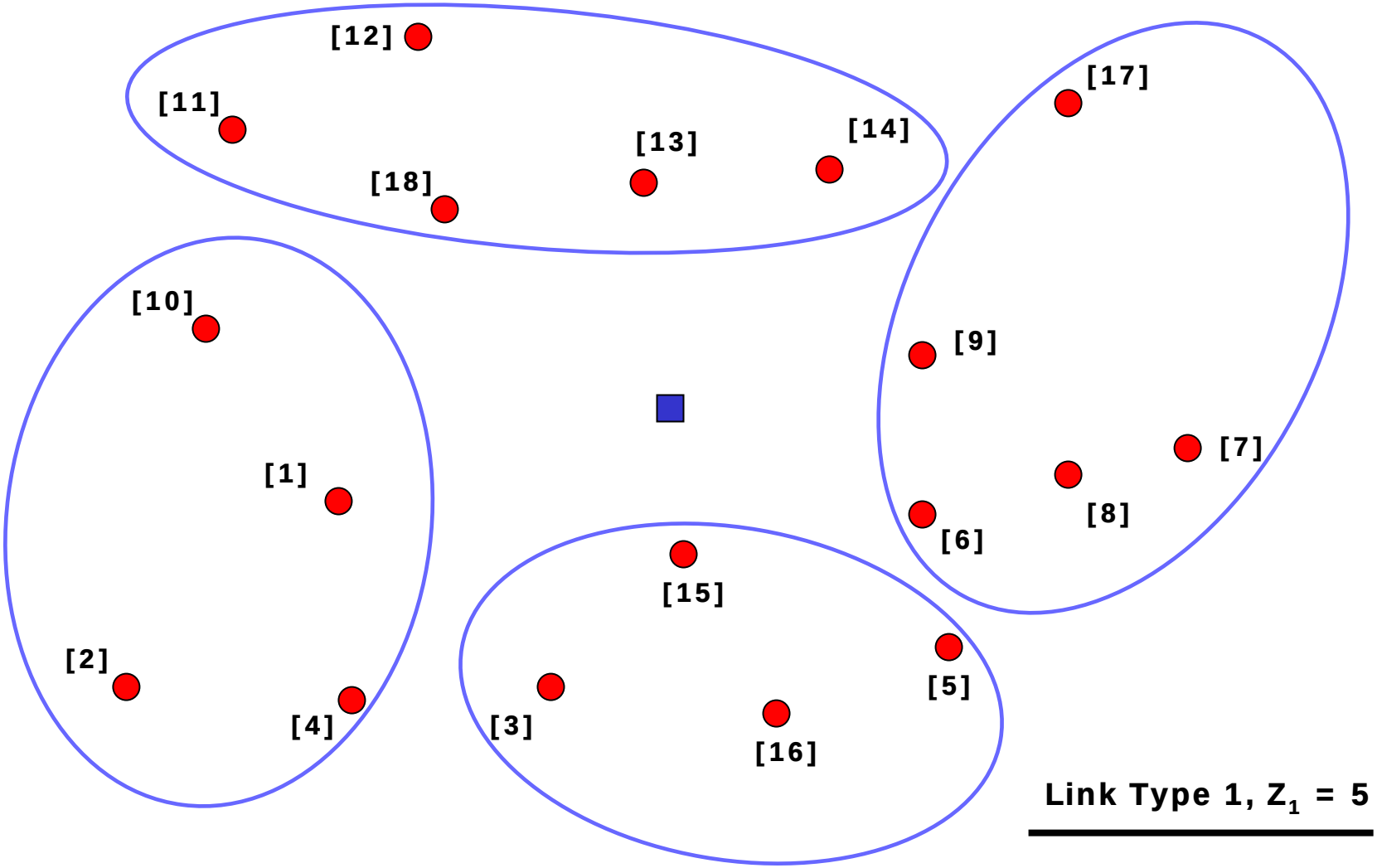
Genetic Algorithm

- Determine groups of nodes that are in the same subtree.
- Find the subtrees by running the MLCMST construction heuristic.
- Representation
  - Based on Falkenauer's approach for grouping problems
- Crossover Operator
- Mutation Operator
  - Based on the local search procedure

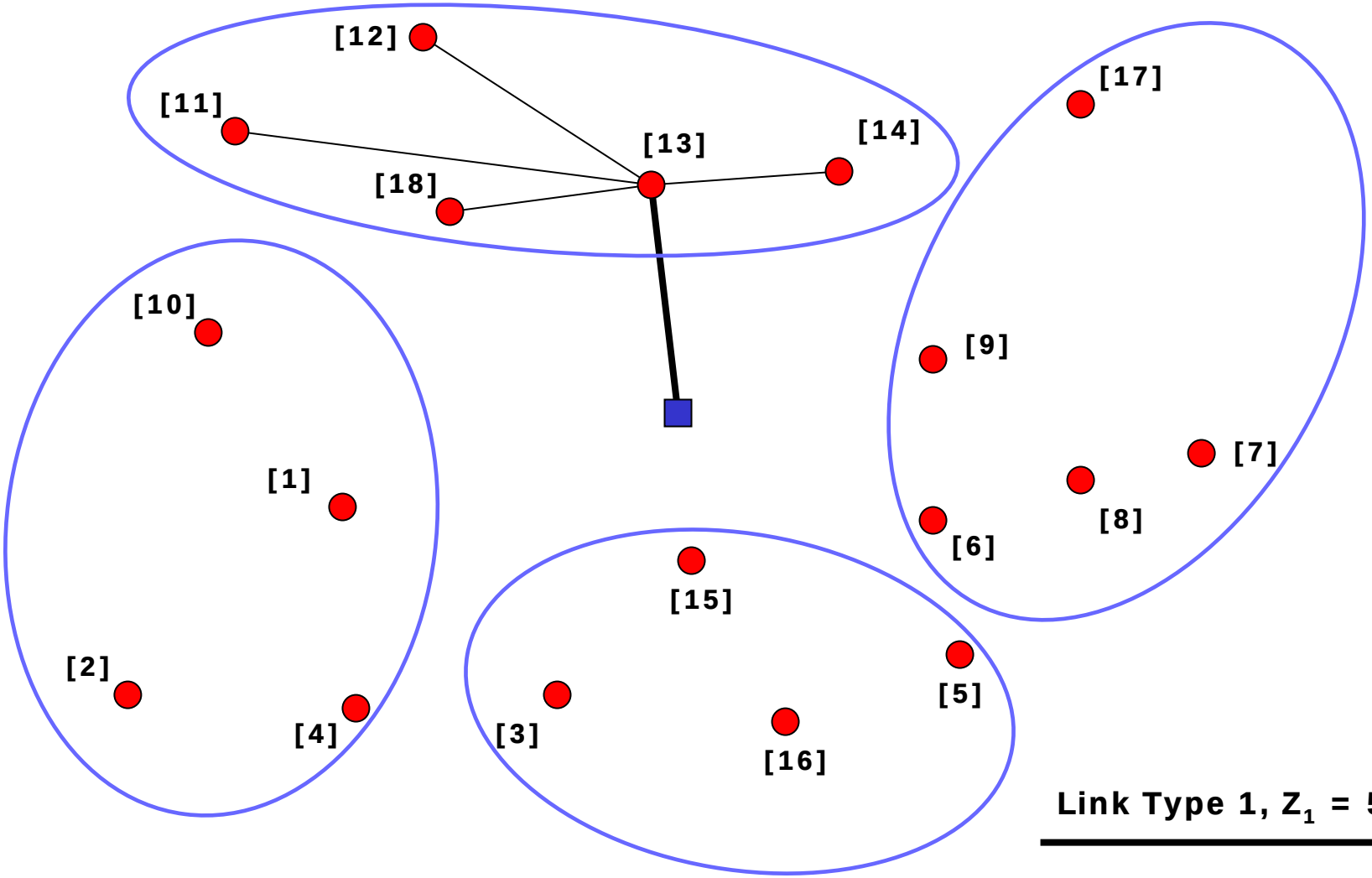
Genetic Algorithm



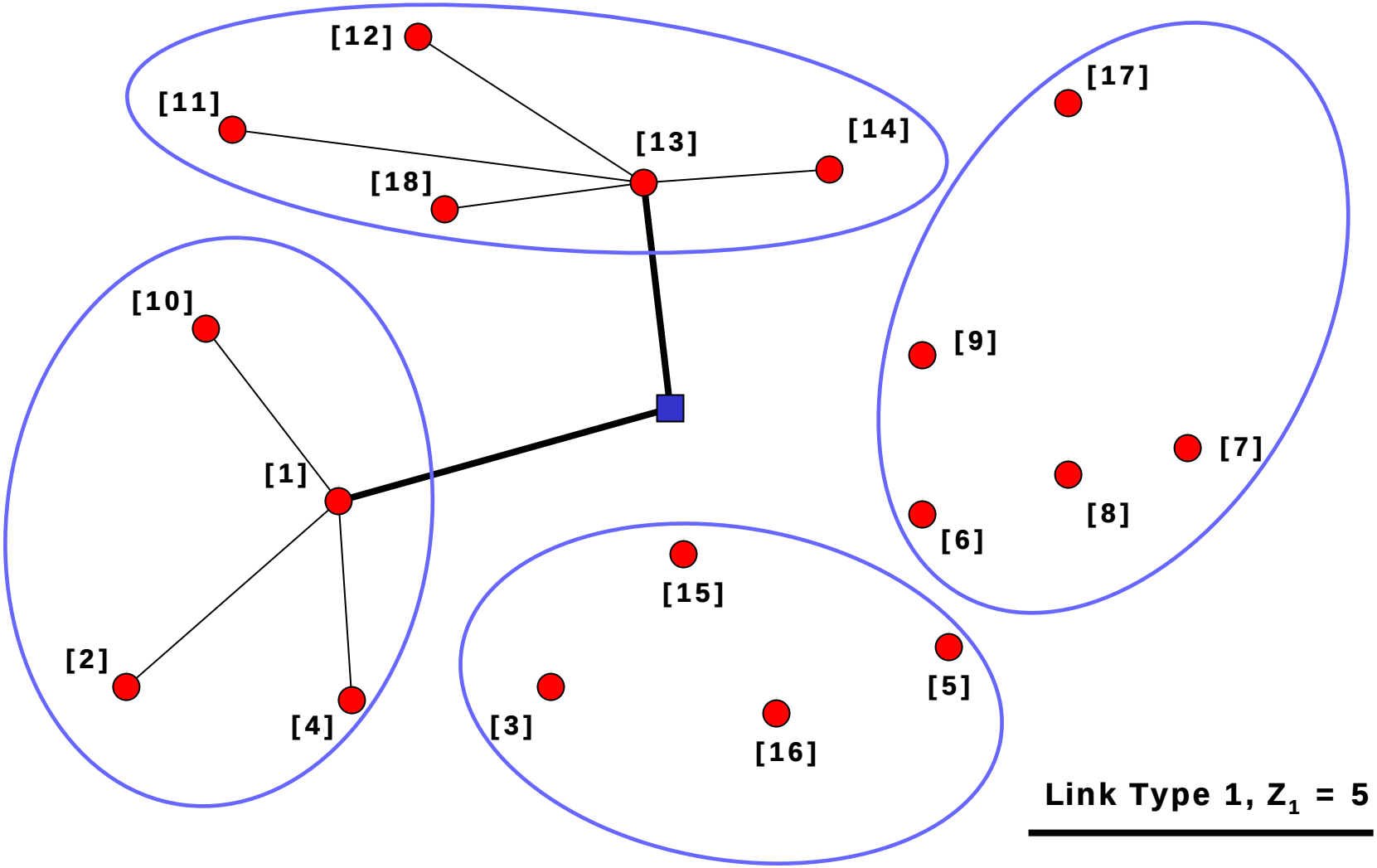
Genetic Algorithm



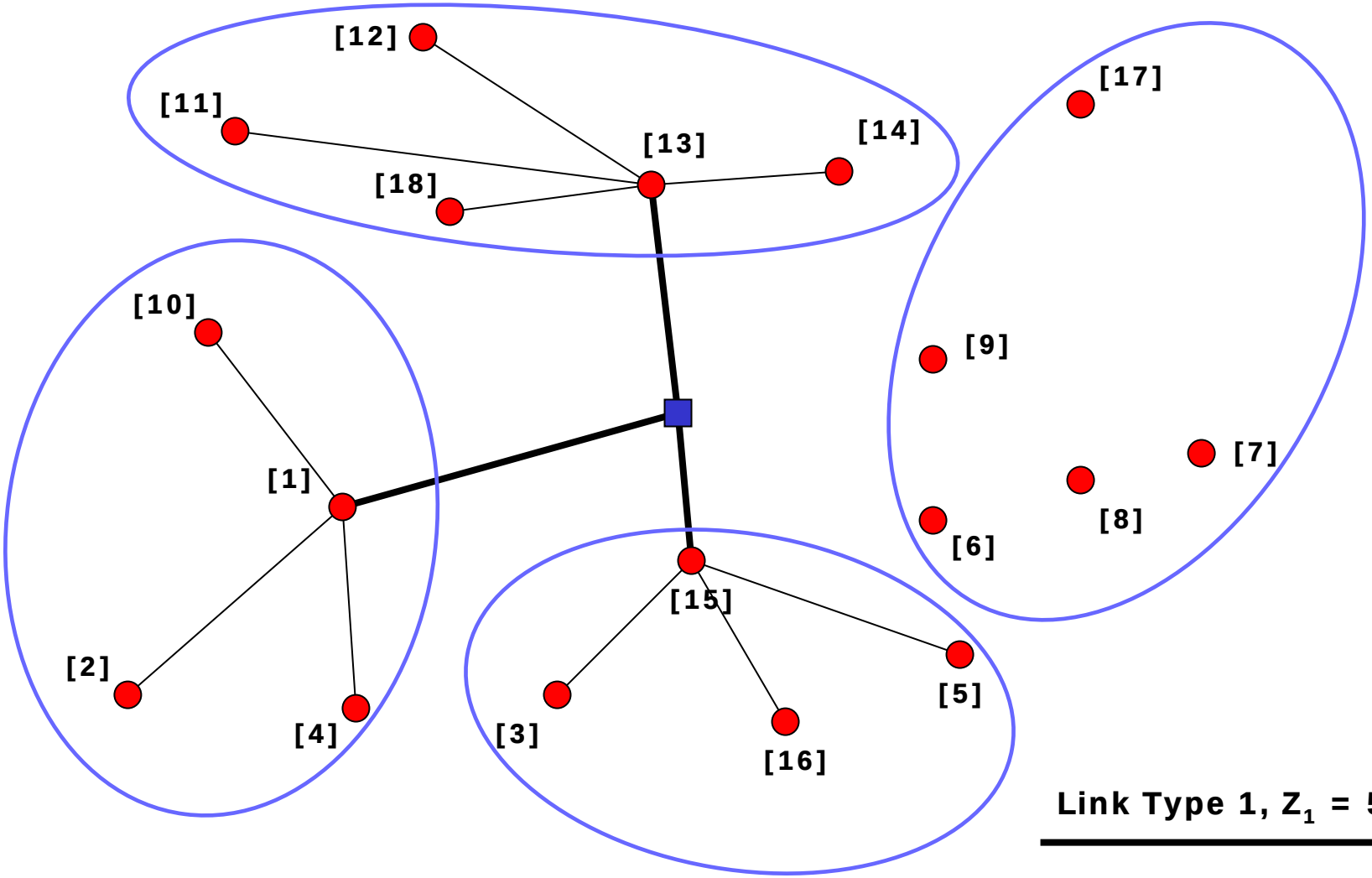
Genetic Algorithm



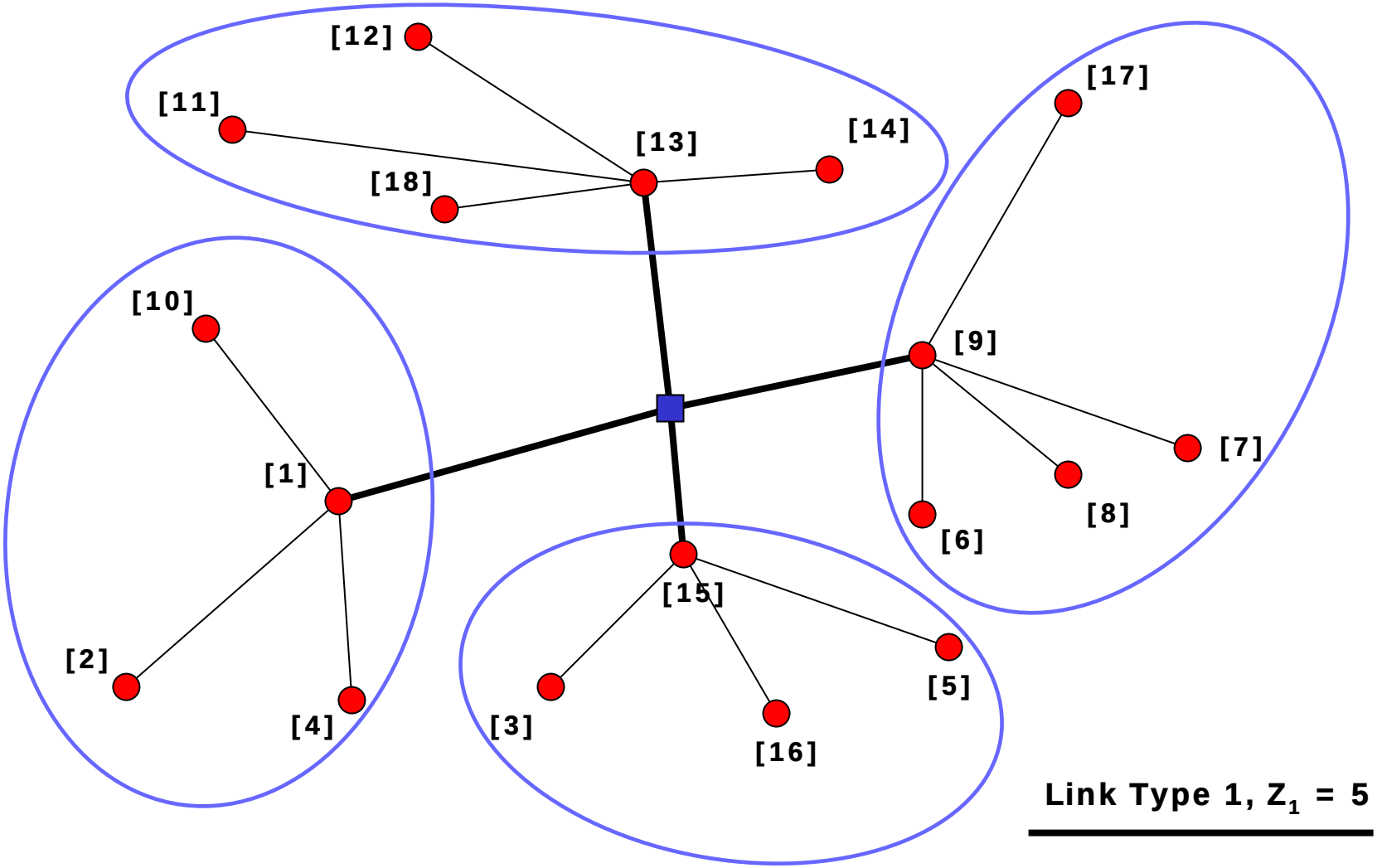
Genetic Algorithm



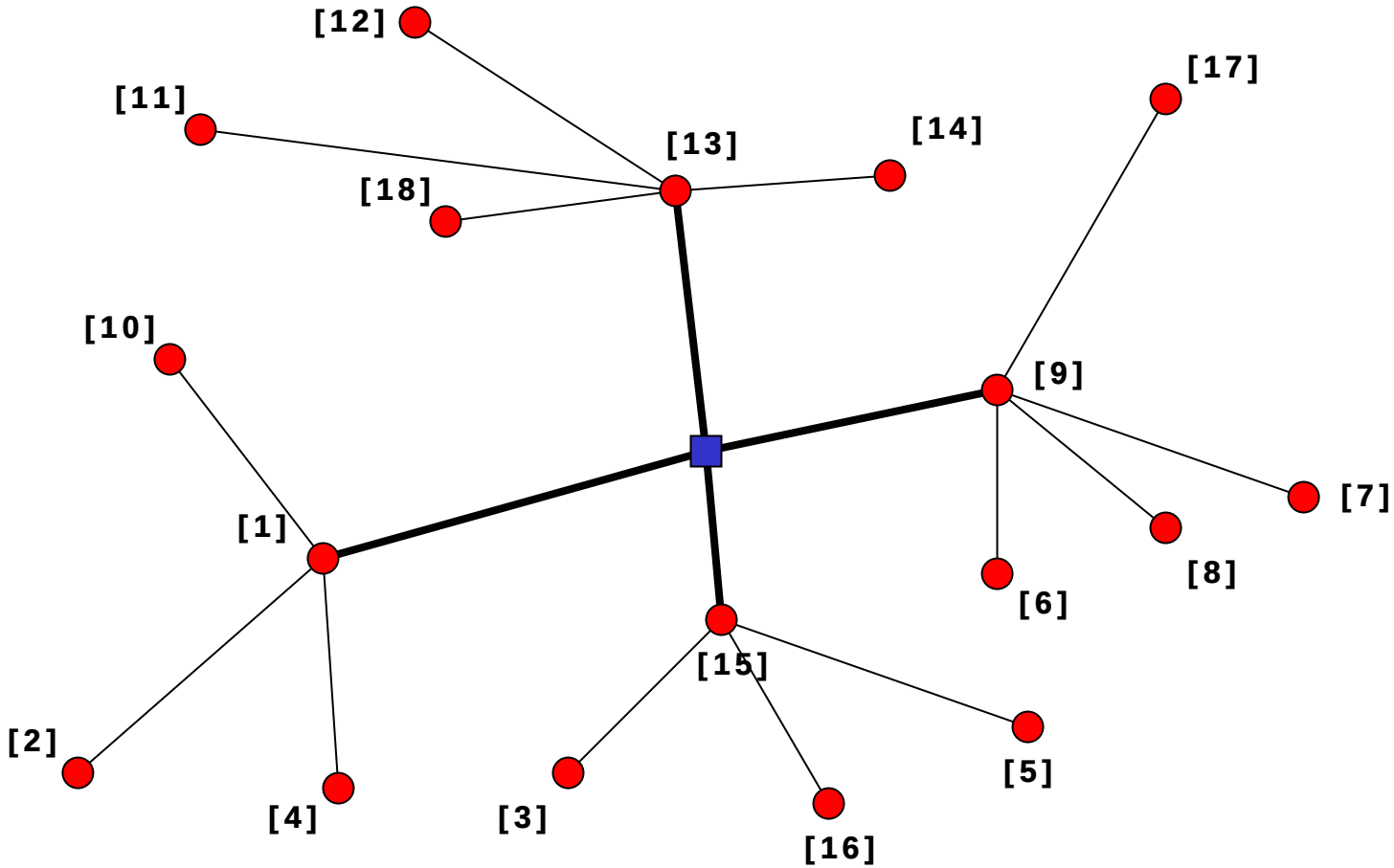
Genetic Algorithm



Genetic Algorithm



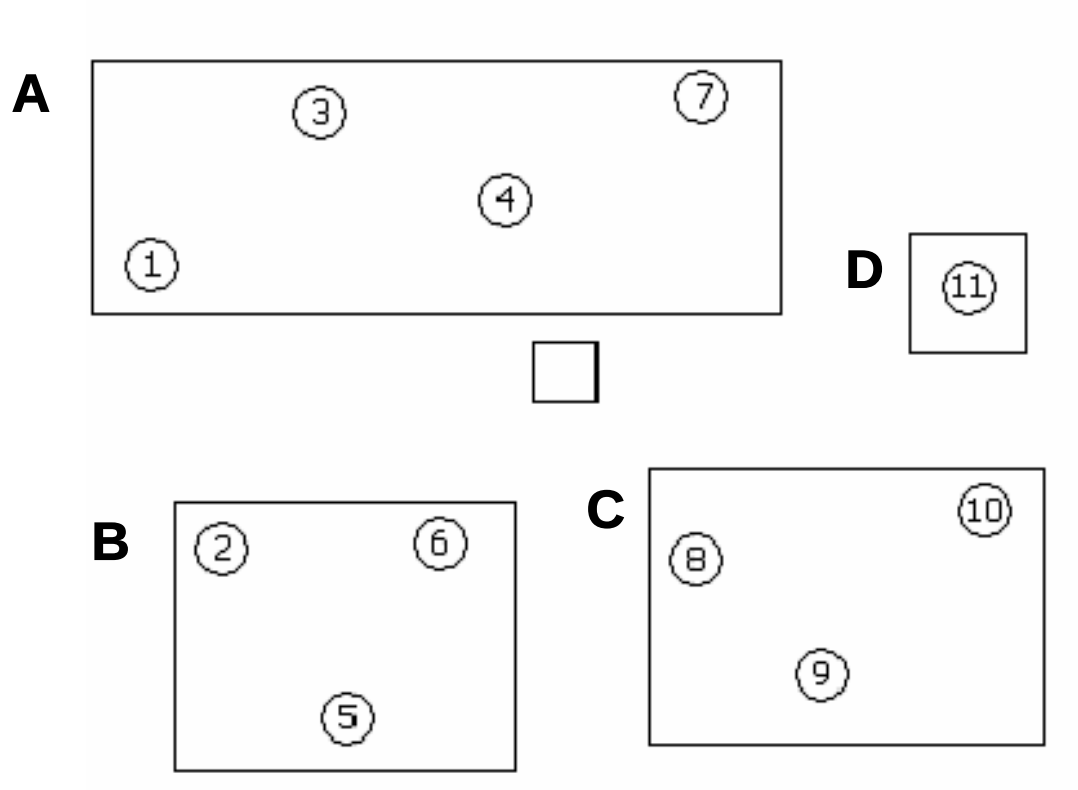
Genetic Algorithm



Link Type 1, $Z_1 = 5$

Link Type 0, $Z_0 = 1$

Genetic Algorithm - Representation



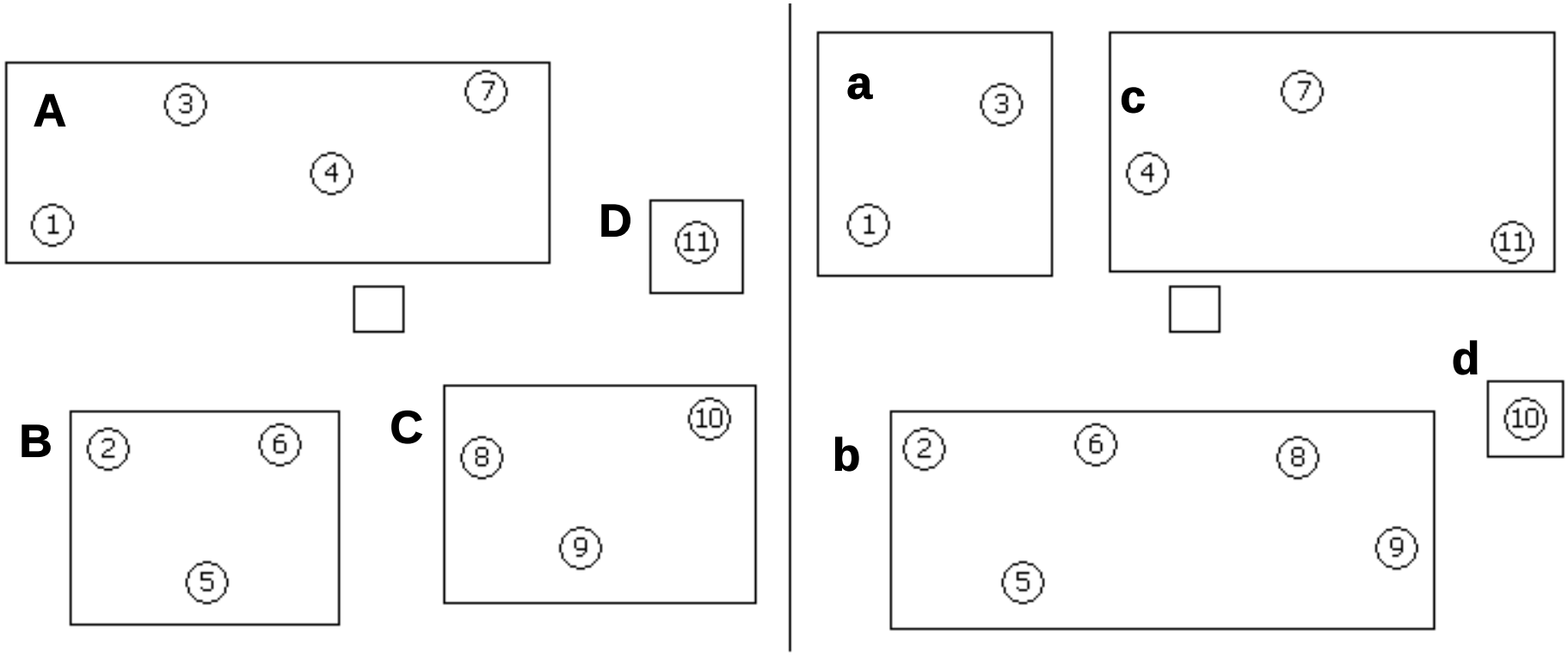
- Representation of group assignments:

A B A A B B A C C C D : A B C D

Item Part

Group Part

Genetic Algorithm - Crossover



Left: **A B A A B B A C C C D : A B C D**

Right: **a b a c b b c b b d c : a b c d**

Multi-Level Capacitated Minimum Spanning Tree

Genetic Algorithm - Crossover

- Crossover (Crossover acts on the group part only):
  - Parent chromosomes:

A B A A B B A C C C D : A B C D

a b a c b b c b b d c : a b c d

  - We select at random the crossing sites:

A B A A B B A C C C D : A | B | C D

a b a c b b c b b d c : | a b c | d

Genetic Algorithm - Crossover

- Injection of the contents of the crossing section of the first parent to the first section of the second parent:

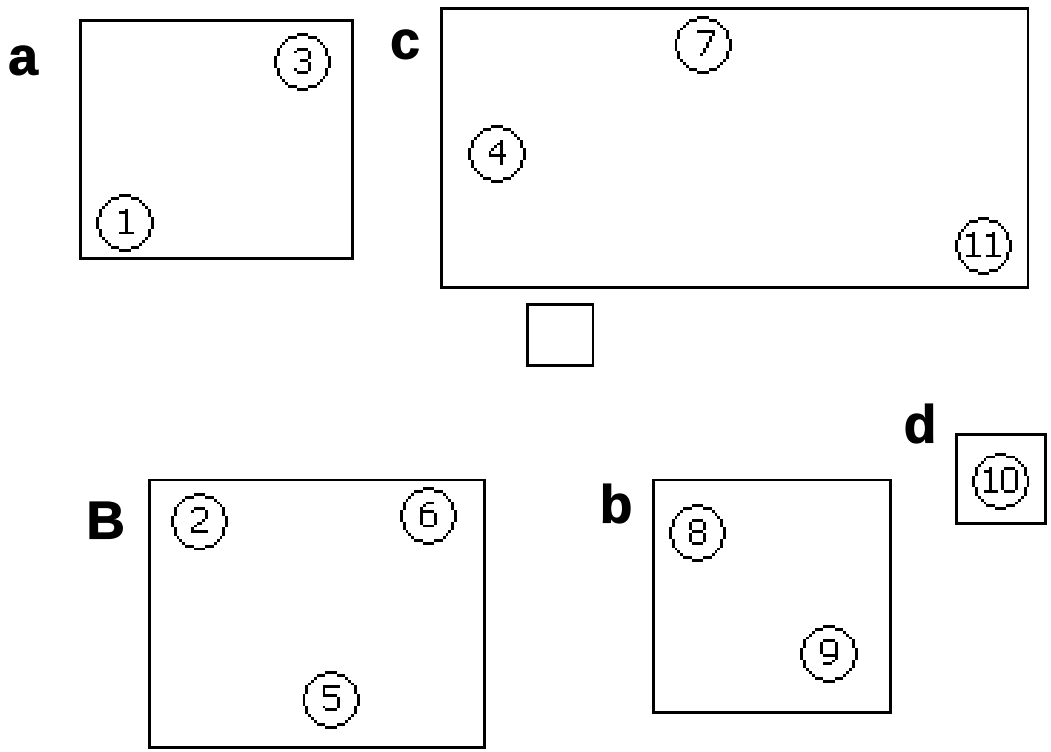
Parents:

A B A A B B A C C C D : A | B | C D

a b a c b b c b b d c : | a b c | d

Offspring: **a B a c B B c b b d c : B a b c d**

Genetic Algorithm - Crossover Result



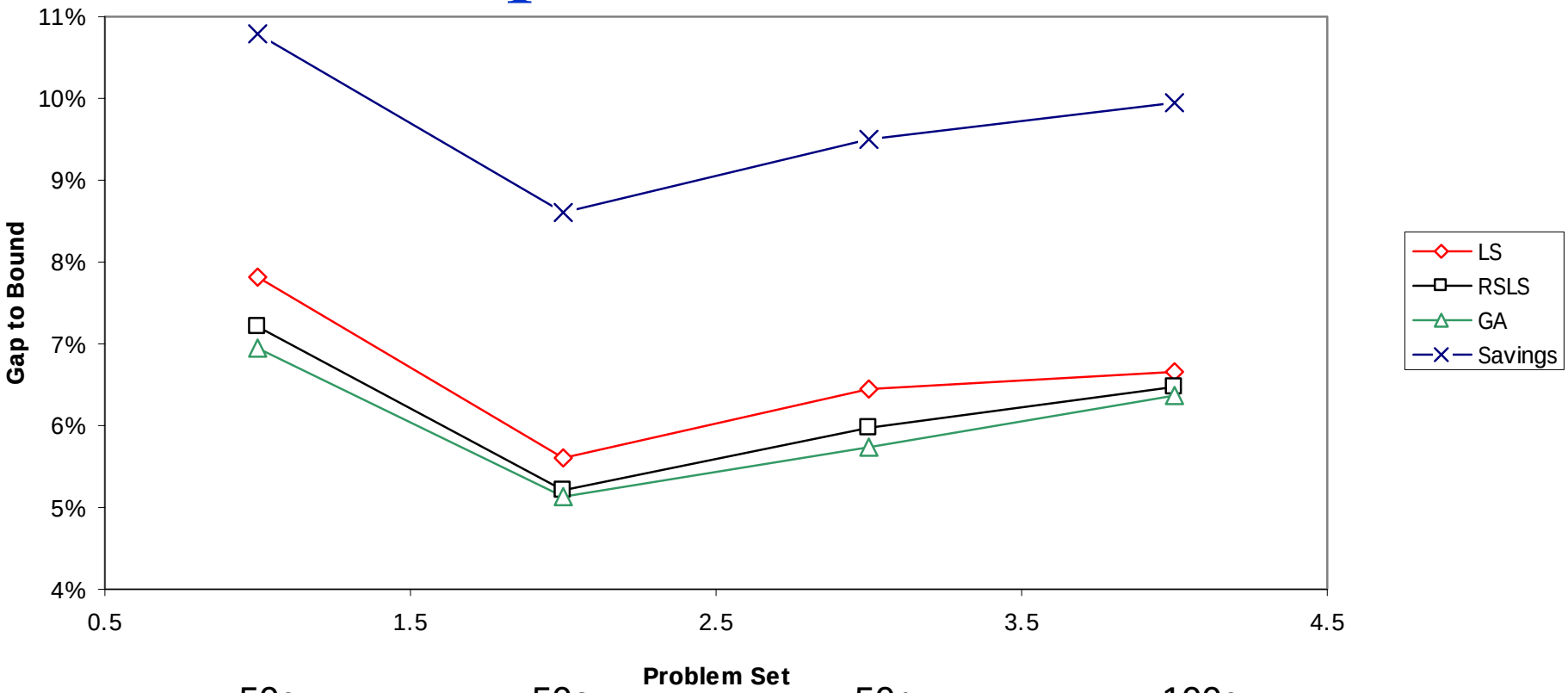
a B a c B B c b b d c : B a b c d

- Representation and Crossover: Falkenauer (96)

Computational Results

- Genetic Algorithm, Local Search and Construction Heuristic coded in Visual C++
- MIP Formulation coded in ILOG OPL Studio programming language
- All runs were made on a Dual Processor Pentium III @ 1GHz with 512MB RAM
- Three 50-node problem sets with the position of the central node at different places
- One 100-node problem set with central node in the center
- All problem sets contained 50 problems

Computational Results



HEU: < 1sec LS : 20sec RSLS: 436sec GA: 255sec

HEU: < 1sec LS : 237sec RSLS: 3,643sec GA: 2,296sec

The Prize-Collecting Generalized Minimum Spanning Tree Problem

B. Golden

S. Raghavan

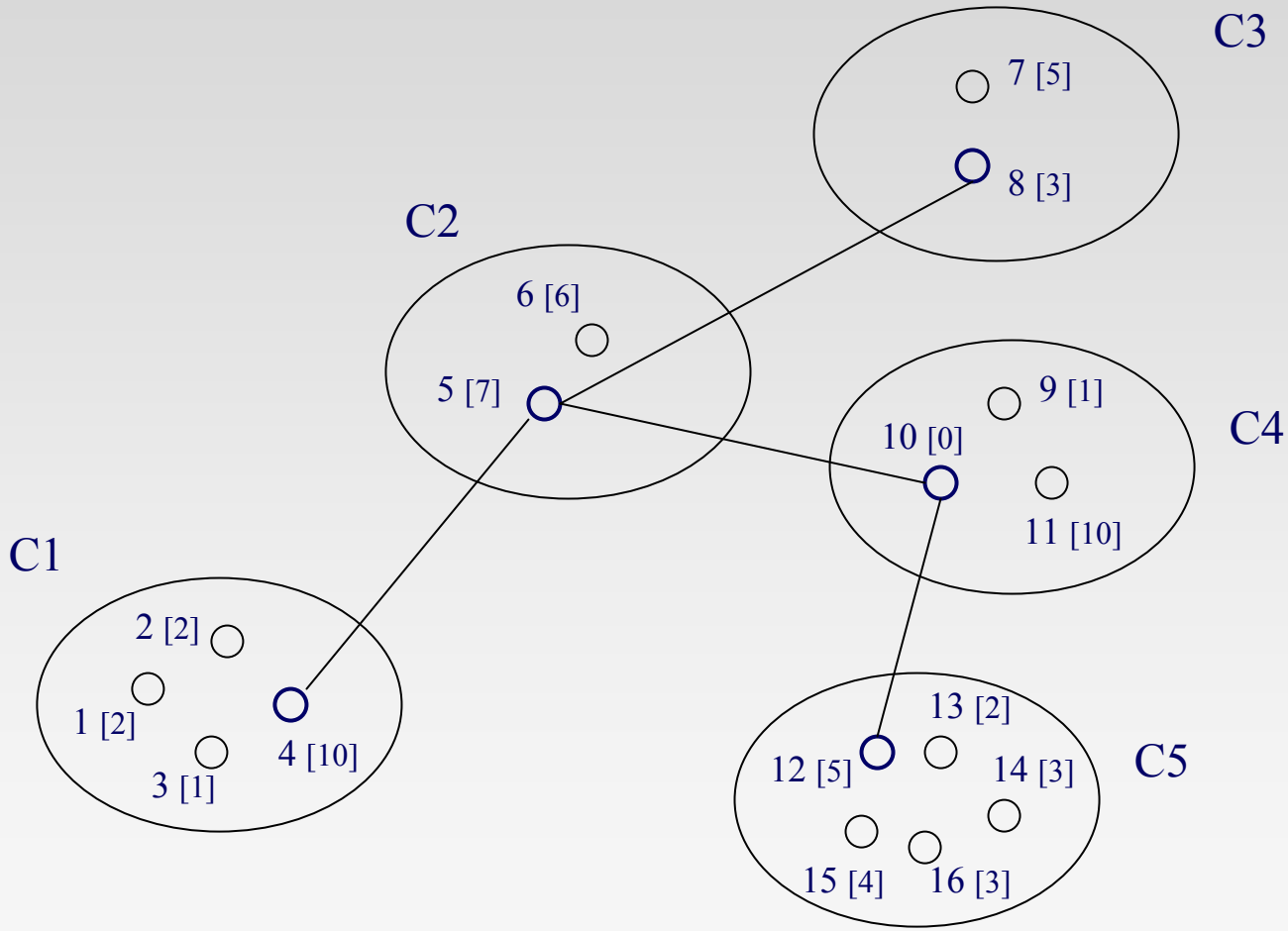
D. Stanojević

Robert H. Smith School of Business, University of Maryland, College Park

Outline

- Problem Definition
- Background
- Heuristics for the PC-GMST problem
  - Local Search
  - Genetic Algorithm
- Computational Results
- Concluding Remarks

Problem Definition



Problem Definition

Given:

- An undirected graph G = (V, E), with node set V and edge set E
- Cost vector c on E
- Prize vector p on V
- Set of mutually exclusive and exhaustive node sets

Find a minimum cost tree than spans exactly one node from each cluster.

Note: Cost function is here defined as a difference between total cost of selected edges and total prizes from selected nodes.

Background

- Motivated by regional connection of local area networks
- Generalization of the *NP* – hard GMST problem (introduced by Myung et al. 1995)
- Existing exact solution procedures for the GMST problem:
  - Dual Ascent (Myung et al. 1995)
  - Branch and Cut (Feremans 2001)
  - Rooting procedure based on the polynomial size MIP relaxation (Pop 2002)

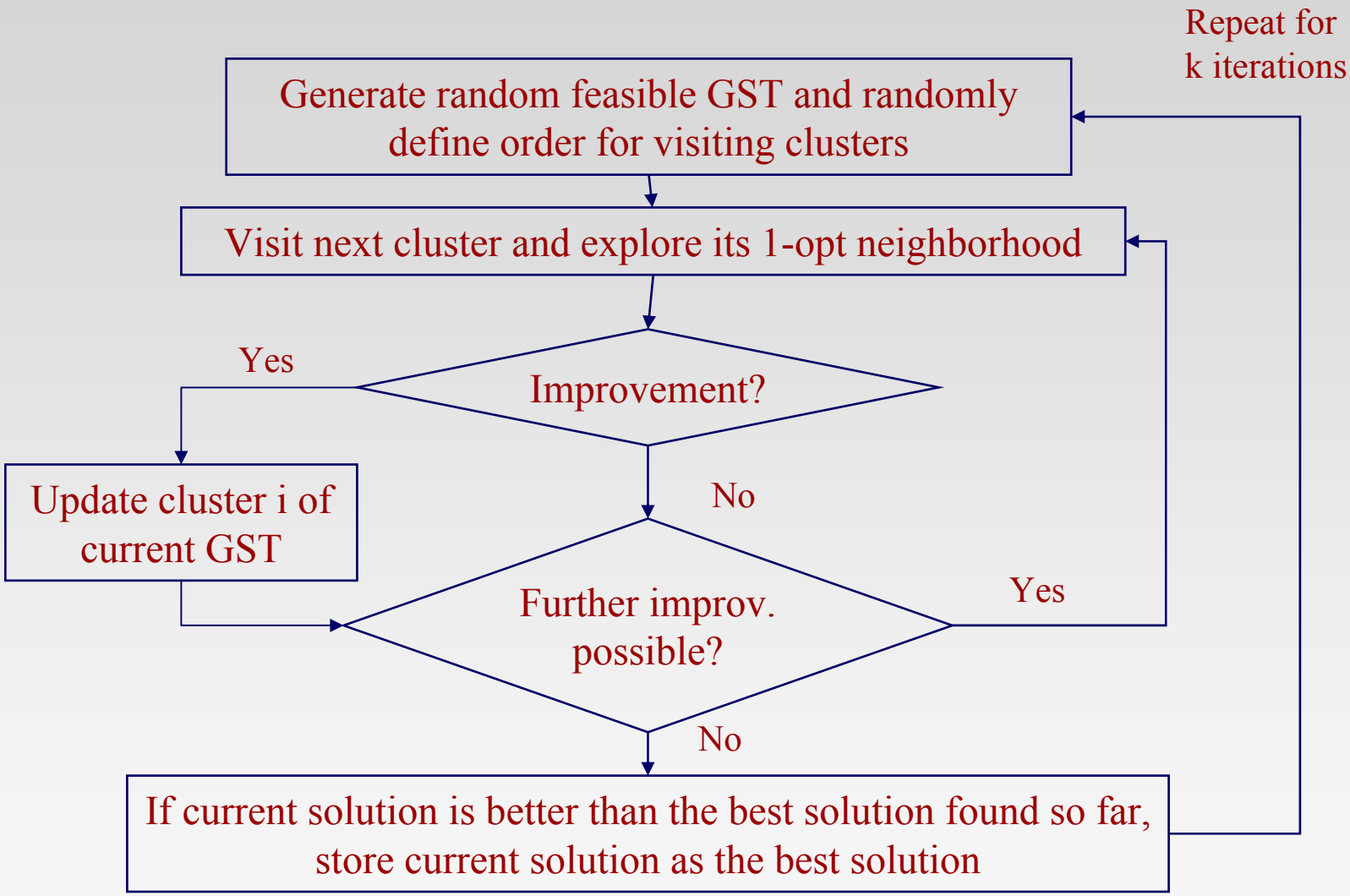
Background (cont.)

- Straightforward adaptations of MST procedures do not provide good upper bounds
- Other existing procedures have computational limits (as applied to GMST problem):
  - Largest problems considered
    - Euclidean: complete graphs with 200 nodes
    - Non-Euclidean: complete graphs with 200 nodes
  - Time needed to solve large instances (200 – 230 nodes)
    - Euclidean: > 2 hours
    - Non-Euclidean: up to 30 minutes
- This gives a motivation for design of fast and high quality heuristics

Heuristics for the PC-GMST problem

- Simple Local Search Algorithm based on the 1-opt exploration of the search space
- Genetic Algorithm

Local Search



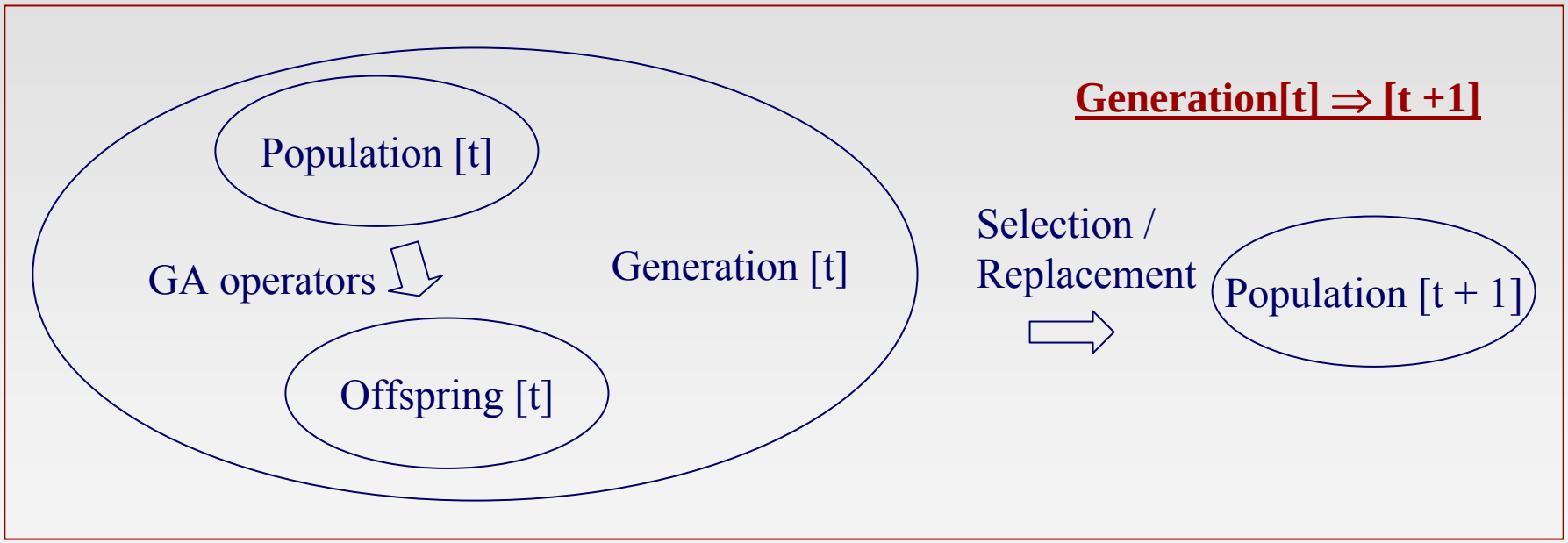
Genetic Algorithm

- GA Mechanism
- Chromosome Representation
- Initial Population
- Genetic Operators
  - Local Search Enhanced (LSE) Crossover
  - Random Mutation
- Selection/Replacement
- Termination Condition

GA Mechanism

Initial Population

Population[1]: Randomly generated feasible GSTs and apply LS to each new solution

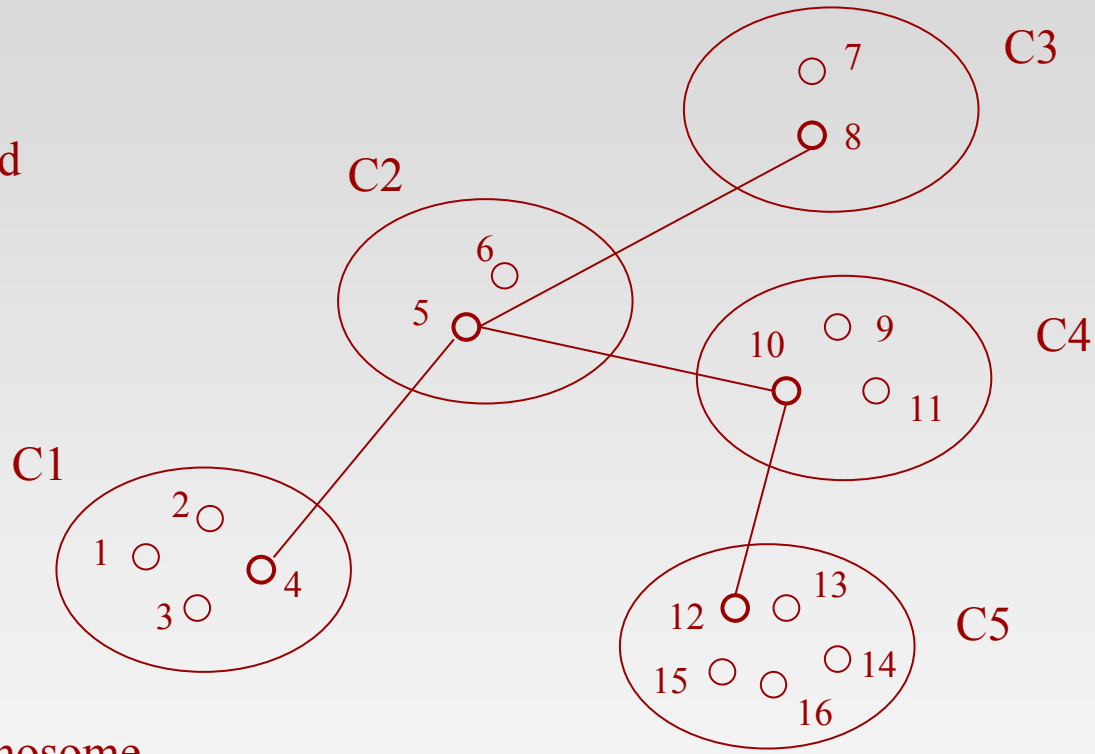


Termination Condition

Pre-specified number of generations

Chromosome Representation

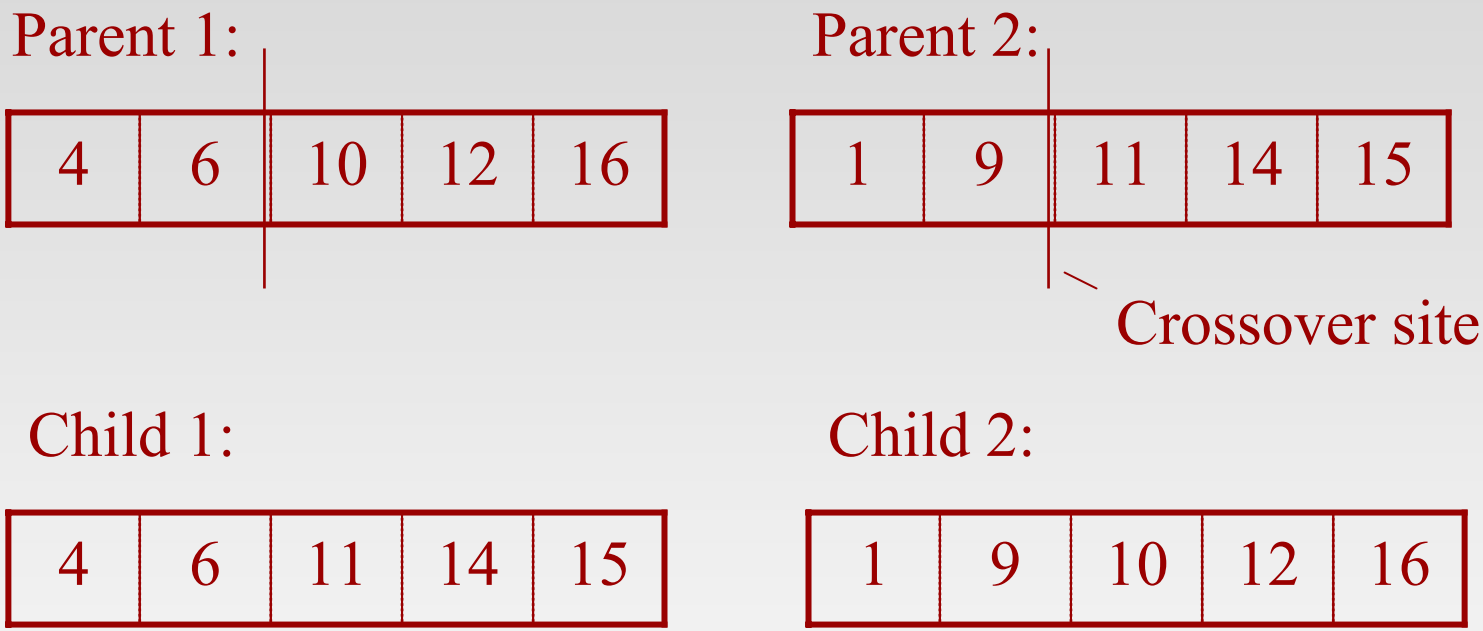
Feasible generalized spanning tree



Corresponding chromosome representation

4	5	8	10	12

Genetic Operators: (LSE) Crossover



LS phase: Before adding child 1 and child 2 to the population (provided they are feasible), LS is applied to each chromosome.

Genetic Operators: Random Mutation

- Random Mutation

  Randomly selects cluster and replaces its current GST node by another randomly selected node

Selection / Replacement

- 50% of the population in specific generation is selected and carried to next generation based on the following two rules:
  - 5% of the current population is selected to be carried over to next generation using elitism
  - Remaining 45% of the current population is selected based on the ranking probability distribution function (Michalewicz, 1996)

Computational Experiments

- 169 TSPLIB instances – Feremans (2001)

  TSPLIB instances for GMST problem represent complete graphs with distances satisfying the triangle inequality, with up to 230 nodes in the network

- 42 random instances – Golden et al. (2004)

  These instances represent incomplete graphs with random distances ranging in size from 15 clusters, 120 nodes, and 2,000 edges; to 40 clusters, 200 nodes and 15,000 edges.

All computations performed on Xeon 2.66 GHz with 2 GB RAM.

Computational Experiments (cont.)

- Local Search parameters
  - Number of iterations: 500
- GA parameters
  - Initial population size: 100
  - Number of generations: 15
  - GA operators:
    - LSE Crossover: 50%
    - Random Mutation: 50%

Computational Results

Problem Set	Instances	# opt	B&C Alg.	LS			GA		
			Time (sec)	# opt	% gap	Time (sec)	# opt	% gap	Time (sec)
TSPLIB, geog	41	35	1397.4	35	0.000%	8.61	35	0.000%	4.94
TSPLIB, m = 3	32	22	707.64	22	0.000%	35.37	22	0.000%	10.37
TSPLIB, m = 5	32	22	941.12	22	0.000%	3.56	22	0.000%	3.92
TSPLIB, m = 7	32	27	981.78	27	0.000%	2.62	27	0.000%	3.41
TSPLIB, m = 10	32	27	75.93	27	0.000%	1.66	27	0.000%	2.00
TSPLIB	169	133	855.19	133	0.000%	9.57	133	0.000%	4.76
Random	42	42	319.2	40	0.119%	6.33	40	0.028%	5.96
Summary	**211**	**175**	**726.55**	**173**	**0.029%**	**8.82**	**173**	**0.007%**	**5.04**

Table 1. Performance of B&C Algorithm, Local Search and Genetic Algorithm with respect to solutions for which optimum is known

Computational Results

Unsolved TSPLIB Instances

- GA found the best known upper bound in all instances
- Out of 36 instances for which the optimum is not known, LS was worse than GA in 8 instances, with 0.14% average gap.

Concluding Remarks

- The two heuristic procedures, LS and GA, rapidly provide optimal or near-optimal solutions in all tested instances, with significantly shorter cpu times

The Minimum Labeling Spanning Tree Problem: Heuristic and Metaheuristic Approaches

Yupei Xiong

Bruce Golden

Edward Wasil

INFORMS Annual Meeting
Denver, Colorado
October 2004

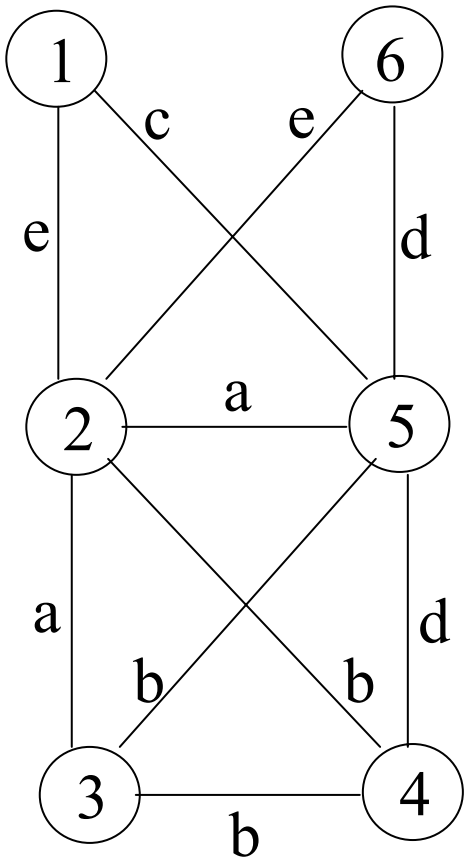
Introduction

- The Minimum Labeling Spanning Tree (MLST) Problem
  - Communications network design
  - Edges may be of different types or media (e.g., fiber optics, cable, microwave, telephone lines, etc.)
  - Each edge type is denoted by a unique letter or color
  - Construct a spanning tree that minimizes the number of colors

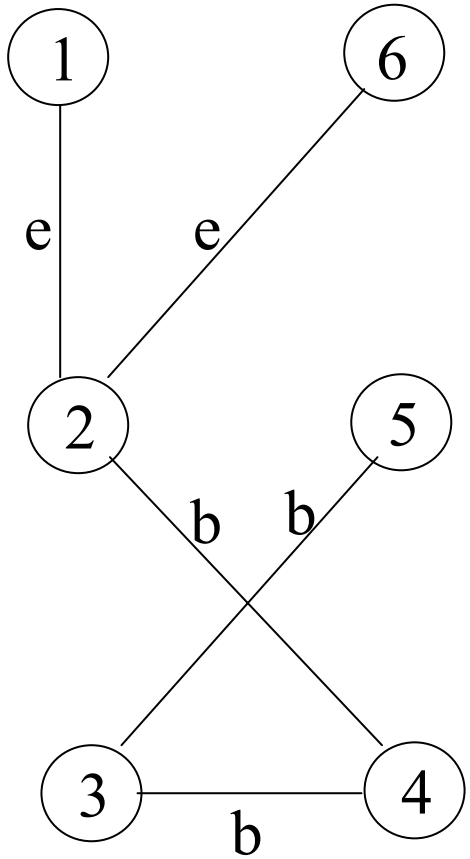
Introduction

- A Small Example

Input



Solution



Literature Review

- Where did we start?
  - The MLST Problem is NP-hard
  - Several heuristics had been proposed
  - One of these, MVCA (version 2), was very fast and effective
  - Worst-case bounds for MVCA had been obtained

Literature Review

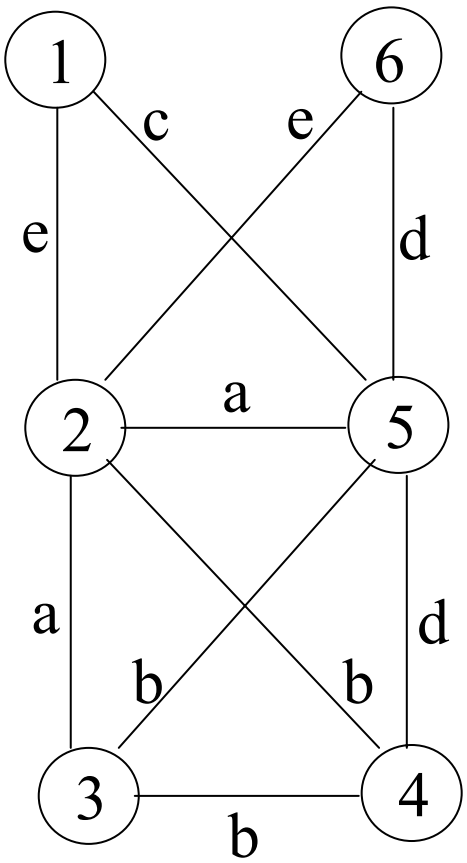
- An optimal algorithm (using backtrack search) had been proposed
- On small problems, MVCA consistently obtained nearly optimal solutions
- See [Chang & Leu, 1997], [Krumke & Wirth, 1998], [Wan, Chen & Xu, 2002], and [Bruggemann, Monnot & Woeginger, 2003]

Description of MVCA

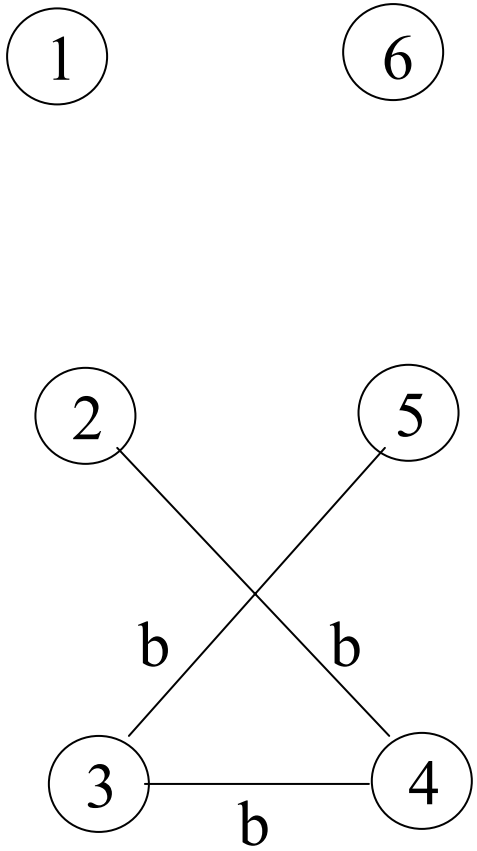
0. Input: $G(V, E, L)$.
1. Let $C \leftarrow \{\}$ be the set of used labels.
2. repeat
3. Let H be the subgraph of G restricted to V and edges with labels from C.
4. for all $i \in L - C$ do
5. Determine the number of connected components when inserting all edges with label i in H.
6. end for
7. Choose label i with the smallest resulting number of components and do: $C \leftarrow C \cup \{i\}$.
8. Until H is connected.

How MVCA Works

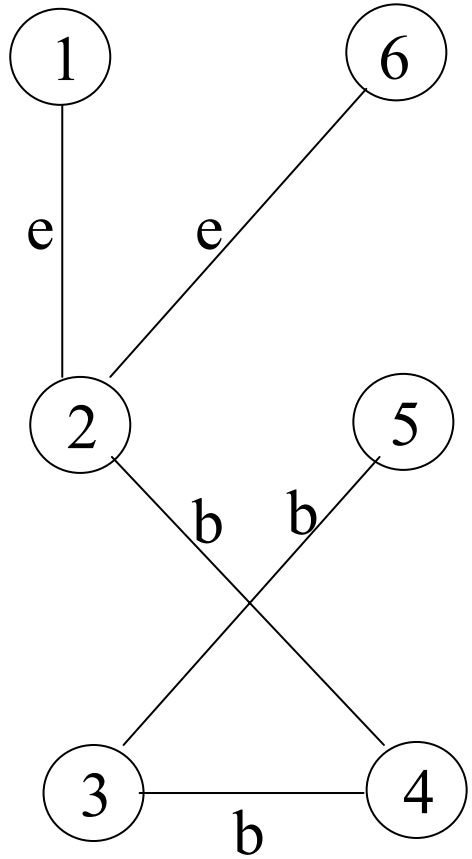
Input



Intermediate Solution



Solution



Worst-Case Results

1. Krumke, Wirth (1998): $\frac{\text{MVCA}}{\text{OPT}} \le 1 + 2\ln n$

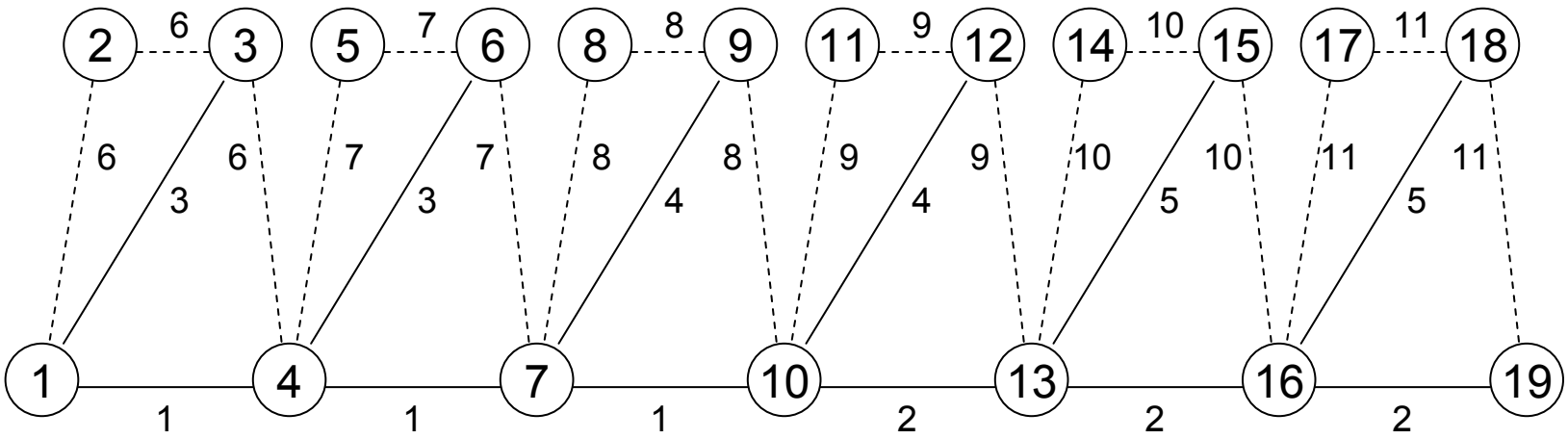
2. Wan, Chen, Xu (2002): $\frac{\text{MVCA}}{\text{OPT}} \le 1 + \ln(n-1)$

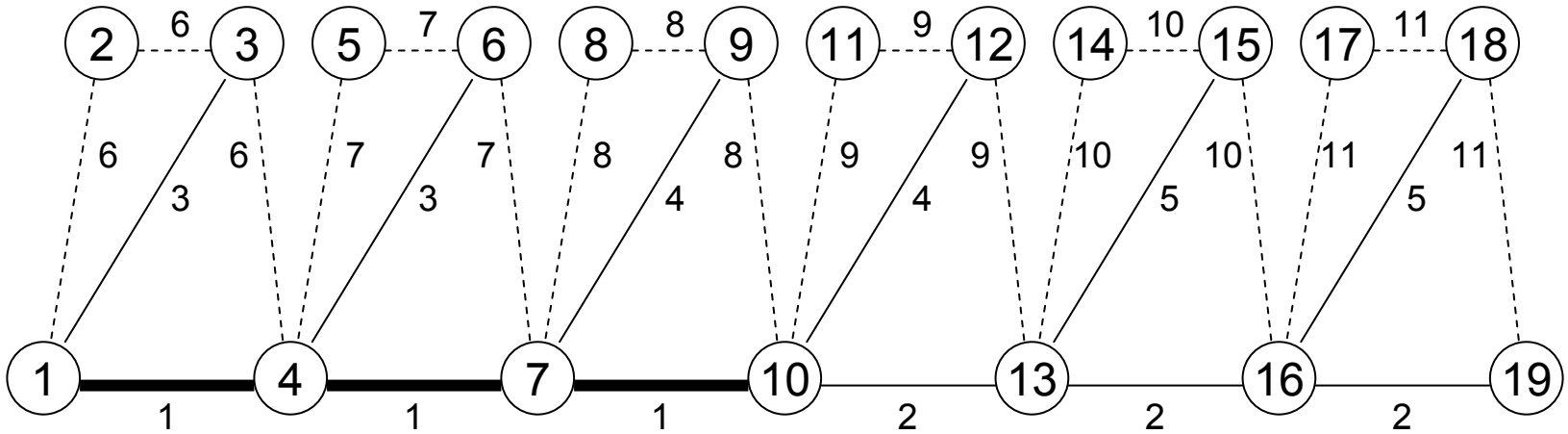
3. Xiong, Golden, Wasil (2005): $\frac{\text{MVCA}}{\text{OPT}} \le H_b = \sum_{i=1}^{b} \frac{1}{i} < 1 + \ln b$

where b = max label frequency, and

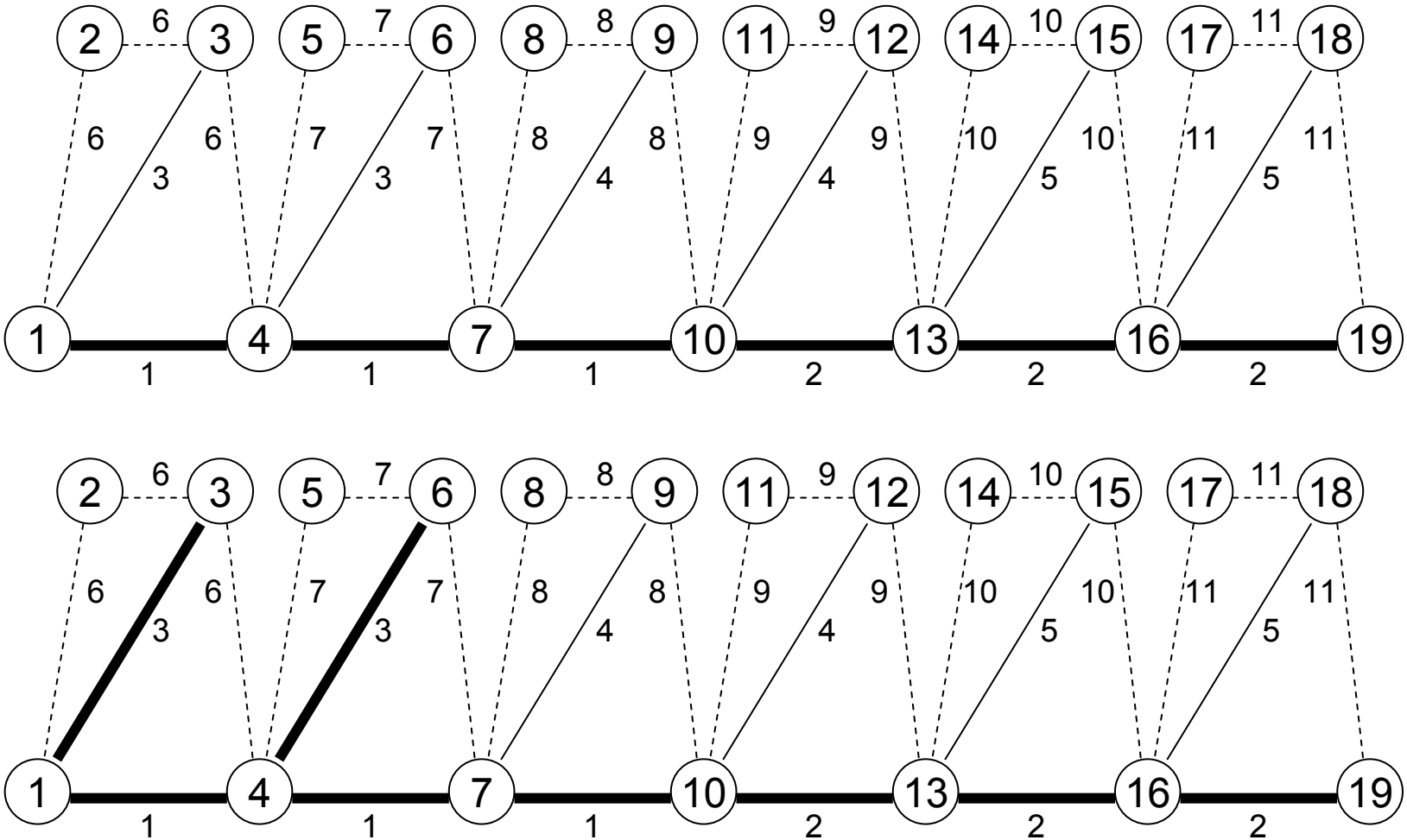
H_b= bth harmonic number

A Worst-Case Example

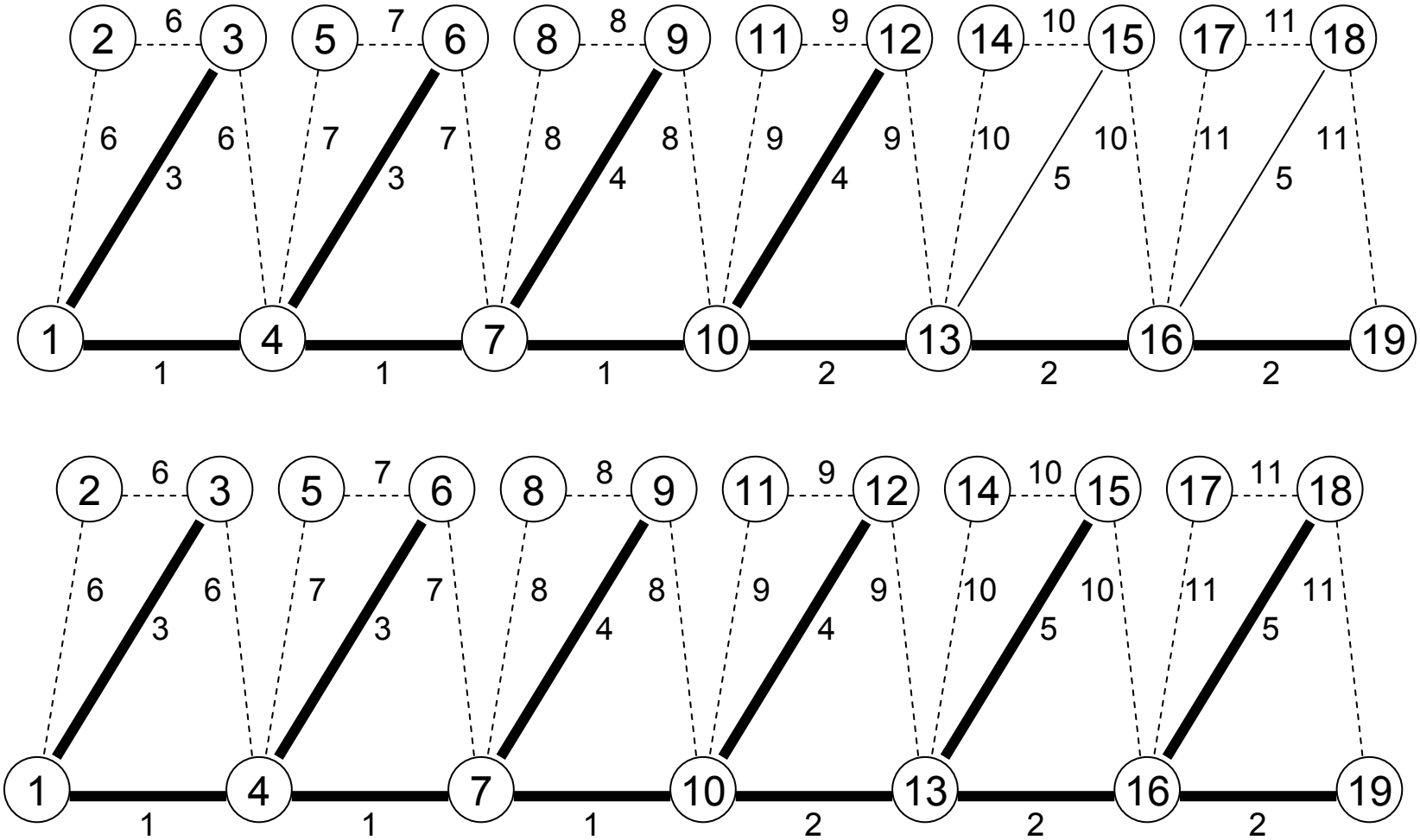




A Worst-Case Example - continued



A Worst-Case Example - continued



Some Observations

- The Xiong, Golden, Wasil worst-case bound is tight
- For the worst-case example with $b = 3$ and $n = 19$,

$$\frac{\text{MVCA}}{\text{OPT}} = \frac{11}{6} = 1 + \frac{1}{2} + \frac{1}{3} = H_3$$

- Unlike the MST, where we focus on the edges, here it makes sense to focus on the labels or colors
- Next, we present a genetic algorithm (GA) for the MLST problem

Genetic Algorithm: Overview

- Randomly choose p solutions to serve as the initial population
- Suppose s [0], s [1], … , s [p – 1] are the individuals (solutions) in generation 0
- Build generation k from generation k – 1 as below

    For each j between 0 and p – 1, do:

        t [j] = crossover { s [j], s [(j + k) mod p] }

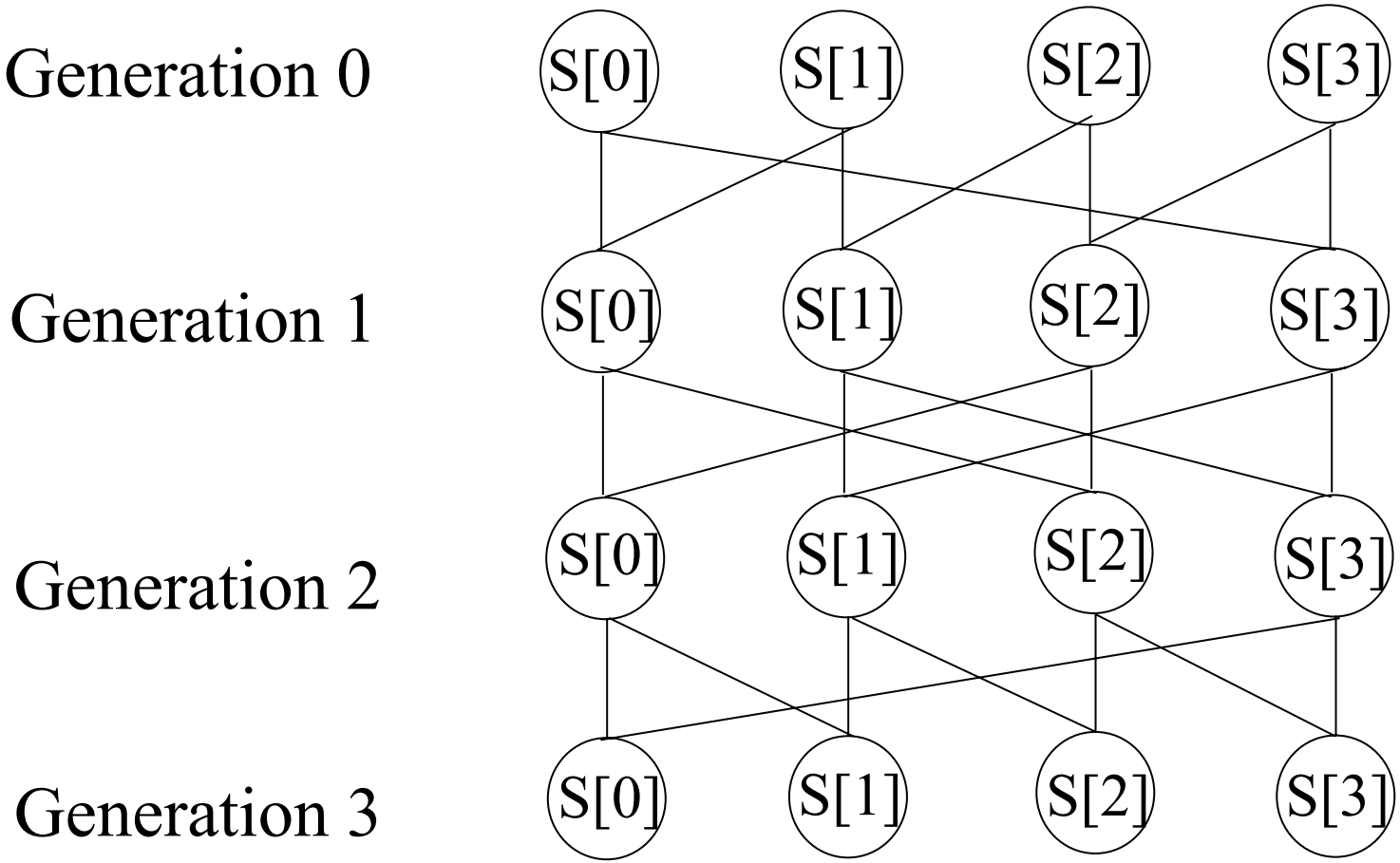
        t [j] = mutation { t [j] }

        s [j] = the better solution of s [j] and t [j]

    End For

- Run until generation p – 1 and output the best solution from the final generation

Crossover Schematic (p = 4)



Crossover

- Given two solutions s [1] and s [2], find the child T = crossover { s [1], s [2] }
- Define each solution by its labels or colors
- Description of Crossover

    a. Let S = s [1] $\cup$ s [2] and T be the empty set

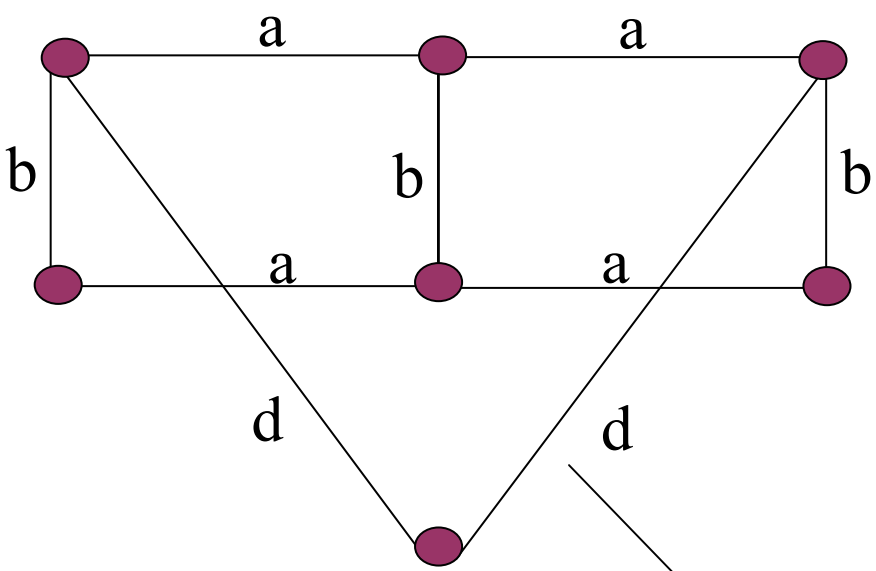
    b. Sort S in decreasing order of the frequency of labels in G

    c. Add labels of S, from the first to the last, to T until T represents a feasible solution

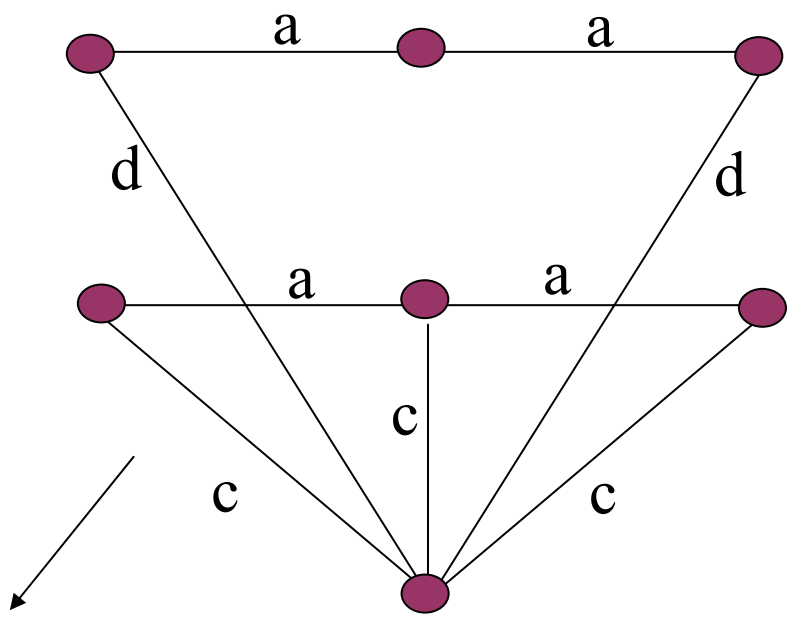
    d. Output T

An Example of Crossover

s [1] = { a, b, d }



s [2] = { a, c, d }



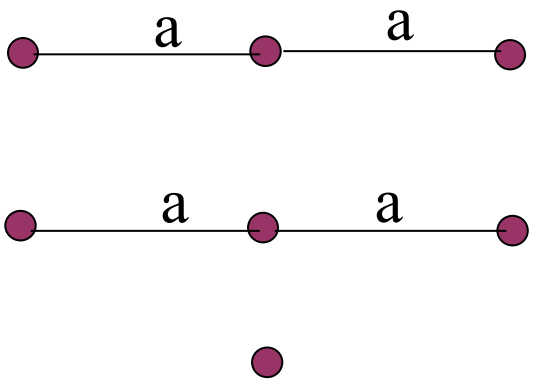
T = { }

S = { a, b, c, d }

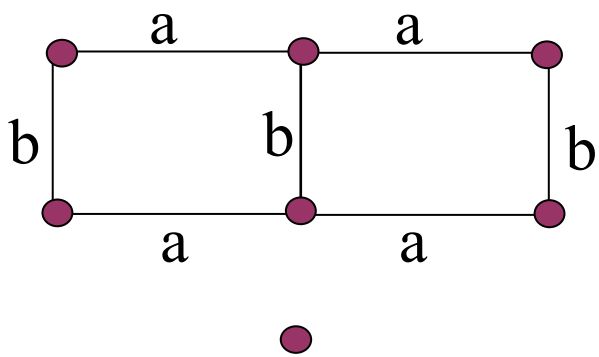
Ordering: a, b, c, d

An Example of Crossover

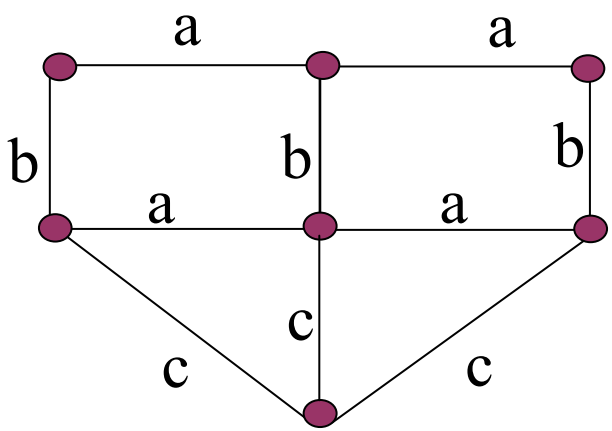
T = { a }



T = { a, b }



T = { a, b, c }



Mutation

- Given a solution S, find a mutation T
- Description of Mutation

  a. Randomly select c not in S and let T = S $\cup$ c

  b. Sort T in decreasing order of the frequency of the labels in G

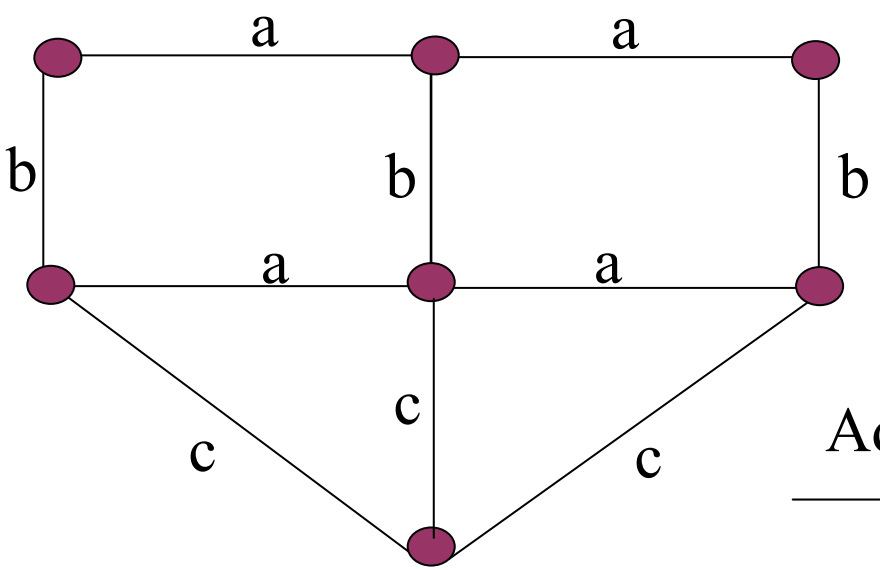
  c. From the last label on the above list to the first, try to remove one label from T and keep T as a feasible solution

  d. Repeat the above step until no labels can be removed

  e. Output T

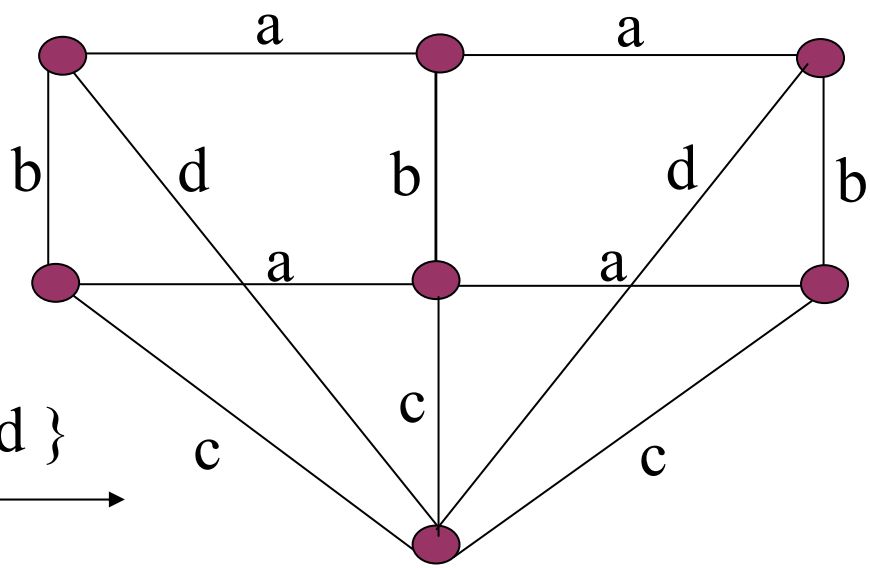
An Example of Mutation

S = { a, b, c }



Add { d }

S = { a, b, c, d }



Ordering: a, b, c, d

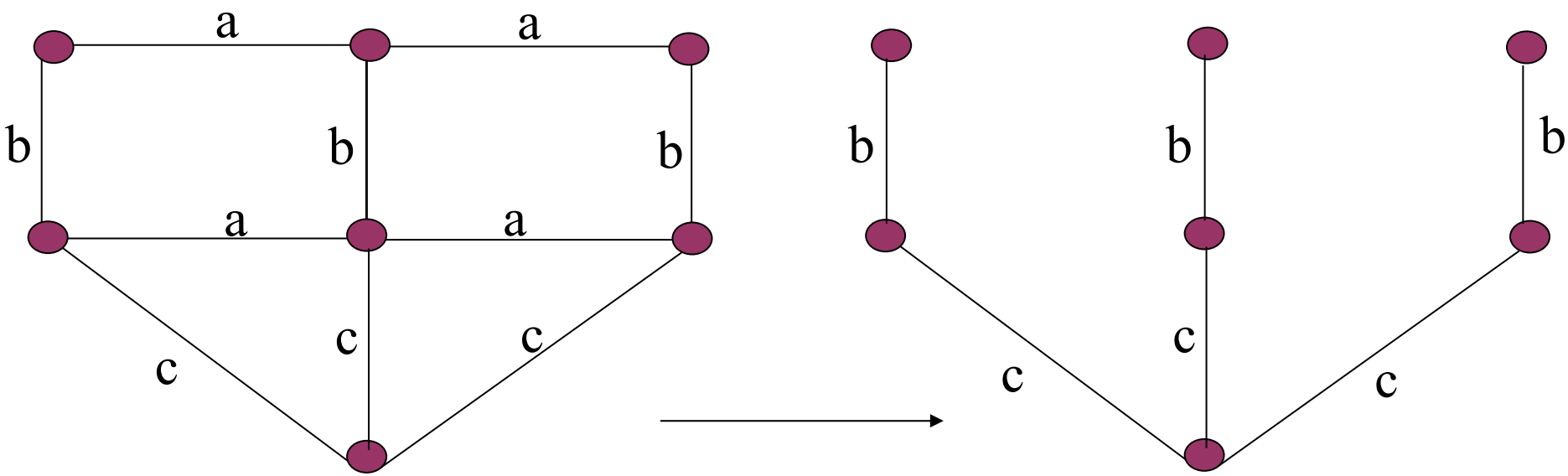
An Example of Mutation

Remove { d }

S = { a, b, c }

Remove { a }

S = { b, c }



T = { b, c }

Computational Results

- 78 combinations: n = 20 to 200 / l = 12 to 250 / density = .2, .5, .8 / p = 30 if n > 100; otherwise p = 20
- 20 sample graphs for each combination
- The average number of labels is compared
- GA beats MVCA in 58 of 78 cases (17 ties, 3 worse)
- The optimal solution was obtained using backtrack search in 740 instances

Computational Results - continued

- The GA obtains optimal solutions in 701 or 94.73% of these instances
- GA is slightly slower than MVCA
- GA required at most 2 seconds CPU time per instance on a Pentium 4 PC with 1.80 GHz and 256 MB RAM
- Backtrack search was run for at most 20 minutes per instance

Conclusions & Future Work

- The GA is fast, conceptually simple, and powerful
- It contains a single parameter
- We think GAs can be applied successfully to a host of other NP – hard problems
- Future work
  - Add edge costs
  - Examine other variants

The Euclidean Non-uniform Steiner Tree Problem

by

Ian Frommer

Bruce Golden

Guruprasad Pundoor

INFORMS Annual Meeting
Denver, Colorado
October 2004

Introduction

- The Steiner Tree Problem (STP)
  - We are given a set of terminal nodes
  - We want to find edges to connect these nodes at minimum cost
  - Additional nodes (Steiner nodes) may be added to the terminal nodes in order to reduce overall cost
  - Applications: laying cable networks, printed circuits, routing of transmission lines, design of communication networks

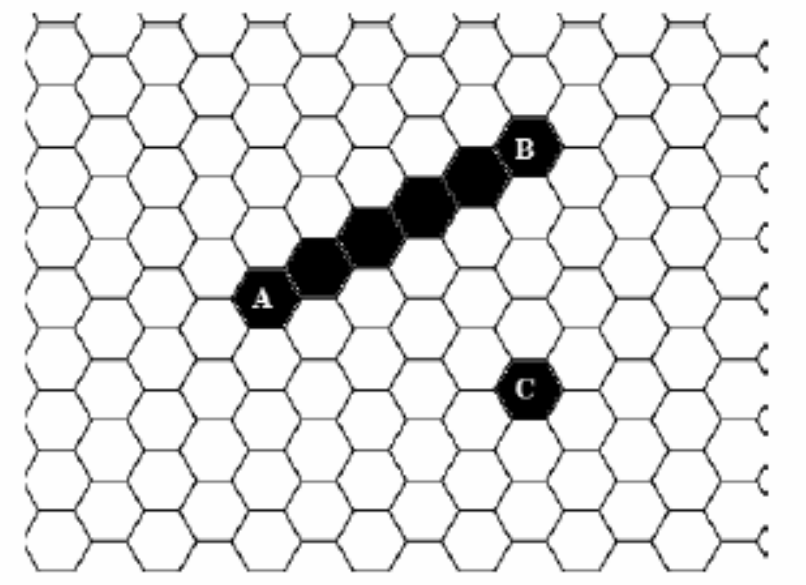
Introduction

- The Euclidean Non-uniform Steiner Tree Problem
  - Many STP variants have been studied
  - They have been shown to be NP-hard
  - In the Euclidean Non-uniform STP (ENSTP), the cost of an edge depends on its location as well as its distance
  - Certain streets are more expensive to rip apart and re-build than others
  - The ENSTP was introduced by Coulston (2003)

Description of New Variant

- Grid Structure
  - Use a hexagonal tiling of the plane
  - Each tile or cell has six adjacent neighbors
  - The distances between centers of adjacent cells are equal
  - Each cell has a cost and it may contain at most one of the nodes in the graph
  - Two nodes can be connected directly only if a straight line of cells can be drawn between the cells containing the two nodes

Hexagonal Grid



- A and B are directly connected, A and C are not

Determining Cost

- When an edge connects cells A and B, the cost of the edge is the sum of the costs associated with all the intermediate cells
- The cost of the tree includes the edge costs plus the costs corresponding to each node (terminal and Steiner) in the tree
- We may charge an additional fee for each Steiner node
- In some applications, Steiner nodes may require the installation of additional hardware

Genetic Algorithm for the ENSTP

1. Input: terminal node set, grid cost structure

2. Generate Initial Population randomly

   Repeat Steps 3 to 7 for TMAX iterations

   3. Find Fitness (= cost) of each individual

   4. Select parents via Queen Bee Selection

   5. Apply Spatial-Horizontal Crossover to parents to produce offspring

   6. Mutation 1 – add Steiner nodes at edge crossings

   7. Mutation 2 – randomly move Steiner nodes

Initial Population

- We use a population size of 40
- Each individual is a fixed-length chromosome of Steiner node locations (coordinates)
- Checks are performed to ensure that Steiner node locations and terminal node locations do not coincide
- Otherwise, locations are randomly selected

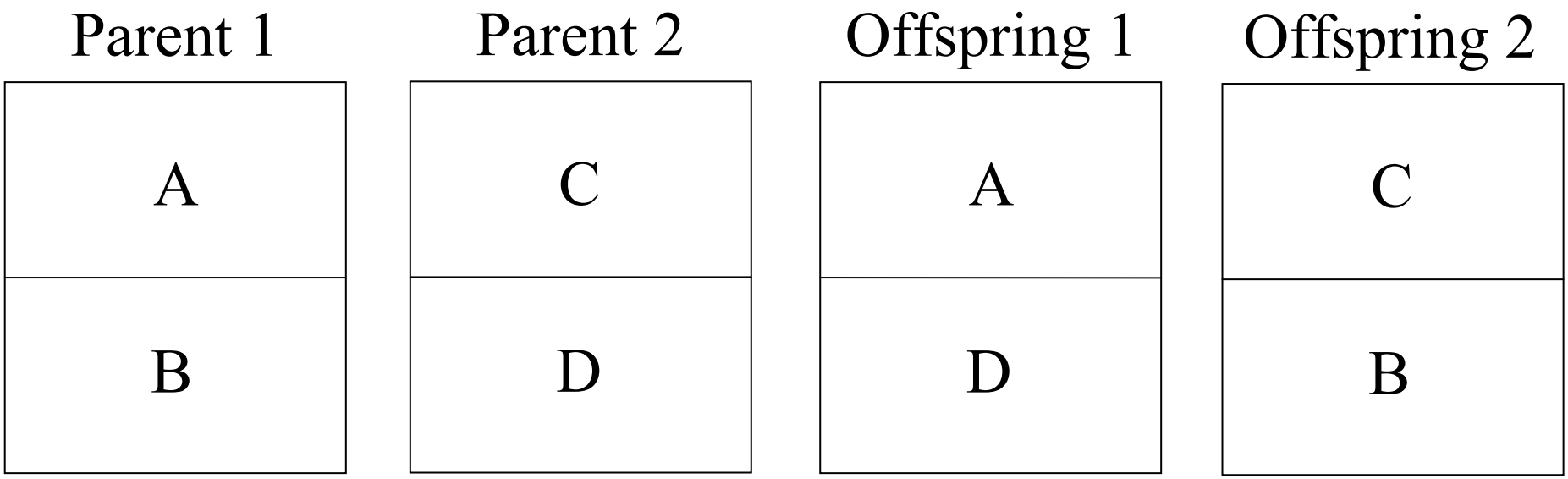
Fitness

- For each individual, form the complete graph over the terminal nodes and Steiner nodes
- Find a MST solution
- Remove degree-1 Steiner nodes and their incident edges
- Fitness is the cost of the resulting Steiner tree

Queen Bee Selection

- The fittest individual (the Queen Bee) mates with each other member of the population to produce two offspring
- This adds 78 offspring
- The 40 fittest of the 40 parents plus 78 offspring are chosen to survive to the next generation

Spatial-Horizontal Crossover



- A, B, C, and D are sets of Steiner nodes
- A and C can have some common nodes, as can B and D
- Terminal nodes are not shown above

More About the GA

- Mutation operations add and move Steiner nodes
- The ENSTP can be converted to a Steiner problem in graphs
- We have implemented the Dreyfus & Wagner algorithm to solve the Steiner problem in graphs optimally
- In preliminary computational experiments, we compare the GA solution to the optimal solution in small and medium-size problems

Preliminary Computational Results

Problem	Optimal	GA (best)	% Gap	GA (average)
1	11.138	11.138	0.0	11.230
2	10.001	10.102	1.0	10.284
3	4.806	4.811	0.1	4.819
4	4.158	4.206	1.2	4.236
5	5.605	5.605	0.0	5.605

- The GA was run 10 times on each problem

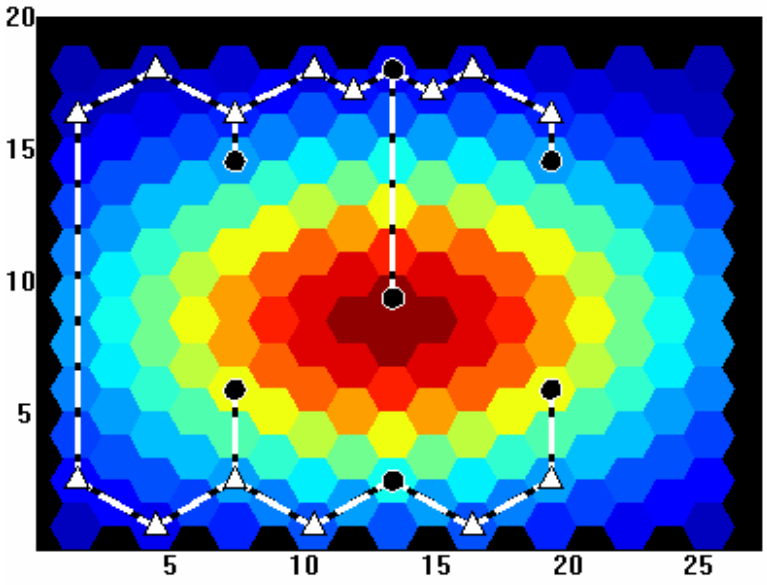
Computational Notes

- Grid size: 21 by 17
- 7 or 10 terminal nodes
- Coded in MATLAB, run on a 3.0 GHz machine with 1.5 GB of RAM
- Each GA run required about a minute
- The optimal algorithm also required a minute per problem

Comparison of Solutions

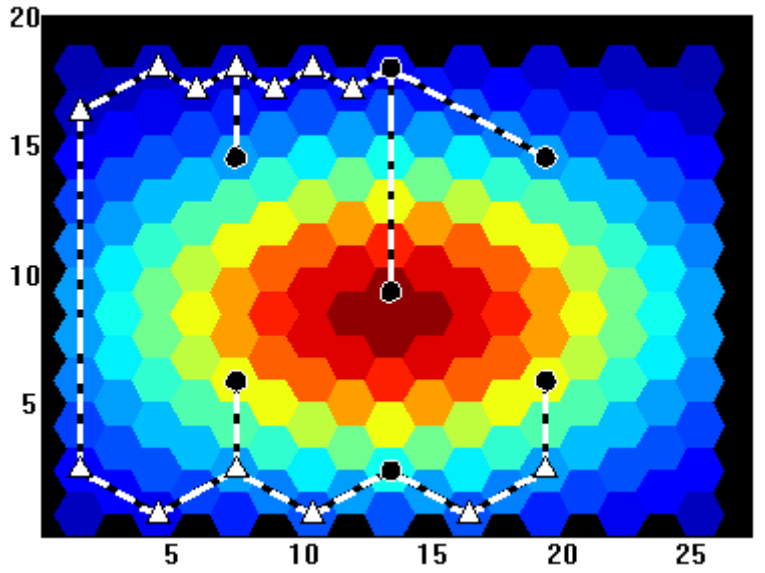
Optimal Solution

Cost = 10.001



Genetic Algorithm Solution

Cost = 10.102



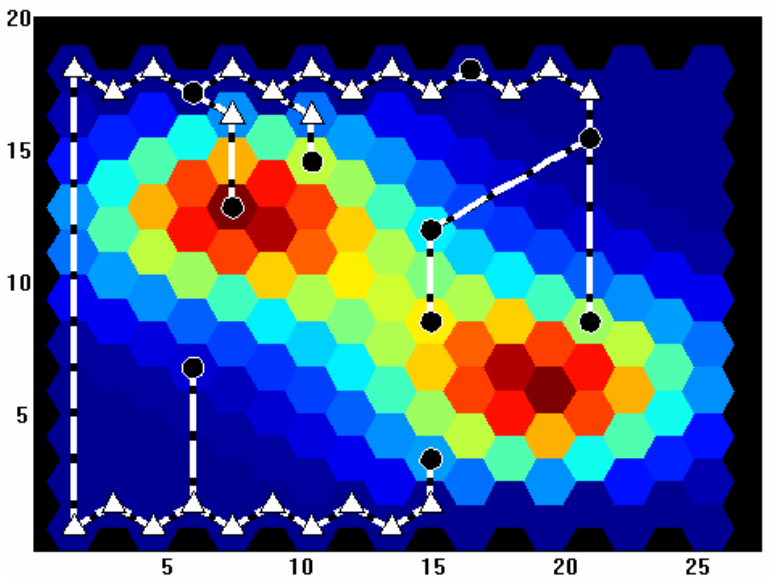
Terminal Nodes

Steiner Nodes

Comparison of Solutions

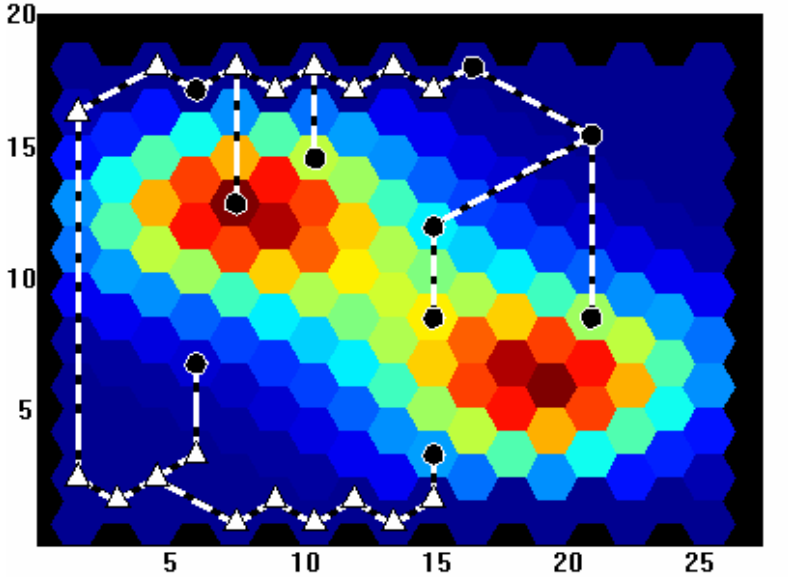
Optimal Solution

Cost = 4.806



Genetic Algorithm Solution

Cost = 4.811



● **Terminal Nodes**

△ **Steiner Nodes**

Two Medium-Size Problems

Problem	Optimal	GA	% Gap	D&C GA	% Gap
1	26.606	29.903	12.4	27.4673	3.2
2	31.3096	34.318	9.6	31.628	1.0

- The GA required 25 minutes per problem
- The optimal algorithm required 12 days per problem
- Divide & conquer GA required 2 minutes per problem
- For the above problems, grid size is 35 by 35 and there are 15 terminal nodes

Conclusions

- There have been many recent applications of heuristic approaches to network design problems
  - Simpler to implement than exact procedures
  - Heuristic approaches are more advanced than before
  - Combining metaheuristics such as GAs with local search is a powerful tool
  - Average processor speed of PCs continues to increase

Conclusions

- We have provided some guidelines, especially, with respect to GAs
- We have described four successful applications of heuristic search to network design problems
- In the process, we have tried to illustrate the simplicity and flexibility of the GA approach