P2P Combinatorial Optimization

Amir H. Payberah (amir@sics.se)
Agenda

• Introduction to Optimization
• Metaheuristics in Combinatorial Optimization
• P2P Combinatorial Optimization
Introduction to Optimization
Objective

• Objective Function
  ▪ Max (Min) some function of decision variables
  ▪ Subject to some constraints
    • Equality (=)
    • Inequality (<, >, ≤, ≥)

• Search Space
  ▪ Range or values of decisions variables that will be searched during optimization.
Types of Solutions

• **Solution** specifies the values of the decision variables, and therefore also the value of the objective function.

• **Feasible solution** satisfies all constraints.

• **Optimal solution** is feasible and provides the best objective function value.

• **Near-optimal solution** is feasible and provides a superior objective function value, but not necessarily the best.
Continuous vs Combinatorial

• Continuous
  ▪ An infinite number of feasible solutions.
  ▪ Generally maximize/minimize a function of continuous variables such as 4x+5y where x and y are real numbers.

• Combinatorial
  ▪ A finite number of feasible solutions.
  ▪ Generally maximize/minimize a function of discrete variables such as 4x+5y where x and y are countable numbers.
Combinatorial Optimization

Combinatorial optimization is the mathematical study of finding an optimal arrangement, grouping, ordering, or selection of discrete objects usually finite in numbers.

- Lawler, 1976
Aspects of Optimization Problem

- Continuous or Combinatorial
- Search space size
- Degree of constraints
- Single or multiple objectives
- Deterministic or Stochastic
  - **Deterministic**: all variables are deterministic.
  - **Stochastic**: the objective function and/or some decision variables and/or some constraints are *random* variables
# Simple and Hard Problems

<table>
<thead>
<tr>
<th>Simple</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Few decision variables</td>
<td>Many decision variables</td>
</tr>
<tr>
<td>Differentiable</td>
<td>Combinatorial</td>
</tr>
<tr>
<td>Objective easy to calculate</td>
<td>Objective difficult to calculate</td>
</tr>
<tr>
<td>No or light constraints</td>
<td>Severely constraints</td>
</tr>
<tr>
<td>Feasibility easy to determine</td>
<td>Feasibility difficult to determine</td>
</tr>
<tr>
<td>Single objective</td>
<td>Multiple objective</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Stochastic</td>
</tr>
</tbody>
</table>
Simple and Hard Problems

Simple
- Few decision variables
- Differentiable
- Objective easy to calculate
- No or light constraints
- Feasibility easy to determine
- Single objective
- Deterministic

Hard
- Many decision variables
- Combinatorial
- Objective difficult to calculate
- Severely constraints
- Feasibility difficult to determine
- Multiple objective
- Stochastic

Enumeration or exact methods such as mathematical programming or branch and bound will work best.

For these, **heuristics** are used.
Heuristics

- **Heuristics** are rules to search to find optimal or near-optimal solutions.

- Heuristics can be
  - **Constructive**: build a solution piece by piece.
  - **Improvement**: take a solution and alter it to find a better solution.
Metaheuristics

Metaheuristics is a rather unfortunate term often used to describe a major subfield, indeed the primary subfield, of stochastic optimization. Stochastic optimization is the general class of algorithms and techniques which employ some degree of randomness to find optimal (or as optimal as possible) solutions to hard problems.

- Sean Luke, 2009
Metaheuristics in Combinatorial Optimization
Optimization Problem under Uncertainty

• Two aspects to be defined:
  ▪ The way uncertain information is formalized.
  ▪ The dynamicity of the model.
Optimization Problem under Uncertainty

- All information is available at decision stage.
  - Traveling Salesman Problem
Optimization Problem under Uncertainty

- Describe uncertain information by means of random variables of known probability distribution.
  - Probabilistic Traveling Salesman Problem
Optimization Problem under Uncertainty

- Identify the uncertain information with fuzzy quantities and constraints with fuzzy set.
Optimization Problem under Uncertainty

- The uncertain information is known in the form of interval values.
  - No knowledge about the probability distribution of random data is known.
  - Traveling Salesman Problem: the cost of arcs between couples of customers is given by interval values.
Optimization Problem under Uncertainty

- Input data is a sequence of data which are supplied to the algorithm incrementally.
- The algorithm produces the output incrementally, without knowing the complete input.
  - Dynamic Traveling Repair Problem
Stochastic Combinatorial Optimization Problems (SCOPs)

- Total uncertainty:
  - Pure Online Problems
  - Robust COPs
  - Fuzzy COPs
  - Stochastic COPs (SCOPs)
  - Deterministic COPs (DCOPs)

- Dynamicity:
  - Static
  - Dynamic
Metaheuristics for SCOPs

- Ant Colony Optimization
- Evolutionary Computation
- Simulated Annealing
- Tabu Search
- Stochastic Partitioning
- Progressive Hedging
- Rollout Algorithms
- Particle Swarm Optimization
- Variable Neighborhood Search
Metaheuristics for SCOPs

- Ant Colony Optimization
- *Evolutionary Computation*
- Simulated Annealing
- Tabu Search
- *Stochastic Partitioning*
- Progressive Hedging
- Rollout Algorithms
- *Particle Swarm Optimization*
- Variable Neighborhood Search
P2P Combinatorial Optimization
Towards a decentralized architecture for optimization

M. Biazzini, M. Brunato, A. Montresor

IPDPS - 2008
• They introduced a generic framework for the distributed execution of combinatorial optimization tasks.

• The description of the generic framework is based on particle swarm optimization.
• **PSO** is a metaheuristic based on the idea of simulating the flight of bird flocks.

• A set of particles is placed in the search space of a given optimization problem.

• Each particle evaluates the objective function corresponding to its current location.

• Then, each particle determines a move through the search space by combining the history of its own current and best locations with those of one or more particles of the swarm, with some random perturbations.
  
  ▪ \( v_i = v_i + c1 \ast \text{rand()} \ast (p_i - x_i) + c2 \ast \text{rand()} \ast (g - x_i) \)
  
  ▪ \( x_i = x_i + v_i \)

• After all particles have been moved, the next iteration starts.
The generic framework is composed of three modules:

- **Topology service** is responsible for creating and maintaining an overlay topology.

- **Function optimization service** evaluates the target function over a set of points in the search space.

- **Coordination service** coordinates the selection of points to be evaluated in the search space.
Topology Service: Peer Sampling

• It is provided by NEWSCAST.

• Each NEWSCAST node maintains a view containing $c$ node descriptors.

• Each NEWSCAST node periodically:
  - Selects a random peer from its partial view
  - Updates its local descriptor
  - Performs a view exchange with the selected peer, during which the two nodes send each other their views, merge them, and keep the $c$ freshest descriptors.
Function Optimization Service: Distributed PSO

• At each node $p$, the PSO function optimization service maintains and executes a particle swarm of size $k$.

• Each particle $i \in \{1, \ldots, k\}$ is characterized by its current position $p_i^\rho$, its current velocity $v_i^\rho$ and the local optimum $x_i^\rho$.

• Each swarm of a node $p$ is associated to a swarm optimum $g_i^\rho$, selected among the particles local optima.

• Different nodes may know different swarm optima. The best optimum among all of them is identified with the term global optimum, denoted $g$.

• The PSO function optimizer service works by iterating over the particles, updating the current position and velocity.
Coordination Service: Global Optimum Diffusion

- The *coordination service* should spread information about the global optimum among nodes.
- Periodically, each node $p$ initiates a communication with a random peer $q$.
- $p$ sends the pair $<g^o, f(g^o)>$ to $q$.
- When $q$ receives such a message, it compares the swarm optimum of $p$ with its local optimum.
- If $f(g^o) < f(g^q)$, then $q$ updates its swarm optimum with the received optimum; otherwise, it replies to $p$ by sending $<g^q, f(g^q)>$. 
Peer-to-peer optimization in large unreliable networks with branch-and-bound and particle swarm

M. Biazzini, B. Bánhelyi, A. Montresor, M. Jelasity

EvoCOMNET - 2009
Contribution

• They have proposed a P2P branch-and-bound (B&B) algorithm based on interval arithmetic.
Stochastic Partitioning Methods

- The search space is recursively partitioned in subspaces.
- The computation effort is concentrated on the sub-spaces that are estimated to be the most promising ones.
Branch and Bound

• Assume that the goal is to find the minimum value of a function $f(x)$, where $x$ ranges over some set $S$.

• A branch-and-bound procedure requires two tools:
  - **Branching**: A splitting procedure that given a set $S$ of candidates, returns two or more smaller sets whose union covers $S$.
  - **Bounding**: A procedure that computes upper and lower bounds for the minimum value of $f(x)$ within a given subset $S$.

• If the lower bound for some set of candidates $A$ is greater than the upper bound for some other set $B$, then $A$ may be safely discarded from the search.
  - It is usually implemented by maintaining a global variable $m$ that records the minimum upper bound seen among all subregions examined so far.
  - Any set whose lower bound is greater than $m$ can be discarded.
Peer Sampling and its Applications

• The algorithm uses NEWSCAST for peer sampling service.

• The peer sampling applications:

  ▪ *Gossip-based broadcasting*: nodes periodically communicate pieces of information they consider interesting to random other nodes.

  ▪ *Diffusion-inspired load balancing*: nodes periodically test random other nodes to see whether those have more load or less load, and then perform a balancing step accordingly.
Algorithm

- The algorithm is guaranteed to **eventually find the global minimum** (in the lack of the failure in the network).

- The basic idea is that the lowest known upper bound of the global minimum is **broadcast using gossip**.

- The intervals to be processed are distributed over the network using **gossip-based load balancing**.

- The lower bound for an interval is calculated using **interval arithmetic**.

- The algorithm is started by sending the search domain $D$ with lower bound $b = \infty$ to a random node.
Algorithm 1 P2P B&B

1: loop  \hspace{1cm} \triangleright \text{main loop}
2: \hspace{0.5cm} I \leftarrow \text{priorityQ.getFirst()} \hspace{1cm} \triangleright \text{most promising interval; if queue empty, blocks}
3: \hspace{0.5cm} (I_1, I_2) \leftarrow \text{branch}(I) \hspace{1cm} \triangleright \text{cut the interval in two along longest side}
4: \hspace{0.5cm} \min_1 \leftarrow \text{upperBound}(I_1) \hspace{1cm} \triangleright \text{minimum of 8 random samples from interval}
5: \hspace{0.5cm} \min_2 \leftarrow \text{upperBound}(I_2)
6: \hspace{0.5cm} \min \leftarrow \min(\min, \min_1, \min_2) \hspace{1cm} \triangleright \text{current best value known locally}
7: \hspace{0.5cm} b_1 \leftarrow \text{lowerBound}(I_1) \hspace{1cm} \triangleright \text{calculates bound using interval arithmetic}
8: \hspace{0.5cm} b_2 \leftarrow \text{lowerBound}(I_2)
9: \hspace{0.5cm} \text{priorityQ.add}(I_1, b_1) \hspace{1cm} \triangleright \text{queue is ordered based on lower bound}
10: \hspace{0.5cm} \text{priorityQ.add}(I_2, b_2)
11: \hspace{0.5cm} \text{priorityQ.prune}(\min) \hspace{1cm} \triangleright \text{remove entries with a higher lower bound than min}
12: \hspace{0.5cm} p \leftarrow \text{getRandomPeer()} \hspace{1cm} \triangleright \text{calls the peer sampling service}
13: \hspace{0.5cm} \text{sendMin}(p, \min) \hspace{1cm} \triangleright \text{gossips current minimum}
14: \hspace{0.5cm} \text{if } p \text{ has empty queue or local second best interval is better than } p \text{'s best then}
15: \hspace{0.5cm} \quad \text{sendInterval}(p, \text{priorityQ.removeSecond()}) \hspace{1cm} \triangleright \text{gossip-based load balancing step}
16: \hspace{0.5cm} \text{end if}
17: \hspace{0.5cm} \text{end loop}
18: \hspace{0.5cm} \text{procedure } \text{ONRECEIVEINTERVAL}(I(\subseteq D), b) \hspace{1cm} \triangleright D \subseteq \mathbb{R}^d \text{ is the search space, } b \text{ is lower bound of } I
19: \hspace{0.5cm} \quad \text{priorityQ.add}(I, b) \hspace{1cm} \triangleright D \subseteq \mathbb{R}^d \text{ is the search space, } b \text{ is lower bound of } I
20: \hspace{0.5cm} \text{end procedure}
21: \hspace{0.5cm} \text{procedure } \text{ONRECEIVEMIN}(\min_p) \hspace{1cm} \triangleright D \subseteq \mathbb{R}^d \text{ is the search space, } b \text{ is lower bound of } I
22: \hspace{0.5cm} \min \leftarrow \min(\min_p, \min) \hspace{1cm} \triangleright \text{gossip-based load balancing step}
23: \hspace{0.5cm} \text{end procedure}
P2P Evolutionary Algorithms: A Suitable Approach for Tackling Large Instances in Hard Optimization Problems


Euro-Par - 2008
Contribution

- They presented a distributed Evolutionary Algorithm (EA) whose population is structured using a gossiping protocol.
Evolutionary Computation

- A solution to a given optimization problem is called *individual*, and a set of solutions is called *population*.

- Every iteration of the algorithm corresponds to a *generation*, where certain operators are applied to some individuals of the current population to generate the individuals of the population of the next generation.

- At each generation, only some individuals are selected for being elaborated by variation operators, or for being just repeated in the next generation without any change, on the base of their fitness measure.

- Individuals with higher fitness have a higher probability to be selected.

---

**Algorithm 2** Evolutionary Computation (EC)

\[
P = \text{GenerateInitialPopulation}()
\]

\[
\text{while termination condition not met do}
\]

\[
P' = \text{Vary}(P)
\]

\[
\text{Evaluate}(P')
\]

\[
P = \text{Select}(P' \cup P)
\]

\[
\text{end while}
\]
The overall architecture of our approach consists of a population of *Evolvable Agents (EvAg)*.

Each EvAg is a node within a *newscast* topology in which the edges define its neighborhood.

---

**Algorithm 1. Evolvable Agent**

\[
\begin{align*}
S_t & \leftarrow \text{Initialize Agent} \\
\text{loop} & \\
\text{Sols} & \leftarrow \text{Local Selection(Newscast)} \\
S_{t+1} & \leftarrow \text{Recombination(Sols, } P_c \text{)} \\
\text{Evaluate(}S_{t+1}\text{)} & \\
\text{if } S_{t+1} \text{ better than } S_t & \text{ then} \\
S_t & \leftarrow S_{t+1} \\
\text{end if} & \\
\text{end loop} & 
\end{align*}
\]
Algorithm

Algorithm 2. Newscast protocol in node $EvAg_i$

Active Thread

loop
  sleep $\Delta T$
  $EvAg_j \leftarrow$ Random selected node from $Cache_i$
  send $Cache_i$ to $EvAg_j$
  receive $Cache_j$ from $EvAg_j$
  $Cache_i \leftarrow$ Aggregate ($Cache_i, Cache_j$)
end loop

Passive Thread

loop
  wait $Cache_j$ from $EvAg_j$
  send $Cache_i$ to $EvAg_j$
  $Cache_i \leftarrow$ Aggregate ($Cache_i, Cache_j$)
end loop

Local Selection (Newscast)

$[EvAg_j, EvAg_k] \leftarrow$ Random selected nodes from $Cache_i$

- The aggregation consists of picking up the newest item for each cache entry in $Cache_i$, $Cache_j$ and merging them into a single cache.
DONE!
References


Question?