

Research Incubator: Combinatorial Optimization

Dr. Lixin Tao

December 9, 2003

Content

- General Nature of Research on Combinatorial Optimization
- Problem Identification and Abstraction
- Problem Properties and Solution Space Design
- Solution Meta-Heuristics and Their Application
 - Exhaustive Search
 - Local Optimization
 - Simulated Annealing
 - Tabu Search
 - Genetic Algorithm
- Experiment Design for Performance Evaluation
- Common Research Pitfalls

DPS Research Challenges

- Various educational and work background, some are weak in computing and math
- Limited in-person meetings with supervisors and instructors
- Finishing theses within three years of part-time study

Research Incubators

- Identify a set of research areas suitable for DPS dissertations
- For each area, identify recurring concepts and reusable knowledge
- Introduce an area to students with an intuitive presentation of sample research projects
- Complement the presentation with an easily readable tutorial for in-depth explanation of the concepts and methodologies, sample thesis outlines, and reference list
- If applicable, provide sample implementation of the methodologies for the sample projects

General Nature of CO Research

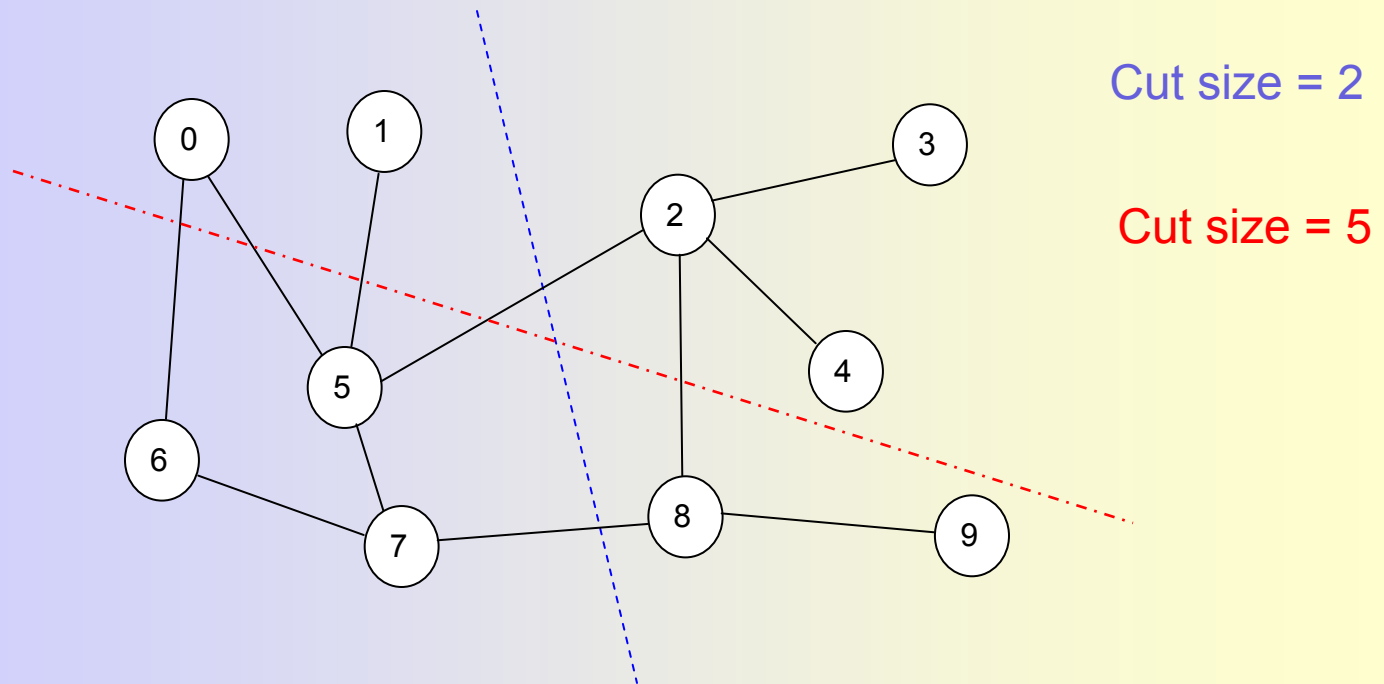
- Develop *reusable* knowledge in problem solving
 - Identify unresolved useful problems
 - Check out its current status and potential solution tools
 - Abstract and formulate the problem
 - Design solution algorithms
 - Design experiments to evaluate performance of your algorithm

Real World Problems

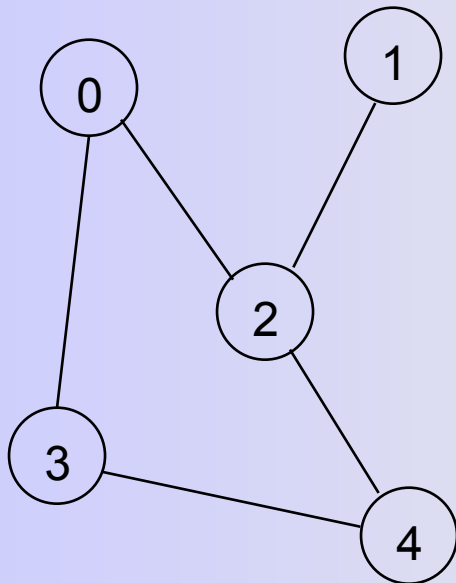
- **VLSI circuit partition**
 - How to implement a large circuit on two chips with balanced component distribution and least connections crossing the chips?
- **Load balance of software components on clustered servers**
 - How to load-balance software components on two server machines and minimize inter-processor communications?

Graph Bisection

How to cut the vertices into two partitions such that the number of edges cut is minimized?



Graph Representation: Adjacency Matrix



	0	1	2	3	4
0	0	0	1	1	0
1	0	0	1	0	0
2	1	1	0	0	1
3	1	0	0	0	1
4	0	0	1	1	0

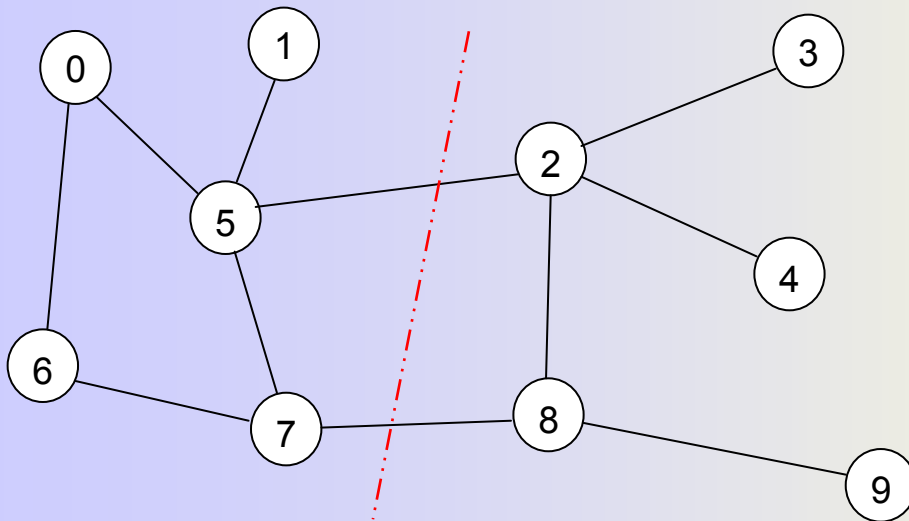
$$a(i, j) = \begin{cases} 1 & \text{If there is edge } (i, j) \\ 0 & \text{otherwise} \end{cases}$$

Represent Partition as a Vector

0	1	2	3	4	5	6	7	8	9
0	0	1	1	1	0	0	0	1	1

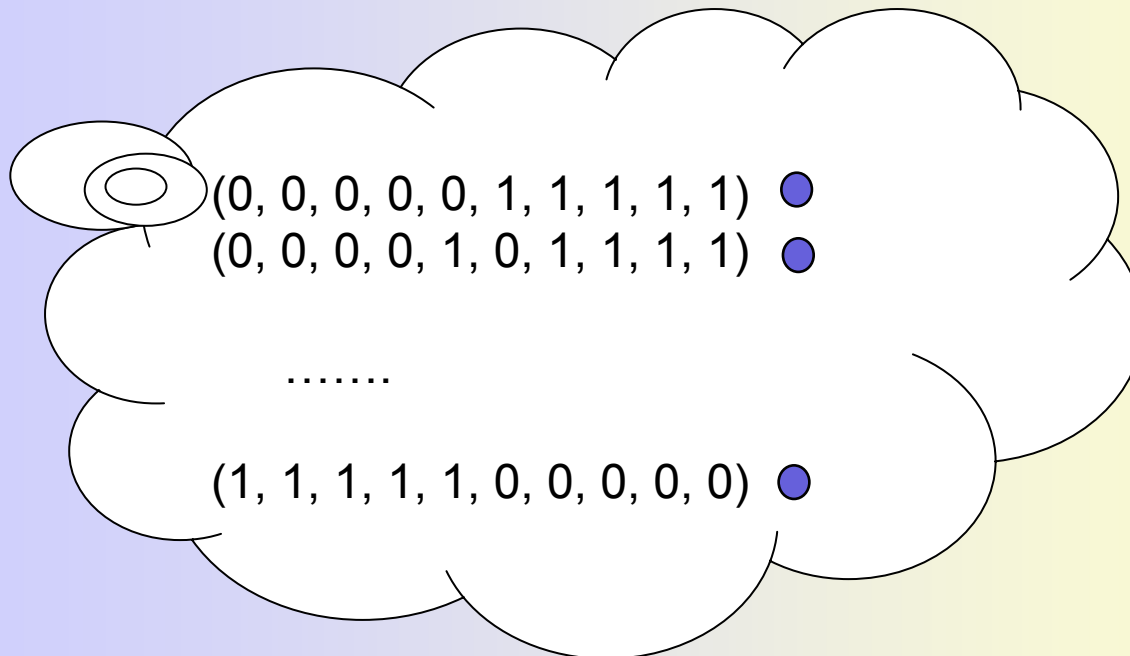
$P[i] = 0$: vertex i is in left partition

$P[i] = 1$: vertex i is in right partition



Solution Space

- Solution space here is made up of all vectors of five 1s and five 0s
- The number of feasible solutions is exponential



NP-Hard Problem

- Any solution algorithm to it has running time $O(2^n)$ where n is problem instance size
- For larger problem instances, no algorithm can produce optimal solution within reasonable amount of time
- 2^{100} operations take 10^{10} years for a supercomputer running at 10^{15} ops per sec.
- Only heuristics can help find *optimized* solutions
- Many engineering problems are NP-hard

Exhaustive Search

For each $(n/2)$ -combination of set V

Let vertices in the $(n/2)$ -combination go to the left partition.

Let other vertices go to the right partition.

Evaluate the cost of the resulting partition.

If the cost improves the best one seen so far, record it.

End For.

Return the best partition visited.

Repeated Random Solutions

Repeat 100 times

Generate a random partition.

Evaluate the cost of the resulting partition.

If the cost improves the best one seen so far, record it.

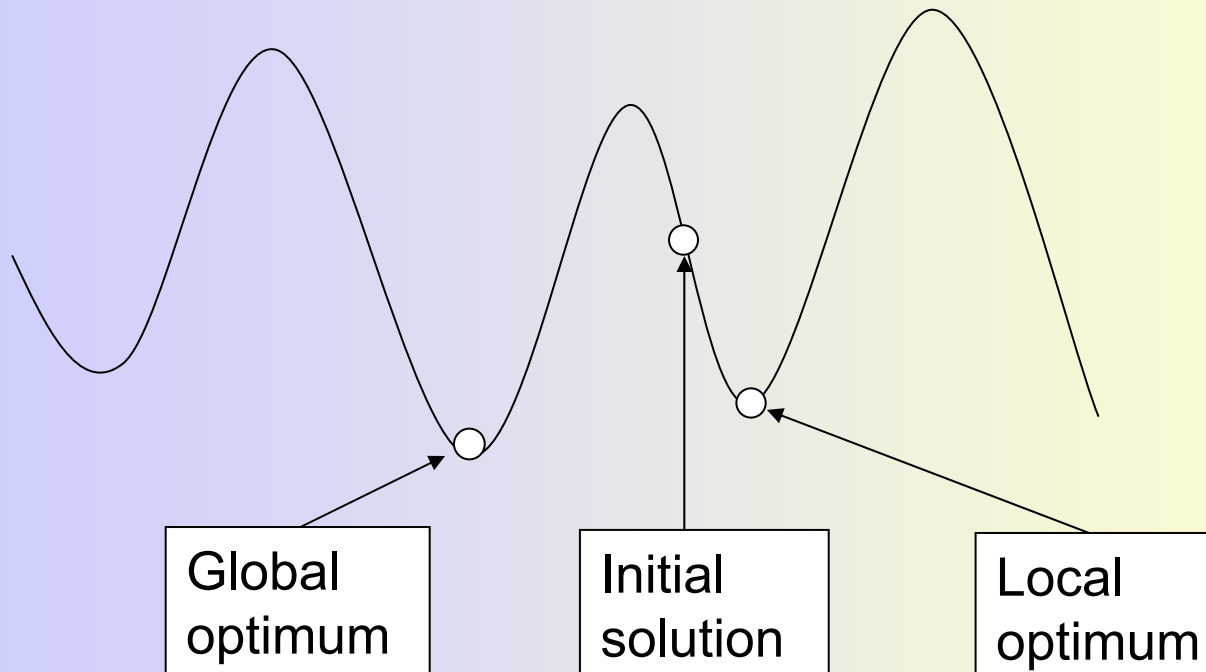
End Repeat.

Return the best partition visited.

Local Optimization

Get a random initial partition as the current partition.
While there is any improvement in the last 100 iterations
 Generate a random neighbor of the current partition.
 Evaluate the neighbor's cost.
 If the neighbor's cost improves the current cost
 Let the neighbor be the new current partition.
 If the neighbor's cost improves the best one seen
 so far, record it.
 End If.
End While.
Return the best partition visited.

Local and Global Optimums



Simulated Annealing

Get a random initial partition π as the current partition.

Get an initial temperature $t > 0$.

While stopping criteria not met do

 Perform the following loop L times.

 Let π' be a random neighbor of π .

 Let $\Delta = \text{cost}(\pi') - \text{cost}(\pi)$.

 If $\Delta < 0$ (downhill move), set $\pi = \pi'$.

 If $\Delta > 0$ (uphill move), set $\pi = \pi'$ with probability $e^{-\Delta/t}$

 Set $t = 0.95t$ (reduce temperature).

End While.

Return the best π visited.

Tabu Search

Get a random initial partition π as the current partition.

While stopping criteria not met do

 Let π' be a neighbor of π minimizing

$$\Delta = \text{cost}(\pi') - \text{cost}(\pi)$$

 and not visited in the last t iterations.

 Set $\pi = \pi'$.

End While.

Return the best π visited.

Solution Coding for Genetic Algorithm

Original coding

	0	1	2	3	4	5	6	7
<i>c</i>	0.3	0.1	0.8	0.5	0.3	0.2	0.9	0.7

After sorting

	1	5	0	4	3	7	2	6
<i>c</i>	0.1	0.2	0.3	0.3	0.5	0.7	0.8	0.9

	0	1	2	3	4	5	6	7
<i>p</i>	0	0	1	1	0	0	1	1

GA Major Operations

- **Selection:** choose parents
- **Crossover:** randomly mix two parents up to generate a child
- **Mutation:** randomly modify a parent to generate a child

Genetic Algorithm

Generate 50 codings of random partitions, and sort them in the generation table according to their costs.

While there are improvements to the best cost seen so far in the last 50 iterations do

- Use crossover to generate 40 children with randomly chosen parents, and insert them into the generation table according to their costs.

- Use mutation to generate 10 children with randomly chosen parents, and insert them into the generation table according to their costs.

The generation table only keeps the best 50 codings.

If the best coding improves the best cost seen so far, record it.

End While.

Return the best partition visited.

Experiment Design for Performance Evaluation

- Collect data from real world (preferred), or generate random data
- Compare both solution quality and running time with the best solution techniques available now

Common Research Pitfalls

- The problem doesn't have practical value
- Strictly follow book descriptions of a meta-heuristic
- Solve any given problem with meta-heuristics
- Solve a problem without proper problem abstraction and formulation
- Formulate a problem based on a meta-heuristic

Common Research Pitfalls ...

- Parameters need be adjusted for different problem instances
- Performance evaluation does not cover running time
- Performance evaluation only for trivially small problem instances
- Methodology evaluation based on user survey only
- Human help during heuristic execution

Research Incubator Resources

- Download slide file, tutorial, and complete Java implementation of all the meta-heuristics at
 - <http://csis.pace.edu/~lixin/dps>
- Play with the project
- Crash it
- Improve it
- Invent your own new meta-heuristics!