Basic Problem Statement

Minimize \( f(x_1, \ldots, x_n) \)  
Subject to\[
\begin{align*}
g_1(x_1, \ldots, x_n) & \leq 0 \\
& \vdots \\
g_m(x_1, \ldots, x_n) & = 0
\end{align*}
\]

Alternatively, one may want to maximize a function subject to constraints.
Types of Optimization Problems

Minimize \( f(x_1, \ldots, x_n) \)
Subject to
\[ g_i(x_1, \ldots, x_n) \quad \text{op} \quad K_i \]
\[ \ldots \]
\[ g_m(x_1, \ldots, x_n) \quad \text{op} \quad K_m \]

where \( \text{op} \in \{<,\leq,=\} \}

If the objective function and all constraint functions are linear, we have a Linear Programming (LP) problem.

An LP in which the variables can only take integer values is an Integer Programming Problem.

A LP in which some of the variables are restricted to be integers is called a mixed Integer Programming Problem.

If the objective function and/or constraints are non-linear we have a non-linear programming problem (NLP).

Example of a NLP

What is the number \( n \) in a cluster that maximizes the throughput \( X(n) = n \times X \) subject to a) cost \( C(n) = n \times C_0 + C_1 \leq C_{\text{max}} \) and b) the probability \( P_{\text{all-up}} \) that all servers are up must be at least 95%.

Maximize \( X(n) = n \times X \)
Subject to
\[ n \times C_0 + C_1 - 6,000 \leq 0 \]
\[ P_{\text{all-up}}(n) = p^n \geq 0.75 \]
Excel Solver

Uses the Generalized Reduced Gradient (GRG2) algorithm for optimizing nonlinear problems.

Adding a Constraint
Solution

Throughput of an individual server 10 tps
Co $500.00
C1 $1,000.00
Cmax $6,000.00
Pfailure 0.05
Min-Prob-Up 0.75

Objective function 50 tps
n 5
Cost (n) $3,500.00
Prob-all-up 0.77

Example of an Optimization Problem

Maximize f (x,y)
0 < x ≤ 4
x + 2y > 3

Problem of finding local optimum and not global optimum.
Grid Resource Allocation and SLAs

• A computation requires NC million CPU cycles and has to be completed in at most $T_{\text{max}}$ time units and its computational cost cannot exceed $C_{\text{max}}$ dollars.
• There are three available computing resources with speed $s_i$ in $10^6$ cycles/sec and cost $c_i$ in dollars/sec.
• Possible co-allocations: (1), (2), (3), (1,2), (1,3), (1,3), and (1,2,3)

Grid Resource Allocation and SLAs

- N: number of computing resources
- $NC_i$: number of cycles allocated to computing resource i.
- T: execution time of a given allocation
- C: cost of an allocation.

Constraints:

$$\sum_{i=1}^{N} NC_i = NC$$
$$T = \max_{i=1}^{N} \left\{ \frac{NC_i}{s_i} \right\} \leq T_{\text{max}}$$
$$C = \sum_{i=1}^{N} \frac{NC_i}{s_i} \times c_i \leq C_{\text{max}}$$
Grid Resource Allocation and SLAs

• If there are no cost constraints, the solution that minimizes the total execution time is the one that allocates more cycles to the faster resources in proportion to its speed:

\[ NC_i = NC \times \frac{S_i}{\sum_{j=1}^{N} S_j} \]

Grid Resource Allocation Optimization Problems

• Problem 1 (execution time minimization): “Find the feasible solution that satisfies the cost constraint at minimum execution time.”

• Problem 2 (cost minimization): “Find the feasible solution that minimizes the cost \( C \) and that satisfies the execution time constraint.”
Execution Time Minimization

Inputs:

<table>
<thead>
<tr>
<th>Node</th>
<th>speed (si)</th>
<th>Mcycles/sec</th>
<th>cost/sec</th>
<th>(ci) $/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,000</td>
<td>0.1</td>
<td>10,000</td>
<td>1,000</td>
</tr>
<tr>
<td>2</td>
<td>2,000</td>
<td>0.25</td>
<td>10,000</td>
<td>2,000</td>
</tr>
<tr>
<td>3</td>
<td>3,000</td>
<td>0.6</td>
<td>10,000</td>
<td>3,000</td>
</tr>
</tbody>
</table>

Solution: Obtained using MS Excel's Solver Tool.

<table>
<thead>
<tr>
<th>Allocation</th>
<th>Cycle Allocation (NCi)</th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10,000,000</td>
<td>-</td>
<td>10,000</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>10,000,000</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>10,000,000</td>
<td>3,333</td>
</tr>
<tr>
<td>1,2</td>
<td>3,333,333</td>
<td>6,666,667</td>
<td>0</td>
</tr>
<tr>
<td>1,3</td>
<td>4,800,000</td>
<td>0</td>
<td>5,200,000</td>
</tr>
<tr>
<td>2,3</td>
<td>0</td>
<td>6,666,667</td>
<td>3,333,333</td>
</tr>
<tr>
<td>1,2,3</td>
<td>2,592,592.41</td>
<td>5,185,185.39</td>
<td>2,222,222.20</td>
</tr>
</tbody>
</table>

© 2007 D.A. Menasce. All Rights Reserved.
Cost Minimization

<table>
<thead>
<tr>
<th>Node</th>
<th>speed (Mcycles/sec)</th>
<th>cost/sec</th>
<th>NC (Mcycles)</th>
<th>Tmax ($/sec)</th>
<th>Cmax ($/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,000</td>
<td>0.1</td>
<td>10,000,000</td>
<td>4,800</td>
<td>1,500</td>
</tr>
<tr>
<td>2</td>
<td>2,000</td>
<td>0.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3,000</td>
<td>0.6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Inputs:

Solution: Obtained using MS Excel's Solver Tool.

<table>
<thead>
<tr>
<th>Cycle Allocation (NCi)</th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10,000,000</td>
<td>1,000</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>5,000,000</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>3,333,000</td>
</tr>
<tr>
<td>1.2</td>
<td>4,800,000</td>
<td>5,200,000</td>
</tr>
<tr>
<td>1.3</td>
<td>5,000,000</td>
<td>0</td>
</tr>
<tr>
<td>2.3</td>
<td>4,800,000</td>
<td>9,600,000</td>
</tr>
<tr>
<td>1.2,3</td>
<td>4,800,000</td>
<td>5,200,000</td>
</tr>
</tbody>
</table>

Optimization Example in Class

• A computer system has ND disks on which NF files need to be stored. The capacity of disk i is $C_i$ bytes, the size of file $j$ is $S_j$, the arrival rate of requests to file $j$ is $\lambda_j$, and the average service time of a request to file $j$ when it is stored on disk $i$ is $s_{i,j}$.

• Assume ND = 2. Find the optimal allocation of files to disks that minimizes the difference in utilization between the two disks.
Optimization Example in Class

minimize $|U_1 - U_2|$

subject to

$U_i = \sum_{j=1}^{NF} a_{j,i} \times \lambda_j \times s_{i,j}$ utilization of disk i

$\sum_{j=1}^{NF} a_{j,i} \times S_j \leq C_i$ disk capacity constraint

$0 \leq U_i < 1$ utilization of disk i constraint

$a_{j,i} \in \{0,1\}$ 0 indicates that file j is not stored on disk i.

<table>
<thead>
<tr>
<th>File number</th>
<th>Size (GB)</th>
<th>Lambda (req/sec)</th>
<th>Service time on disk 1 (sec)</th>
<th>Service time on disk 2 (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>20</td>
<td>0.010</td>
<td>0.014</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>30</td>
<td>0.008</td>
<td>0.010</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>15</td>
<td>0.015</td>
<td>0.017</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>9</td>
<td>0.020</td>
<td>0.022</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>10</td>
<td>0.025</td>
<td>0.030</td>
</tr>
</tbody>
</table>

Disk capacity (in GB)

<table>
<thead>
<tr>
<th>Disk capacity (in GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>
Combinatorial Search

- Space of solutions: vectors of the type $\vec{c} = (c_1, \ldots, c_n)$
- A cost function is associated with each vector: $C(\vec{c})$
- Number of possible values of $c_i$: $|c_i|$
- Size of the search space: $\prod |c_i|$
- Search: find point $c^*$ that optimizes $C(\vec{c})$
- If the size of the search space is too large, an exhaustive search is not feasible: need to use a heuristic approach.

Heuristic Combinatorial Search Techniques

- Purpose:
  - Reduce significantly the number of points visited in the solution space
  - Find a point whose cost is very close to the optimal solution.
- Two examples of heuristic search techniques:
  - Hill climbing
  - Beam search
Hill Climbing Search

Set initial point: \( c \leftarrow \text{co}; \)
Initialize iteration counter: \( i \leftarrow 0; \)
\( \text{ThereIsLargestNeighbor} \leftarrow \text{True} \)

\textbf{While} \ (i < \text{MaxIterations} \& \ 
\text{ThereIsLargestNeighbor})

\quad \text{If} \ \text{there is a neighbor} \ c' \ \text{of} \ c \ \text{with higher cost than} \ c \\
\quad \quad \text{then} \ c \leftarrow c' \ \textbf{else} \ \text{ThereIsLargestNeighbor} \leftarrow \text{False}; \\
\quad i \leftarrow i + 1

\textbf{EndWhile}

Solution is point \( c \)
Hill-Climbing Search

© 2007 D.A. Menasce. All Rights Reserved.
(k,d) Beam Search

- $\mathcal{V}(\vec{n})$: set of neighbors of configuration vector $\vec{n}$.
- LevelList$_i$: set of configuration vectors examined at level $i$ of the beam search tree.
- CandidateList: set of all configuration vectors selected as the $k$ best at all levels of the beam search tree.
- Top $(k, \mathcal{L})$: set of configuration vectors with the $k$ highest utility function values from the set $\mathcal{L}$.
- $\vec{n}_0$: current configuration vector.

Levellist$_0 \leftarrow \vec{n}_0$;
CandidateList $\leftarrow$ LevelList$_0$;
For $i = 1$ to $d$ Do

   Begin
      LevelList$_i \leftarrow \emptyset$;
      For each $\vec{n} \in$ LevelList$_{i-1}$ Do
         LevelList$_i \leftarrow$ LevelList$_i \cup \mathcal{V}(\vec{n})$;
         LevelList$_i \leftarrow$ Top $(k, \text{LevelList}_i)$;
         CandidateList $\leftarrow$ CandidateList $\cup$ LevelList$_i$;
      End;
   End;
$\vec{n}_{opt} \leftarrow \max$ (CandidateList)
Applications of Combinatorial Search Techniques

- Autonomic Computing (self-configuring, self-organizing)
  - Search space: space of all possible configurations of software and hardware parameters of a computer system.
  - Associated with each point, a QoS function (using a performance model) computes the QoS for that configuration.
  - Goal: find the configuration that optimizes the QoS of the system.
Example: optimal selection of CPU share in VMs

state

(30,45,25) -> (25,50,25) -> (25,60,15) -> (25,45,30)
(35,30,35) -> (15,60,25) -> (15,50,35) -> (35,15,50)
(35,25,40) -> (15,70,15) -> (15,45,40) -> (30,25,45)

Example: optimal selection of CPU share in VMs

state

utility

(30,45,25) -> (25,50,25) -> (25,60,15) -> (25,45,30)
(35,30,35) -> (15,60,25) -> (15,50,35) -> (35,15,50)
(35,25,40) -> (15,70,15) -> (15,45,40) -> (30,25,45)

© 2007 D.A. Menasce. All Rights Reserved.
Example: optimal selection of CPU share in VMs

state
(30,45,25)
utility
(35,30,35)
(35,25,40)
0.2
0.5
0.6
(25,50,25)
(15,60,25)
(15,70,15)
0.3
0.8
0.9
(25,60,15)
(15,50,35)
(15,45,40)
0.6
0.1
0.95
(25,45,30)
(35,15,50)
(30,25,45)
-0.2
-0.1
0.5
Example: optimal selection of CPU share in VMs

state
(30,45,25)
utility
(35,30,35)
(35,25,40)
0.2
0.5
0.6
(25,50,25)
(15,60,25)
(15,70,15)
0.3
0.8
0.9
(25,60,15)
(15,50,35)
(15,45,40)
0.6
0.1
0.95
(25,45,30)
(35,15,50)
(30,25,45)
-0.2
-0.1
0.5

© 2007 D.A. Menasce. All Rights Reserved.
Example: optimal selection of CPU share in VMs

Example: optimal selection of CPU share in VMs
Example: optimal selection of CPU share in VMs

State: CPU share allocated to each VM:
(a1, ..., aM)

The utility function at each state is computed with the help of a predictive queuing network model. The best state to move to is determined using combinatorial search techniques.