CIV E 711
COMPUTER-AIDED PROJECT ORGANIZATION & MANAGEMENT

Instructor: Dr. Tarek Hegazy
Term: Winter 2005
Office: CPH 2373G
Ext.: 2174
E-mail: tarek@uwaterloo.ca

Description:
Application of computerized tools to develop decision-support systems for effectively managing the time, money, and resources of projects. It covers: introduction to computer tools, review of the CPM method and project management software, optimization using Excel Solver, Expert Systems, Neural Networks, OOP programming, Genetic Algorithms, process modeling and simulation, integrated project management tools, Enterprise resource planning, Asset Management, Internet, dealing with project uncertainty using Monte-Carlo simulation, various case studies and computer workshops.


SUGGESTED REFERENCES:

Evaluation
Assignments 10%
Projects 40%
Final 50%
Contents:
Tentatively, the following subjects will be covered:

<table>
<thead>
<tr>
<th>Week</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>• Excel Programming &amp; Introduction to AI</td>
</tr>
<tr>
<td>2</td>
<td>• Optimization using Excel Solver.</td>
</tr>
<tr>
<td>3</td>
<td>• CPM, EasyPlan &amp; Microsoft Project Software.</td>
</tr>
<tr>
<td>4</td>
<td>• Genetic Algorithms.</td>
</tr>
<tr>
<td>5</td>
<td>• AI &amp; Expert Systems.</td>
</tr>
<tr>
<td>6</td>
<td>• Neural Networks. Computer Implementation.</td>
</tr>
<tr>
<td>7</td>
<td>• Planning of repetitive projects. Computer Implementation.</td>
</tr>
<tr>
<td>8</td>
<td>• Resource Management.</td>
</tr>
<tr>
<td>9</td>
<td>• Asset Management.</td>
</tr>
<tr>
<td>10</td>
<td>• Fuzzy Logic. Computer Applications.</td>
</tr>
<tr>
<td>11</td>
<td>• Project control techniques and earned-value. Enterprise resource planning</td>
</tr>
<tr>
<td>12</td>
<td>• Dealing with uncertainty; Monte-Carlo simulation.</td>
</tr>
<tr>
<td>13</td>
<td>• Class presentations.</td>
</tr>
</tbody>
</table>

PROJECTS
- Project Case Study
- Suggested Research Topics
  - New Evolutionary Algorithms (e.g., Honey Bee)
  - New Neural Networks
  - New AI Tools
  - Life-Cycle cost optimization for infrastructure assets
  - Facility Condition Assessment
  - Health-Monitoring of Assets
  - Planning of Hi-rise construction
  - Claim analysis on construction projects
  - Dynamic project control
  - Internet-based applications
  - Modelling and simulation of construction operations.
  - Advanced project control
  - Integrating simulation, GIS, and Genetic Algorithms for resource optimization

Reference Sources:
- Books on Project Management and Construction Management;
- Trade magazines (e.g., ENR);
- International journals: Construction Eng. and Management (ASCE), Computing in Civil Eng. (ASCE), Infrastructure Systems (ASCE), Constructed Facilities (ASCE), CA Civil & Infrastructure Eng., Automation in construction, Constr. Mgmt & Economics; TRB, Transportation Research Board
- Databases such as "current contents", “compendex”, & “CISTI”;
- Organizations such as Project Management Institute (PMI) & American Association of Cost Engineers (AACE);
- Internet search & Web sites;
- European Journal of Operations Research; and
- Government publications such as statistics Canada, etc.
Setup of Your Account

1. □ Activate Excel
2. □ Change macro security level to low (Tools – Macro – Security)
3. □ Unselect having Excel help to appear on Excel start
4. □ Close Excel
5. □ Activate Microsoft Project
6. □ Change macro security level to low (Tools – Macro – Security)
7. □ Unselect having the help to appear on Microsoft Project start
8. □ Close Microsoft Project
9. □ Download workshop.exe and expand it to your N drive.

Computer Lab. On Excel

1. Log into your NEXUS account
2. Activate Excel
3. In a new sheet, construct the following data table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Project Code</td>
<td>Type</td>
<td>Area</td>
<td>Budget</td>
<td>Actual Cost</td>
</tr>
<tr>
<td>2</td>
<td>Pr1</td>
<td>School</td>
<td>Central</td>
<td>1</td>
<td>0.85</td>
</tr>
<tr>
<td>3</td>
<td>Pr2</td>
<td>Bridge</td>
<td>East</td>
<td>7</td>
<td>7.25</td>
</tr>
<tr>
<td>4</td>
<td>Pr3</td>
<td>Road</td>
<td>West</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Pr4</td>
<td>School</td>
<td>West</td>
<td>1.2</td>
<td>1.46</td>
</tr>
<tr>
<td>6</td>
<td>Pr5</td>
<td>Road</td>
<td>East</td>
<td>2.5</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>Pr6</td>
<td>School</td>
<td>Central</td>
<td>1.75</td>
<td>1.8</td>
</tr>
<tr>
<td>8</td>
<td>Pr7</td>
<td>Bridge</td>
<td>Central</td>
<td>4</td>
<td>3.7</td>
</tr>
<tr>
<td>9</td>
<td>Pr8</td>
<td>Bridge</td>
<td>West</td>
<td>2.7</td>
<td>2.4</td>
</tr>
<tr>
<td>10</td>
<td>Pr9</td>
<td>Road</td>
<td>East</td>
<td>11</td>
<td>11.5</td>
</tr>
<tr>
<td>11</td>
<td>Pr10</td>
<td>School</td>
<td>Central</td>
<td>0.75</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Notice how Excel organizes data in the form of a list (or Table). No empty columns or rows.

Experiments

1. Write a simple equation to calculate the profit for project Pr1 in cell (F2), as a percentage.
2. Copy the previous equation to calculate the profit for all projects.
3. Sort the data in the table to show the maximum profits first.
4. Add another sort to show the maximum profit under each type of project.
5. Use auto filter to show only the projects in central area.
6. Use auto filter to show only positive profits in all areas and all types of projects.
7. Add more projects using the Data - Form menu option.
8. Write a (VLOOKUP) function to retrieve the profit of any project as a function of project code.
9. Construct a Pivot Table to show a report in which you can select any project type and any area. The report directly shows the budget, actual cost, and profit of associated projects.
10. Use the previous pivot table to show the total profit for Bridges in the West area.
11. Download file Chapter2.xls from the course web site and experiment with it.
Log into your NEXUS account.

Activate Excel

Make sure Excel Solver is on (Tools, Add-Ins) and Macros enabled (Tools, Macro, Security)

a) Solving problems with one or more variables using “GOAL-SEEK”:

- Look at the example. Do it on Excel.
- Write the needed equation in cell B8 to calculate the total cost as a function of the quantity.
- Activate Goal-Seek from the Tools menu and Experiment with different goal values.

b) Using “SOLVER” to solve a multivariable optimization problem:

Optimization tries to Maximize or Minimize an Objective Function by determining the Optimum values for a set of Variables so that

A set of Constraints are met

The example involves the shipment of aggregates from three quarries to five projects. The aggregates can be shipped from any plant to any project, but it costs more to ship over long distances than over short distances. The problem is to determine the amounts to ship from each quarry to each project so that the total shipping cost is minimized, while the supply limits of each quarry are not exceeded and the demand amounts at each project are satisfied.

Data:
- The three quarries A, B, & C can produce a maximum of 310, 260, & 280 truck loads per day, respectively.
- The five projects PR1, PR2, PR3, PR4, & PR5 have daily demands of 180, 80, 200, 160, & 220 truck loads, respectively.
- The shipping costs ($) are as follows:

<table>
<thead>
<tr>
<th></th>
<th>PR1</th>
<th>PR2</th>
<th>PR3</th>
<th>PR4</th>
<th>PR5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>

Solution:
- Always represent each variable in a separate cell and give it an initial quantity of 1.0.
- Always represent each data element in a separate cell. Use another adjacent cell to type labels.
- Write the worksheet formulas to do intermediate calculations.
- Have a single separate cell to represent the objective function.
- Activate Solver from the Tools menu, input the data, and solve it. Experiment with Solver.
- Read the HELP on the mathematical basis of SOLVER.
Planning: Part 1 – Network Diagrams

Identifying Activity Logical Relationships

Jigsaw puzzle - Brainstorming
Which activities are parallel?
Which activities must precede?
Which activities must succeed

Drawing the Project Network Diagram

Benefits of AON

Does not need dummy activities.
The sequence step calculation also made the AON to look more organized and clearer to read.
The technique is also well suited to computer implementation.
Has a major advantage in terms of the types of logical relationships it allows
(Finish-to-Start, Start-to-Start, Start-to-Finish, and Finish-to-Finish).

Exercise:
The following table gives the work items of a certain contract together with their estimated quantities and total direct cost. Total of indirect cost and markup is $140,000.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Quantity</th>
<th>Unit</th>
<th>Direct Cost</th>
<th>Indirect Cost</th>
<th>Bid Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common excavation</td>
<td>500,000</td>
<td>m3</td>
<td>$475,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rock excavation</td>
<td>200,000</td>
<td>m3</td>
<td>$2,400,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structures</td>
<td>---</td>
<td>LS</td>
<td>$400,000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Detailed Estimating:**

Resources? Durations? Costs?

Cost References? e.g., Means

Allowance for job conditions? Setup Time? Weather?

**An Example**

If the **daily production** rate of the crew (CR-06) that works in a certain activity is **175 units/day** (e.g., ft\(^2\)/day) and the total crew cost per day is $1,800. The material needed for daily work is 4.5 units of M1 ($100/unit).

a) Calculate the time and cost it takes the crew to finish 1,400 units; and

b) Calculate the total unit cost. Consider an 8 hour work day.

\[
Duration = \frac{Q}{P \cdot F} ; \quad Cost = D \cdot $/day
\]
Scheduling
- Calculate ES, LF, & TF for all activities. What are the critical ones?
- Draw an Early Bar Chart for the project.
- What is the effect of delaying activity H by two days on the total project duration?
Using Bar Chart to Accumulate Resources and Show Planned versus Actual Schedule

Profile of the labor resource demand

Activity Float?
Critical Activities?
Critical Path?

Early Bar Chart

Late Bar Chart
Using Microsoft Project

Browse the various features of Microsoft Project.

Let’s now use Microsoft Project and try to see if it has all the features we need to plan our project.

Setup

Once a new file is open, use the "Tools-Options" menu item to start setting up the Microsoft Project Software.

Setup Default Options

With the "Schedule" tab, adjust default options as shown. Important ones are:
- Scheduling from start date;
- Duration entered in days
- Default task type is "Fixed Duration"

Click the Set as Default" button, then "OK".

Setup Working Times

Use the right mouse button on the calendar and select "Change Working Time". Then, as shown, select the Saturday and Sunday columns and specify them as "Working Time". This gives us a 7-day working week. You may also specify any day as off or change the work hours on any day. Then, click "OK".
Setup Time Scale

Use the right mouse button on the calendar and select "Time Scale". Set the major scale units as months labeled as shown. Also, set the Minor scale units as days labeled as shown.

<table>
<thead>
<tr>
<th>Timescale</th>
<th>Nonworking Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major scale</td>
<td></td>
</tr>
<tr>
<td>Units: Months</td>
<td>Label: 1, 2, 3, 4, … (From Start)</td>
</tr>
<tr>
<td>Count: 1</td>
<td>Align: Left</td>
</tr>
<tr>
<td>Minor scale</td>
<td></td>
</tr>
<tr>
<td>Units: Days</td>
<td>Label: 1, 2, 3, 4, … (From Start)</td>
</tr>
<tr>
<td>Count: 1</td>
<td>Align: Center</td>
</tr>
<tr>
<td>General</td>
<td>Scale separator</td>
</tr>
</tbody>
</table>

Setup the Layout

Use the "Format-Layout" menu option to select how the bar chart will look like.

Input Project Activities

To input the activities of the project, let's enter their names one-by-one in the Sheet with their durations in the two columns shown.

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Excavation</td>
<td>2 days</td>
</tr>
<tr>
<td>2 Foundation</td>
<td>2 days</td>
</tr>
<tr>
<td>3 Building Wall</td>
<td>1 day</td>
</tr>
<tr>
<td>4 House Walls</td>
<td>4 days</td>
</tr>
<tr>
<td>5 House Roof</td>
<td>3 days</td>
</tr>
<tr>
<td>6 Select Finishes</td>
<td>1 day</td>
</tr>
<tr>
<td>7 Interior Finishing</td>
<td>3 days</td>
</tr>
<tr>
<td>8 Clean Up</td>
<td>1 day</td>
</tr>
<tr>
<td>9 Finish Garage Doors</td>
<td>6 days</td>
</tr>
<tr>
<td>10 Garage Walls</td>
<td>3 days</td>
</tr>
<tr>
<td>11 Garage Roof</td>
<td>2 days</td>
</tr>
<tr>
<td>12 Garage Doors</td>
<td>2 days</td>
</tr>
</tbody>
</table>

Specifying Relationships

There are several ways to specify the relationships among the tasks.

Move the divider bar until you see the "Predecessors" column. Then type the row numbers of the predecessors separated by commas and hit the ENTER key. A relationship will be inserted (arrow) and task 2 is made to follow task 1, as shown. If you double click the mouse on the relationship arrow, a window for specifying the relationship type and lag time appears. Another way is to drag from the middle of a task into another task, and a relationship will be inserted and predecessor ID is written into the "Predecessors" column.
The Schedule

Once relationships are entered and chart is formatted using "Format-GanttChartWizard", the following schedule of 16 days will result.

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
<th>Predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Foundation</td>
<td>2 days</td>
<td>Fri 11/20/04</td>
<td>Sat 11/27/04</td>
<td></td>
</tr>
<tr>
<td>2 Wall</td>
<td>2 days</td>
<td>Sun 11/20/04</td>
<td>Mon 11/20/04</td>
<td>1</td>
</tr>
<tr>
<td>3 Fixing Wall</td>
<td>1 day</td>
<td>Tue 11/22/04</td>
<td>Tue 11/29/04</td>
<td>2</td>
</tr>
<tr>
<td>4 House Walls</td>
<td>4 days</td>
<td>Wed 12/2/04</td>
<td>Sat 12/9/04</td>
<td>3</td>
</tr>
<tr>
<td>5 House Roof</td>
<td>3 days</td>
<td>Sun 12/20/04</td>
<td>Tue 12/27/04</td>
<td>4</td>
</tr>
<tr>
<td>6 Select Finishes</td>
<td>1 day</td>
<td>Fri 11/29/04 &amp; Sun 12/5/04</td>
<td>Fri 11/29/04, Sat 12/10/04</td>
<td>7,11</td>
</tr>
<tr>
<td>7 Interior Finishes</td>
<td>3 days</td>
<td>Wed 12/29/04</td>
<td>Fri 12/31/04</td>
<td>6.5</td>
</tr>
<tr>
<td>8 Clean Up</td>
<td>1 day</td>
<td>Sat 12/31/04</td>
<td>Thu 1/5/05</td>
<td>7.12</td>
</tr>
<tr>
<td>9 Fix Garage Doors</td>
<td>6 days</td>
<td>Fri 11/29/04</td>
<td>Wed 12/6/04</td>
<td>10</td>
</tr>
<tr>
<td>10 Garage Doors</td>
<td>3 days</td>
<td>Wed 12/29/04</td>
<td>Fri 12/31/04</td>
<td>3</td>
</tr>
<tr>
<td>11 Garage Roof</td>
<td>2 days</td>
<td>Sat 12/24/04</td>
<td>Sun 1/5/05</td>
<td>10</td>
</tr>
<tr>
<td>12 Garage Doors</td>
<td>2 days</td>
<td>Mon 12/28/04</td>
<td>Tue 1/5/05</td>
<td>9.11</td>
</tr>
</tbody>
</table>

It is now possible to view many of the software's preset tables. Use the "View-Table-Schedule" menu option to show all schedule data, as shown here.

Viewing the Project Network

Now, you may view the project network. Notice that critical activities have bold borders. To specify what data to view in the box of each task use "Format-Box Styles" menu option. Experiment with this option.

Specifying Resources

Now, let's view the resource sheet and specify the resource categories and maximum available amount. Specify our (2 of L5).

Assign Resources to Tasks

From the Gantt chart, select each toolbar button shown and type the units as shown, then hit the "Assign" button and continue to next activity, and so on. Once finished, you will notice that project duration is still 16 days.

Comments on Microsoft Project:
Costs?
Deadline? Penalty? Incentive?
Productivity Factors?
Optimization? Actual Progress?
Resource Leveling vs Resource Allocation

**Resource Leveling:**
- How to smooth resource demands?
- No problem with time or resources.
- Strategy?
- Method of moments?
- Method of double moments?
- Multi-Resources?
- Desired (best) profiles?

**Example:**
Two schedule alternatives have associated resource profiles as shown below, which alternative would you choose and why? Also calculate the total **worker-weeks** needed for both cases:

\[
\begin{array}{c|c}
R & Start Delay \\
2 & A \\
1 & B \\
1 & C \\
2 & D \\
\end{array}
\]

\[
\begin{array}{c|c}
R & Start Delay \\
2 & E \\
\end{array}
\]

**Resource Profile:**

Mx =

My =
Resource Allocation:

- Allocate limited resources to top-priority activities.
- Strategy?
- Heuristic rules
- Inconsistency among existing software
- Excel Implementation
- EasyPlan Optimization

\[ \text{Resource limit} = 2 / \text{day} \]

Priority Rules:
Resource Allocation Example:

- Resource Limit = 2 /day
- Each activity requires One resource / day

<table>
<thead>
<tr>
<th>Time</th>
<th>Eligible Activities</th>
<th>Resources (limit = 2)</th>
<th>Duration</th>
<th>Rule (ELS)</th>
<th>Decision</th>
<th>Finish Time</th>
</tr>
</thead>
</table>

Microsoft Project Solution?
Case of Multi-Resources? Case of Multi-Resources Multi-Skills?

- **Meeting Deadline**

(Using faster more expensive methods of construction)

Strategy?
**Cash-Flow Planning**

**How much is the total monthly cost?**

**How to arrange for financing?**

**How to improve cash flow:**
- Credit from suppliers;
- Subcontracting;
- Mobilization payment;
- Better scheduling

*S-Curve*: Calculated based on the Bar Chart.

**Cumulative % of work completed**

**Time %**

- 15
- 50
- 85
- 100

**Cost ($)**

**Expenses Profile**

**Income Profile**

**Month**

**Cash out-of-flow**

**Mobilization Payment**

**Retainage**

**Total monthly cost?**
<table>
<thead>
<tr>
<th>Objective</th>
<th>Allocation</th>
<th>TCT</th>
<th>Leveling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resources are limited</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project can be delayed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activities can be delayed beyond their floats</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Critical activities are affected</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-Critical activities are affected</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Objective is to reduce delay</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimize total (direct + indirect) project cost</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reduce resource fluctuation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Exercise on dealing with Limited Resources:**

<table>
<thead>
<tr>
<th>Objective</th>
<th>Exercise:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demo Excel file to show how to deal with limited resources.</td>
<td>□ Load the file <strong>Sch-Level.xls</strong>. Go to Sheet “CPM” and experiment with the “Delay” column. Its purpose is __________________________. Then, go to sheet “CPM-BarChart” and look at how the simple bar chart looks on Excel. Then go to sheet “CPM-Res” and look at how resources are input and accumulated below the bar chart. Project duration: _____ ; Max resource amount used: ___; After deleting all delays, Critical activities are: _______________ Project duration: _____ ; Max resource amount used: ___; Now, try manually to change delay values to resolve the problem. Setup solver for the problem and try to minimize project delay under limited resources. Does Solver work?</td>
</tr>
<tr>
<td>Note: Changes in activity start using a delay value, affects resource profiles.</td>
<td>□ Load the file <strong>Sch-Level.xls</strong>. Go to Sheet “CPM” and experiment with the “Delay” column. Its purpose is __________________________. Then, go to sheet “CPM-BarChart” and look at how the simple bar chart looks on Excel. Then go to sheet “CPM-Res” and look at how resources are input and accumulated below the bar chart. Project duration: _____ ; Max resource amount used: ___; After deleting all delays, Critical activities are: _______________ Project duration: _____ ; Max resource amount used: ___; Now, try manually to change delay values to resolve the problem. Setup solver for the problem and try to minimize project delay under limited resources. Does Solver work?</td>
</tr>
<tr>
<td>Demo MS Project file to see how commercial software deal with resource limits.</td>
<td>□ Load a file, Use view-resource sheet and view-resource graph. Then, go to Gantt chart and increase the width of sheet to view the resources assigned to the activities. Now, do the following: Clear resource leveling; duration is: _____ ; Res. problem ___(Y/N); Use leveling within float; duration is: _____ ; Res. problem ___(Y/N); Use leveling without limit; duration : _____; Res. problem ___(Y/N).</td>
</tr>
</tbody>
</table>

**Exercise on Crashing Project Duration:**

<table>
<thead>
<tr>
<th>Objective</th>
<th>Exercise:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Excel file to try meet project duration by selecting shorter construction method.</td>
<td>□ Load the file <strong>TCT.xls</strong>. Go to Sheet “CPM-TCT” and experiment with the “Selected” column. Its purpose is __________________________. Look at how the simple bar chart looks. Clear the “selected” column and try to meet a 110-day deadline.</td>
</tr>
</tbody>
</table>

Exercise on EasyPlan
Types of Optimization Problems:

- Linear Problems
- Non-linear Problems
- Combinatorial problems

Linear Problems

In linear problems, all the outputs are simple linear functions of the inputs, as in $y=mx+b$. When problems only use simple arithmetic operations such as addition, subtraction, and Excel functions such as TREND() and FORECAST() it indicates there are purely linear relationships between the variables.

Linear problems have been fairly easy to solve since the advent of computers and the invention by George Dantzig of the Simplex Method. A simple linear problem can be solved most quickly and accurately with a linear programming utility. The Solver utility included with Excel becomes a linear programming tool when you set the "Assume Linear Model" checkbox. Solver then uses a linear programming routine to quickly find the perfect solution. If your problem can be expressed in purely linear terms, you should use linear programming. Unfortunately, most real-world problems cannot be described linearly.

Evolutionary Systems?

Compromise between Local versus Global search strategies.

Non-linear Problems

If the cost to manufacture and ship out 5,000 widgets was $5,000, would it cost $1 to manufacture and ship 1 widget? Probably not. The assembly line in the widget factory would still consume energy, the paperwork would still need to be filled out and processed through the various departments, the materials would still be bought in bulk, the trucks would require the same amount of gas to deliver the widgets, and the truck driver would still get paid a full day's salary no matter how full the load was. Most real-world problems do not involve simple linear relationships. Non-linear problems involve multiplication, division, exponents, and built in Excel functions such as SORT() and GROWTH().

Whenever the variables share a disproportional relationship to one another, the problem becomes non-linear.

If we simply need to find the minimum level of reactants that will give us the highest rate of reaction, we can just start anywhere on the graph and climb along the curve until we reach the top. This method of finding an answer is called hill climbing. Hill climbing will always find the best answer if a) the function being explored is smooth, and b) the initial variable values place you on the side of the highest mountain. If either condition is not met, hill climbing can end up in a local solution, rather than the global solution.

Highly non-linear problems, the kind often seen in practice, have many possible solutions across a complicated landscape. If a problem has many variables, and/or if the formulas involved are very noisy or curvy, the best answer will probably not be found with hill climbing, even after trying hundreds of times with different starting points. Most likely, a sub-optimal, and extremely local solution will be found (see figure below).

Combinatorial problems

There is a large class of problems that are very different from the numerical problems examined so far. Problems where the outputs involve changing the order of existing input variables, or grouping subsets of the inputs are called combinatorial problems. These problems are usually very hard to solve, because they often require exponential time; that is, the amount of time needed to solve a problem with 4 variables might be $4 \times 3 \times 2 \times 1$, and doubling the number of variables to 8 raises the solving time to $8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$, or a factor of 1,680. The number of variables doubles, but the number of possible solutions that must be checked increases 1,680 times. For example, choosing the starting lineup for a baseball team is a combinatorial problem. For 9 players, you can choose one out of the 9 as the first batter. You can then choose one out of the remaining 8 as the second batter, one of the remaining 7 will be the third, and so on. There are thus $9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$ (9 factorial) ways to choose a lineup of 9 players. This is about 362,880 different orderings. Now if you double the number of players, there are 18 factorial possible lineups, or 6,402,373,705,728,000 possible lineups!
Example on Genetic Algorithms

**Problem:** A square construction site is divided into 9 grid units. We need to use GAs to determine the best location of two temporary facilities A and B, so that:

- Facility A is as close as possible to facility B.
- Facility A is as close as possible to the fixed facility F.
- Facility B is as far as possible to the fixed facility F.

**Step 1:** Problem Representation (how to define a facility location)

**Option 1**
Using coordinates

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|}
 & A & B & F \\
\hline
A & & & \ \ \\
B & & & \ \ \\
\end{array}
\]

X has X = 2 and Y = 1
B has X = 1 and Y = 3

**Option 2**
Using location index

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
A & 2 & 1 & & & & & & \ \\
B & & & & & & & & 7 & \ \\
F & & & & & & & & & \ \\
\end{array}
\]

A is in Location index 2
B is in Location index 7

**Step 2:** Chromosome Structure
The variables in our problem are the locations of facilities A & B. Then, the chromosome structure for each of the two options in Step 1 are as follows. Note that the genes of a chromosome are the variables.

**Option 1**

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|}
X & Y & XB & YB \\
\hline
A & A & & & & & & & \ \\
\end{array}
\]

4 Genes
(Values range from 1 to 3)

**Option 2**

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|}
Inde & Inde \\
\hline
x & A & x & B \\
\hline
2 & 7 & & & & & & & \ \\
\end{array}
\]

2 Genes
(Values range from 1 to 8)

**Step 3:** Generate Population (50 to 100 is reasonable diversity & processing time)
(note: for this exercise, let’s consider option 1 representation and a population of 3)

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|}
P1 & X & Y & XB & YB \\
\hline
A & A & & & & & & & \ \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|}
P2 & X & Y & XB & YB \\
\hline
A & A & & & & & & & \ \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|}
P3 & X & Y & XB & YB \\
\hline
A & A & & & & & & & \ \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|}
A & B & \ \\
\hline
B & F & \ \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|}
A & B & \ \\
\hline
B & F & \ \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|}
A & B & \ \\
\hline
B & F & \ \\
\end{array}
\]
**Step 4: Evaluate the Population**

<table>
<thead>
<tr>
<th>P1</th>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>A</td>
<td>B</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>B</td>
<td>A</td>
<td>F</td>
</tr>
</tbody>
</table>

Objective function = Minimize site score = Minimum of $\sum d \cdot W$

Score = $d_{AB} \cdot W_{AB} + d_{AF} \cdot W_{AF} + d_{BF} \cdot W_{BF}$

Let’s consider the closeness weights $(W)$ as follows (from past notes):

- $W_{AB} = 100$ (positive means A & B close to each other)
- $W_{AF} = 100$ (A & F close to each other)
- $W_{BF} = -100$ (negative means B & F far from each other)

Let’s also consider the distance $(d)$ between two facilities as the number of horizontal and vertical blocks between them.

- **P1** Score = $d_{AB} \cdot W_{AB} + d_{AF} \cdot W_{AF} + d_{BF} \cdot W_{BF} = 3 \cdot 100 + 4 \cdot 100 + 1 \cdot -100 = 600$
- **P2** Score = $d_{AB} \cdot W_{AB} + d_{AF} \cdot W_{AF} + d_{BF} \cdot W_{BF} = 3 \cdot 100 + 3 \cdot 100 + 2 \cdot -100 = 400$
- **P3** Score = $d_{AB} \cdot W_{AB} + d_{AF} \cdot W_{AF} + d_{BF} \cdot W_{BF} = 2 \cdot 100 + 2 \cdot 100 + 2 \cdot -100 = 200$

**Step 5: Calculate the Merits of Population Members**

- Merit of **P1** = $(600+400+200) / 600 = 2$
- Merit of **P2** = $(600+400+200) / 400 = 3$
- Merit of **P1** = $(600+400+200) / 200 = 6$ Notice the sum of merits = 11

Notice that smaller score gives higher merit because we are interested in minimization. In case of maximization, we use the inverse of the merit calculation.

**Step 6: Calculate the Relative Merits of Population Members**

- RM of **P1** = merit * 100 / Sum of merits = $2 \cdot 100 / 11 = 18$
- RM of **P2** = merit * 100 / Sum of merits = $3 \cdot 100 / 11 = 27$
- RM of **P3** = merit * 100 / Sum of merits = $6 \cdot 100 / 11 = 55$

**Step 7: Randomly Select Operator (Crossover or Mutation)**

Crossover rate = 96% (marriage is the main avenue for evolution)
Mutation rate = 4% (genius people are very rare)

To select which operator to use in current cycle, we generate a random number (from 0 to 100). If the value is between 0 to 96, then crossover, otherwise, mutation.

**Step 8: Use the Selected Operator (Assume Crossover)**
8.a) Randomly select two parents according to their relative merits of Step 6

For first parent, we generate a random number (0 to 100). According to its value, we pick the parent. For example, assume value is 76, then P3 is selected.

For the 2nd parent, get a random number (0 to 100). Assume 39, Then P2 is picked.

8.b) Let’s apply crossover to generate an offspring

For the crossover range, we get 2 random numbers, say 2 and 3

Step 9: Evaluate the Offspring

Notice that Offspring 2 is invalid because both facilities A & B are at same coordinates (x = 2 and Y = 3) and this is not allowed

Offspring Score = \( d_{AB} \cdot W_{AB} + d_{AF} \cdot W_{AF} + d_{BF} \cdot W_{BF} = 1 \cdot 100 + 4 \cdot 100 + 3 \cdot -100 = 200 \)

Step 10: Compare the Offspring with the Population (Evolve the Population)

Since the offspring score = 200 is better than the worst population member (P1 has a score of 600), then the offspring survives and P1 dies (will be replaced by the offspring).

Accordingly, P1 becomes: 1 1 1 2

At the end of this step, we GOTO STEP 4, repeating the process thousands of times until the best solution is determined. One of the top solutions is as follows:

Score = 0
Comparison among five evolutionary-based optimization algorithms
by
Emad Elbeltagi; Tarek Hegazy; and Donald Grierson

ABSTRACT: Evolutionary algorithms are stochastic search methods that mimic the natural biological evolution and/or the social behavior of species. Such algorithms have been developed to arrive at near-optimum solutions to large-scale optimization problems, for which traditional mathematical techniques may fail. This paper compares the formulation and results of five recent evolutionary-based algorithms: genetic algorithms, memetic algorithms, particle swarm, ant colony systems, and shuffled frog leaping. A brief description of each algorithm is presented along with a pseudocode to facilitate the implementation and use of such algorithms by researchers and practitioners. Benchmark comparisons among the algorithms are presented for both continuous and discrete optimization problems, in terms of processing time, convergence speed, and quality of the results. Based on this comparative analysis, the performance of evolutionary algorithms is discussed along with some guidelines for determining the best operators for each algorithm. The study presents sophisticated ideas in a simplified form that should be beneficial to both practitioners and researchers involved in solving optimization problems.

1. Introduction
The difficulties associated with using mathematical optimization on large scale engineering problems have contributed to the development of alternative solutions. Linear programming and dynamic programming techniques, for example, often fail (or reach local optimum) in solving NP-hard problems with large number of variables and non-linear objective functions [1]. To overcome these problems, researchers have proposed evolutionary-based algorithms for searching near-optimum solutions to problems.

Evolutionary Algorithms are stochastic search methods that mimic the metaphor of natural biological evolution and/or the social behavior of species. Examples include how ants find the shortest route to a source of food and how birds find their destination during migration. The behaviour of such species is guided by learning, adaptation, and evolution [1]. To mimic the efficient behaviour of these species, various researchers have developed computational systems that seek fast and robust solutions to complex optimization problems. The first evolutionary-based technique introduced in the literature was the Genetic Algorithms, [2]. Genetic Algorithms (GAs) were developed based on the Darwinian principle of the “survival of the fittest” and the natural process of evolution through reproduction. Based on its demonstrated ability to reach near-optimum solutions to large problems, the GAs technique has been used in many applications in science and engineering [e.g.,3,4,5]. Despite their benefits, GAs may require long processing time for a near-optimum solution to evolve. Also, not all problems lend themselves well to a solution with GAs [6].

In an attempt to reduce processing time and improve the quality of solutions, particularly to avoid being trapped in local optima, other Evolutionary Algorithms (EAs) have been introduced during the past 10 years. In addition to various GA improvements, recent developments in EAs include four other techniques inspired by different natural processes: memetic algorithms [7], particle swarm optimization [8], ant colony systems [9], and shuffled frog leaping [10]. A schematic diagram of the natural processes that the five algorithms mimic is shown in Fig. 1.

In this paper, the five EAs presented in Fig. 1 are reviewed and a pseudocode for each algorithm is presented to facilitate its implementation. Performance comparison among the five algorithms is then presented. Guidelines are then presented for determining the proper parameters to use with each algorithm.

2. Five evolutionary algorithms
In general, EAs share a common approach for their application to a given problem. The problem first requires some representation to suit each method. Then, the evolutionary search algorithm is applied iteratively to arrive at a near-optimum solution. A brief description of the five algorithms is presented in the following subsections.

2.1. Genetic algorithms
Genetic algorithms (GAs) are inspired by biological systems’ improved fitness through evolution [2]. A solution to a given problem is represented in the form of a string, called “chromosome”, consisting of a set of elements, called “genes”, that hold a set of values for the optimization variables [11].
GAs work with a random population of solutions (chromosomes). The fitness of each chromosome is determined by evaluating it against an objective function. To simulate the natural “survival of the fittest” process, best chromosomes exchange information (through crossover or mutation) to produce offspring chromosomes. The offspring solutions are then evaluated and used to evolve the population if they provide better solutions than weak population members. Usually, the process is continued for a large number of generations to obtain a best-fit (near-optimum) solution. More details on the mechanism of GAs can be found in Goldberg [11] and Al-Tabtabai and Alex [3].

A pseudocode for the GAs algorithm is shown in Appendix I. Four main parameters affect the performance of GAs: population size, number of generations, crossover rate, and mutation rate. Larger population size (i.e., hundreds of chromosomes) and large number of generations (thousands) increase the likelihood of obtaining a global optimum solution, but substantially increase processing time.

Crossover among parent chromosomes is a common natural process [12] and traditionally is given a rate that ranges from 0.6 to 1.0. In crossover, the exchange of parents’ information produces an offspring, as shown in Fig. 2. As opposed to crossover, mutation is a rare process that resembles a sudden change to an offspring. This can be done by randomly selecting one chromosome from the population and then arbitrarily changing some of its information. The benefit of mutation is that it randomly introduces new genetic material to the evolutionary process, perhaps thereby avoiding stagnation around local minima. A small mutation rate less than 0.1 is usually used [11].

The GA used in this study is steady state (an offspring replaces the worst chromosome only if is better than it) and real coded (the variables are represented in real numbers). The main parameters used in the GA procedure are population size, number of generations, crossover rate and mutation rate.
2.2. Memetic algorithms

Memetic algorithms (MAs) are inspired by Dawkins’ notion of a meme [13]. MAs are similar to GAs but the elements that form a chromosome are called memes, not genes. The unique aspect of the MAs algorithm is that all chromosomes and offsprings are allowed to gain some experience, through a local search, before being involved in the evolutionary process [14]. As such, the term MAs is used to describe GAs that heavily use local search [15]. A pseudocode for a MA procedure is given in Appendix II.

Similar to the GAs, an initial population is created at random. Afterwards, a local search is performed on each population member to improve its experience and thus obtain a population of local optimum solutions. Then, crossover and mutation operators are applied, similar to GAs, to produce offsprings. These offsprings are then subjected to the local search so that local optimality is always maintained.

Merz and Freisleben [14] proposed one approach to perform local search through a pair-wise interchange heuristic (Fig. 3). In this method, the local search neighborhood is defined as the set of all solutions that can be reached from the current solution by swapping two elements (memes) in the chromosome. For a chromosome of length \( n \), the neighborhood size for the local search is:

\[
N = \frac{1}{2} \cdot n \cdot (n - 1)
\]

(1)

The number of swaps and consequently the size of the neighborhood grow quadratically with the chromosome length (problem variables). In order to reduce processing time, Merz and Freisleben [14] suggested stopping the pair-wise interchange after performing the first swap that enhances the objective function of the current chromosome. The local search algorithm, however, can be designed to suit the problem nature. For example, another local search can be conducted by adding or subtracting an incremental value from every gene and testing the chromosome’s performance. The change is kept if the chromosome’s performance improves; otherwise, the change is ignored. A pseudocode of this modified local search is given in Appendix III. As discussed, the parameters involved in MAs are the same four parameters used in GAs: population size, number of generations, crossover rate, and mutation rate in addition to a local search mechanism.

2.3. Particle swarm optimization

Particle swarm optimization (PSO) was developed by Kennedy and Eberhart [8]. The PSO is inspired by the social behavior of a flock of migrating birds trying to reach an unknown destination. In PSO, each solution is a “bird” in the flock and is referred to as a “particle”. A particle is analogous to a chromosome (population member) in GAs. As opposed to GAs, the evolutionary process in the PSO doesn’t create new birds from parent ones. Rather, the birds in the population only evolve their social behavior and accordingly their movement towards a destination [16].
Physically, this mimics a flock of birds that communicate together as they fly. Each bird looks in a specific direction, and then when communicating together, they identify the bird that is in the best location. Accordingly, each bird speeds towards the best bird using a velocity that depends on its current position. Each bird, then, investigates the search space from its new local position, and the process repeats until the flock reaches a desired destination. It is important to note that the process involves both social interaction and intelligence so that birds learn from their own experience (local search) and also from the experience of others around them (global search).

The pseudocode for the PSO is shown in Appendix IV. The process is initialized with a group of random particles (solutions), \( N \). The \( i \)th particle is represented by its position as a point in a \( S \)-dimensional space, where \( S \) is the number of variables. Throughout the process, each particle \( j \) monitors three values: its current position \( (X_i) \); the best position it reached in previous cycles \( (P_i) \); and its flying velocity \( (V_i) \). These three values are represented as follows:

\[
\begin{align*}
    \text{Current position} & \quad X_i = (x_{i1}, x_{i2}, \ldots, x_{iS}) \\
    \text{Best previous position} & \quad P_i = (p_{i1}, p_{i2}, \ldots, p_{iS}) \\
    \text{Flying velocity} & \quad V_i = (v_{i1}, v_{i2}, \ldots, v_{iS})
\end{align*}
\] (2)

In each time interval (cycle), the position \( (P_g) \) of the best particle \( (g) \) is calculated as the best fitness of all particles. Accordingly, each particle updates its velocity \( V_i \) to catch up with the best particle \( g \), as follows [16]:

\[
\text{New velocity } V_i = \omega \cdot \text{current velocity } V_i + c_1 \cdot \text{rand()} \times (P_i - X_i) + c_2 \cdot \text{Rand()} \times (P_g - X_i)
\] (3)

As such, using the new velocity \( V_i \), the particle’s updated position becomes:

\[
\text{New position } X_i = \text{current position } X_i + \text{New velocity } V_i \quad \text{where } V_{\text{max}} \geq V_i \geq -V_{\text{max}}
\] (4)

where \( c_1 \) and \( c_2 \) are two positive constants named learning factors (usually \( c_1 = c_2 = 2 \)); \( \text{rand()} \) and \( \text{Rand()} \) are two random functions in the range \([0, 1]\); \( V_{\text{max}} \) is an upper limit on the maximum change of particle velocity [8], and \( \omega \) is an inertia weight employed as an improvement proposed by Shi and Eberhart [16] to control the impact of the previous history of velocities on the current velocity. The operator \( \omega \) plays the role of balancing the global search and the local search; and was proposed to decrease linearly with time from a value of 1.4 to 0.5 [16]. As such, global search starts with a large weight and then decreases with time to favor local search over global search [17].

It is noted that the second term in Eq. 3 represents "cognition", or the private thinking of the particle when comparing its current position to its own best. The third term in Eq. 3, on the other hand, represents the "social" collaboration among the particles, which compares a particle’s current position to that of the best particle [18]. Also, to control the change of particles’ velocities, upper and lower bounds for velocity change is limited to a user-specified value of \( V_{\text{max}} \). Once the new position of a particle is calculated using Eq. 4, the particle, then, flies towards it [16]. As such, the main parameters used in the PSO technique are: the population size (number of birds); number of generation cycles; the maximum change of a particle velocity \( V_{\text{max}} \); and \( \omega \).

### 2.4. Ant colony optimization

Similar to PSO, Ant colony optimization (ACO) Algorithms evolve not in their genetics but in their social behavior. ACO was developed by Dorigo et al. [9] based on the fact that ants are able to find the shortest route between their nest and a source of food. This is done using pheromone trails, which ants deposit whenever they travel, as a form of indirect communication.

As shown in Fig. 1-d, when ants leave their nest to search for a food source, they randomly rotate around an obstacle, and initially the pheromone deposits will be the same for the right and left directions. When the ants in the shorter direction find a food source, they carry the food and start returning back, following their pheromone trails, and still depositing more pheromone. As indicated in Fig. 1-d, an ant will most likely choose the shortest path when returning back to the nest with food as this path will have the most deposited pheromone. For the same reason, new ants that later starts out from the nest to find food will also choose the shortest path. Over time, this positive feedback (autocatalytic) process prompts all ants to choose the shorter path [19].

Implementing the ACO for a certain problem requires a representation of \( S \) variables for each ant, with each variable \( i \) has a set of \( n_i \) options with their values \( l_{ij} \) and their associated pheromone concentrations \( \tau_{ij} \), where \( i = 1, 2, \ldots, S \) and \( j = 1, 2, \ldots n_i \). As such, an ant is consisted of \( S \) values that describe the path chosen by the ant as
shown in Fig. 4 [20]. A pseudocode for the ACO is shown in Appendix V. Other researchers use a variation of this general algorithm, incorporating a local search to improve the solution [21].

In the ACO, the process starts by generating \( m \) random ants (solutions). An ant \( k \) \((k = 1, 2, \ldots, m)\) represents a solution string, with a selected value for each variable. Each ant is then evaluated according to an objective function. Accordingly, pheromone concentration associated with each possible route (variable value) is changed in a way to reinforce good solutions, as follows [9]:

\[
\tau_{ij}(t) = \rho \tau_{ij}(t-1) + \Delta \tau_{ij} \quad ; \quad t = 1, 2, \ldots, T
\]

(5)

where \( T \) is the number of iterations (generation cycles); \( \tau_{ij}(t) \) is the revised concentration of pheromone associated with option \( l_{ij} \) at iteration \( t \); \( \tau_{ij}(t-1) \) is the concentration of pheromone at the previous iteration \( (t-1) \); \( \Delta \tau_{ij} \) = change in pheromone concentration; and \( \rho \) = pheromone evaporation rate (0 to 1). The reason for allowing pheromone evaporation is to avoid too strong influence of the old pheromone to avoid premature solution stagnation [22]. In Eq. 5, the change in pheromone concentration \( \Delta \tau_{ij} \) is calculated as [9]:

\[
\Delta \tau_{ij} = \sum_{k=1}^{m} \begin{cases} R / \text{fitness}_k & \text{if option } l_{ij} \text{ is chosen by ant } k \\ 0 & \text{otherwise} \end{cases}
\]

(6)

where \( R \) is a constant called the pheromone reward factor; and \( \text{fitness}_k \) is the value of the objective function (solution performance) calculated for ant \( k \). It is noted that the amount of pheromone gets higher as the solution improves. Therefore, for minimization problems, Eq. 6 shows the pheromone change as proportional to the inverse of the fitness. In maximization problems, on the other hand, the fitness value itself can be directly used.

Once the pheromone is updated after an iteration, the next iteration starts by changing the ants’ paths (i.e., associated variable values) in a manner that respects pheromone concentration and also some heuristic preference. As such, an ant \( k \) at iteration \( t \) will change the value for each variable according to the following probability [9]:

\[
P_{ij}(k,t) = \frac{[\tau_{ij}(t)]^\alpha \times [\eta_{ij}]^\beta}{\sum_{l_j}[\tau_{ij}(t)]^\alpha \times [\eta_{ij}]^\beta}
\]

(7)

where \( P_{ij}(k,t) \) = probability that option \( l_{ij} \) is chosen by ant \( k \) for variable \( i \) at iteration \( t \); \( \tau_{ij}(t) \) = pheromone concentration associated with option \( l_{ij} \) at iteration \( t \); \( \eta_{ij} \) = heuristic factor for preferring among available options and is an indicator of how good it is for ant \( k \) to select option \( l_{ij} \) (this heuristic factor is generated by some problem characteristics and its value is fixed for each option \( l_{ij} \)); and \( \alpha \) and \( \beta \) are exponent parameters that control the relative importance of pheromone concentration versus the heuristic factor [20]. Both \( \alpha \) and \( \beta \) can take values greater than zero and can be determined by trial and error. Based on the previous discussion, the main parameters involved in ACO are: number of ants \( m \); number of iterations \( t \); exponents \( \alpha \) and \( \beta \); pheromone evaporation rate \( \rho \); and pheromone reward factor \( R \).

2.5. Shuffled frog leaping algorithm
The shuffled frog leaping (SFL) algorithm, in essence, combines the benefits of the genetic-based memetic algorithms and the social behavior-based particle swarm optimization algorithms. In the SFL, the population consists of a set of frogs (solutions) that is partitioned into subsets referred to as memeplexes. The different memeplexes are considered as different cultures of frogs, each performing a local search. Within each memeplex, the individual frogs hold ideas, that can be influenced by the ideas of other frogs, and evolve through a process of memetic evolution. After a defined number of memetic evolution steps, ideas are passed among memeplexes in a shuffling process [23]. The local search and the shuffling processes continue until defined convergence criteria are satisfied [10].

As described in the pseudocode of Appendix VI, an initial population of "P" frogs is created randomly. For S-dimensional problems (S variables), a frog i is represented as \( X_i = (x_{i1}, x_{i2}, \ldots, x_{iS}) \). Afterwards, the frogs are sorted in a descending order according to their fitness. Then, the entire population is divided into \( m \) memeplexes, each containing \( n \) frogs (i.e., \( P = m \times n \)). In this process, the first frog goes to the first memeplex, the second frog goes to the second memeplex, frog m goes to the \( m^{th} \) memeplex, and frog \( m+1 \) goes back to the first memeplex, etc.

Within each memeplex, the frogs with the best and the worst fitnesses are identified as \( X_b \) and \( X_w \), respectively. Also, the frog with the global best fitness is identified as \( X_g \). Then, a process similar to PSO is applied to improve only the frog with the worst fitness (not all frogs) in each cycle. Accordingly, the position of the frog with the worst fitness is adjusted as follows:

\[
\text{Change in frog position (D_i)} = \text{rand()} \times (X_b - X_w)
\]

\[
\text{New position } X_{w_{new}} = \text{current position } X_w + D_i; \quad D_{max} \geq D_i \geq -D_{max}
\]

where \( \text{rand()} \) is a random number between 0 and 1; and \( D_{max} \) is the maximum allowed change in a frog’s position. If this process produces a better solution, it replaces the worst frog. Otherwise, the calculations in Eqs. 8 and 9 are repeated but with respect to the global best frog (i.e., \( X_g \) replaces \( X_b \)). If no improvement becomes possible in this case, then a new solution is randomly generated to replace that frog. The calculations then continue for a specific number of iterations [10]. Accordingly, the main parameters of SFL are: number of frogs \( P \); number of memeplexes; number of generation for each memeplex before shuffling; number of shuffling iterations; and maximum step size.

3. Comparison among evolutionary algorithms’ results

All the EAs described earlier have been coded using the Visual Basic programming language and all experiments took place on a 1.8 GHz AMD Laptop machine. The performance of the five evolutionary algorithms is compared using two benchmark problems for continuous optimization and a third problem for discrete optimization. A description of these test problems is given in the following.

3.1. Continuous optimization

Two well-known continuous optimization problems are used to test four of the EAs: \( F8 \) (Griewank’s) function and the \( F10 \) function. Details of these functions are as follows:

**\( F8 \) (Griewank’s function):** The objective function to be optimized is a scalable, non-linear, and non-separable function that may take any number of variables \( x_i \)’s, i.e.,

\[
f(x_{i=1,N}) = 1 + \sum_{i=1}^{N} \frac{x_i^2}{4000} - \prod_{i=1}^{N} (\cos(x_i / \sqrt{i}))
\]

The summation term of the \( F8 \) function (Eq. 10) includes a parabolic shape while the cosine function in the product term creates waves over the parabolic surface. These waves create local optima over the solution space [24]. The \( F8 \) function can be scaled to any number of variables \( N \). The values of each variable are constrained to a range (-512 to 511). The global optimum (minimum) solution for this function is known to be zero when all \( N \) variables equal zero.

**\( F10 \) Function:** This function is non-linear, non-separable, and involves two variables \( x \) and \( y \), i.e.,

\[
f^{10}(x, y) = (x^2 + y^2)^{0.25} \left[ \sin^2(50(x^2 + y^2)^{0.1}) + 1 \right]
\]

Accordingly, the extended \( F10 \) function is:
Table 1: Test problem for discrete optimization

<table>
<thead>
<tr>
<th>Activity No.</th>
<th>Depends On</th>
<th>Option 1</th>
<th>Option 2</th>
<th>Option 3</th>
<th>Option 4</th>
<th>Option 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Duration (days)</td>
<td>Cost ($)</td>
<td>Duration (days)</td>
<td>Cost ($)</td>
<td>Duration (days)</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>14</td>
<td>2 400</td>
<td>15</td>
<td>2 150</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>15</td>
<td>3 000</td>
<td>18</td>
<td>2 400</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>15</td>
<td>4 500</td>
<td>22</td>
<td>4 000</td>
<td>33</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>12</td>
<td>45 000</td>
<td>16</td>
<td>35 000</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>22</td>
<td>20 000</td>
<td>24</td>
<td>17 500</td>
<td>28</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>14</td>
<td>40 000</td>
<td>18</td>
<td>32 000</td>
<td>24</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>9</td>
<td>30 000</td>
<td>15</td>
<td>24 000</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>14</td>
<td>220</td>
<td>15</td>
<td>215</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>15</td>
<td>300</td>
<td>18</td>
<td>240</td>
<td>20</td>
</tr>
<tr>
<td>10</td>
<td>2, 6</td>
<td>15</td>
<td>450</td>
<td>22</td>
<td>400</td>
<td>33</td>
</tr>
<tr>
<td>11</td>
<td>7, 8</td>
<td>12</td>
<td>450</td>
<td>16</td>
<td>350</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>5, 9, 10</td>
<td>22</td>
<td>2000</td>
<td>24</td>
<td>1750</td>
<td>28</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>14</td>
<td>4000</td>
<td>18</td>
<td>3200</td>
<td>24</td>
</tr>
<tr>
<td>14</td>
<td>4, 10</td>
<td>9</td>
<td>3000</td>
<td>15</td>
<td>2400</td>
<td>18</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>12</td>
<td>4500</td>
<td>16</td>
<td>3500</td>
<td>—</td>
</tr>
<tr>
<td>16</td>
<td>13, 14</td>
<td>20</td>
<td>3000</td>
<td>22</td>
<td>2000</td>
<td>24</td>
</tr>
<tr>
<td>17</td>
<td>11, 14, 15</td>
<td>14</td>
<td>4000</td>
<td>18</td>
<td>3200</td>
<td>24</td>
</tr>
<tr>
<td>18</td>
<td>16, 17</td>
<td>9</td>
<td>3000</td>
<td>15</td>
<td>2400</td>
<td>18</td>
</tr>
</tbody>
</table>

With the initial schedule exceeding a desired deadline of 110-days, it is required to search for the optimum set of construction options that meet the deadline at minimum total cost. In this problem, the decision variables are the different methods of construction possible for each activity (i.e., five discrete options, 1 to 5, with associated durations and costs). The objective function is to minimize the total project cost (direct and indirect) and is formulated as follows:

$$
\text{Min} \quad (T \cdot I + \sum_{i=1}^{n} C_y)
$$

where $n =$ number of activities; $C_y =$ direct cost of activity $i$ using its method of construction $j$; $T =$ total project duration; and $I =$ daily indirect cost. To facilitate the optimization using the different EAs, macro programs of the 5 EAs were written using the VBA language that comes with the Microsoft Project software. The data in Table 1 were stored in one of the tables associated with the software. When any one of the EA routines is activated, the evolutionary process selects one of the five construction options to set the activities’ durations and costs. Accordingly, the project’s total cost (objective function) and duration changes. The evolutionary process then continues to attempt to optimize the objective function.

3.3. Parameter settings for evolutionary algorithms
As discussed earlier, each algorithm has its own parameters that affect its performance in terms of solution quality and processing time. To obtain the most suitable parameter values that suit the test problems, a large number of experiments were conducted. For each algorithm, an initial setting of the parameters was established using values previously reported in the literature[Emad, list the source references here]. Then, the parameter values were changed one by one and the results were monitored in terms of the solution quality and speed. The final parameter values adopted for each of the five EAs are given in the following.

**Genetic Algorithms:** The crossover probability \( (C_P) \) and the mutation probability \( (M_P) \) were set to 0.8 and 0.08, respectively. The population size was set at 200 and 500 offsprings. The evolutionary process was kept running until no improvements were made in the objective function for 10 consecutive generation cycles (i.e., 500 * 10 offsprings or the objective function reached its known target value, whichever comes first.

**Memetic Algorithms:** MAs are similar to GAs but apply local search on chromosomes and offsprings. The standard pair-wise interchange search does not suit the continuous functions \( F_8 \) and \( F_{10} \), and the local search procedure in Appendix III is used instead. For the discrete problem, on the other hand, the pair-wise interchange was used. The same values of \( C_P = 0.8 \) and \( M_P = 0.08 \) that were used for the GAs are applied to the MAs. After experimenting with various values, a population size of 100 chromosomes was used for the MAs.

**Particle Swarm Optimization:** Upon experimentation, the suitable numbers of particles and generations were found to be 40 and 10000, respectively. Also, the maximum velocity was set as 20 for the continuous problems and 2 for the discrete problem. The inertia weight factor \( \omega \) was also set as a time-variant linear function decreasing with the increase of number of generations where, at any generation \( i \),

\[
\omega = 0.4 + 0.8 * (\text{number of generations} - i) / (\text{number of generations} - 1)
\] (15)

such that \( \omega = 1.2 \) and 0.4 at the first and last generation, respectively.

**Ant Colony Optimization:** As the ACO algorithm is suited to discrete problems alone, no experiments were done using it for the \( F_8 \) and \( F_{10} \) test functions. However, the TCT discrete problem was used for experimentation with the ACO. After extensive experimentation, 30 ants and 100 iterations were found suitable. Also, the other parameters were set as follows: \( \alpha = 0.5; \beta = 2.5; \rho \) (pheromone evaporation rate) = 0.4; and \( R \) (reward factor depends on problem nature) = 10.

**Shuffled Frog Leaping:** Different settings were experimented with to determine suitable values for parameters to solve the test problems using the SFL algorithm. A population of 200 frogs, 20 memeplexes, and 10 iterations per memeplex were found suitable to obtain good solutions.

### 3.4. Results and discussions

The results found from solving the three test problems using the five evolutionary algorithms, which represents a fairly wide class of problems, are summarized in Tables 2 and 3, and Fig. 5 (the Y axis of Fig. 5 is a log scale to show long computer run times). It is noted that the processing time for solving the \( E_{F10} \) function was similar to that of the \( F_8 \) function and follows the same trend as shown in Fig. 5.

![Fig. 5. Processing time to reach the optimum for \( F_8 \) function](image-url)
known optimum value of zero), or 110 days for the TCT problem; or 2) the objective function value did not improve in ten consecutive generations. To experiment with different problem sizes, the \( F8 \) test function in Eq. (10) was solved using 10, 20, 50, and 100 variables, while the \( EF10 \) test function in Eq. (13) was solved using 10, 20, and 50 variables (it becomes too complex for larger numbers of variables).

Table 2 - Results of the continuous optimization problems

<table>
<thead>
<tr>
<th>Comparison criteria</th>
<th>Algorithm</th>
<th>( F8 ) ( EF10 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of variables</td>
<td></td>
</tr>
<tr>
<td>% Success</td>
<td>GAs (Evolver)</td>
<td>50  20  50  100  20  0  0 0</td>
</tr>
<tr>
<td></td>
<td>MAs</td>
<td>90  100  100  100  100  70  0</td>
</tr>
<tr>
<td></td>
<td>PSO</td>
<td>30  100  100  100  100  80  0</td>
</tr>
<tr>
<td></td>
<td>ACO</td>
<td>- - - - - - -</td>
</tr>
<tr>
<td></td>
<td>SFL</td>
<td>50  70  90  100  80  20  0</td>
</tr>
</tbody>
</table>

Mean solution

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( F8 )</th>
<th>( EF10 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAs (Evolver)</td>
<td>0.060</td>
<td>0.0060</td>
</tr>
<tr>
<td>MAs</td>
<td>0.014</td>
<td>0.013</td>
</tr>
<tr>
<td>PSO</td>
<td>0.093</td>
<td>0.011</td>
</tr>
<tr>
<td>ACO</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SFL</td>
<td>0.080</td>
<td>0.063</td>
</tr>
</tbody>
</table>

Table 3 - Results of the discrete optimization problem

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Minimum Project Duration (days)</th>
<th>Average Project Duration (days)</th>
<th>Minimum Cost ($)</th>
<th>Average Cost ($)</th>
<th>% Success Rate</th>
<th>Processing Time (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAs</td>
<td>113</td>
<td>120</td>
<td>162,270</td>
<td>164,772</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>MAs</td>
<td>110</td>
<td>114</td>
<td>161,270</td>
<td>162,495</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>PSO</td>
<td>110</td>
<td>112</td>
<td>161,270</td>
<td>161,940</td>
<td>60</td>
<td>15</td>
</tr>
<tr>
<td>ACO</td>
<td>110</td>
<td>122</td>
<td>161,270</td>
<td>166,675</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>SFL</td>
<td>112</td>
<td>123</td>
<td>162,020</td>
<td>166,045</td>
<td>0</td>
<td>15</td>
</tr>
</tbody>
</table>

Surprisingly, the GA performed more poorly than all the other four algorithms. In fact, it was found to perform more poorly than even that reported in Whitley et al. [24] and Raphael and Smith [26] when using the CHC and Genitor genetic algorithms, while it performed better than the ESGAT genetic algorithm version. A commercial GA package, Evolver [27], was used to verify the results. Evolver is an add-in program to Microsoft Excel, where the objective function, variables (adjustable cells), and the constraints are readily specified by highlighting the corresponding spreadsheet cells. Evolver performed almost the same way as the VB code with slight improvement. The results of using Evolver are reported in Table 2. The difference in Evolver’s results compared to those of the other EA algorithms may in part be because Evolver uses binary rather than real coding.

As shown in Table 2 for the \( F8 \) function, the GA was able to reach the target for 50% of the trials with 10 variables, and the number of successes decreased as the number of variables increased. Despite its inability to reach the optimum value of zero with the larger number of 100 variables, the GA was able to achieve a solution close to the optimum (0.432 for the \( F8 \) function with 100 variables). Also, it is noticed from Fig 5 that as the number of variables increased, the processing time to reach the target also increased (from 5min:12sec with 10 variables to 40min:27sec with 50 variables). As shown in Table 2 for the \( EF10 \) test function, the GA was only able to achieve 20% success using 10 variables, and that the solution quality decreased as the number of variables increased (e.g., the objective function = 5.951 using 50 variables). Using the GA to solve the TCT problem, the minimum solution obtained was 113 days with a minimum total cost of $162,270 and the success rate for reaching the optimum solution was zero, as shown in Table 3.

Upon applying the MA, the results improved significantly compared to those obtained using the GA, in terms of both the success rate (Table 2) and the processing time (Fig. 5). Solving the \( F8 \) function using 100 variables, for example, the success rate was 100% with a processing time of 7min:08 sec. Even for the trials with less success rate, as shown in Table 2, the solutions were very close to the optimum. That is to say, the local search of the MA improved upon the performance of the GA. When applying the MA to the TCT problem, it was able to reach the optimum project duration of 110 days and a total cost of $161,270, with a 20% success rate and an average cost
that improved upon that of the GA (Table 3). It is to be noted that the local search module presented in Appendix III was applied for the $F8$ and $EF8$ functions, while the pair-wise interchange local search module was applied to the TCT problem.

The PSO algorithm outperformed the GA and the MA in solving the $EF10$ function in terms of the success rate (Table 2), the processing time (Fig. 5), while it was less successful than the MA in solving the $F8$ function. Also, the PSO algorithm outperformed all other algorithms when used to solve the TCT problem, with a success rate of 60% and average total cost of $161,940, as shown in Table 3.

The ACO algorithm was applied only to the TCT discrete optimization problem. While it was able to achieve the same success rate as the GA (20%), the average total cost of the 20 runs was greater than that of all other algorithms (Table 3). This is due to the scattered nature of the obtained results (minimum duration of 110 days, and maximum duration of 139 days) caused by premature convergence that happened in some runs. To avoid premature convergence, the pair-wise inter-change local search module was applied and the results obtained were greatly improved with a success rate of 100%, but the average processing time increased from 10 to 48 seconds. When solving the $F8$ and $EF10$ test functions using the SFL algorithm, it was found that the success rate (Table 2) was better than the GA and similar to that for PSO. However, it performed less well when used to solve the $EF10$ function. As shown in Fig. 5, the SFL processing times were the least among all algorithms. Interestingly, it is noticed from Table 2 that as the number of variables increased for the $F8$ function, the success rates for SFL, MA and PSO all increased. This is because the $F8$ function becomes smoother as its dimensions increase [2]. As opposed to this trend, the success rate decreased for the GA as the number of variables increased. The same trend for the GA was also reported in [24] and [26] when used to solve the $F8$ function. Also, using the SFL algorithm to solve the TCT problem, the minimum duration obtained was 112 days with minimum total cost of $162,020 (Table 3). While the success rate for the SFL was zero, its performance was better than the GA.

It is interesting to observe that the behavior of each optimization algorithm in all test problems (continuous and discrete) was consistent. In particular, the PSO algorithm generally outperformed all other algorithms in solving all the test problems in terms of solution quality (except for the $F8$ function with 10 and 50 variables). Accordingly, it can be concluded that the PSO is a promising optimization tool, in part due to the effect of the inertia weight factor $\omega$. In fact, to take advantage of the fast speed of the SFL algorithm, the authors suggest using a weight factor in Eq. (3) for SFL that is similar to that used for PSO (some preliminary experiments conducted by the authors in this regard have shown good results).

4. Conclusions
In this paper, five evolutionary-based search methods were presented. These include: genetic algorithm (GA), memetic algorithm (MA), particle swarm optimization (PSO), ant colony optimization (ACO), and shuffled frog leaping (SFL). A brief description of each method is presented along with a pseudocode to facilitate their implementation. Visual Basic programs were written to implement each algorithm. Two benchmark continuous optimization test problems were solved using all but the ACO algorithm, and the comparative results were presented. Also presented were the comparative results found when a discrete optimization test problem was solved using all five algorithms. The PSO method was generally found to perform better than other algorithms in terms of success rate and solution quality, while being second best in terms of processing time.

### Appendix I. Pseudocode for a GA Procedure

Begin;
  Generate random population of $P$ solutions (chromosomes);
  For each individual $i \in P$: calculate fitness ($i$);
  For $i = 1$ to number of generations;
    Randomly select an operation (crossover or mutation);
    If crossover;
      Select two parents at random $i_a$ and $i_b$;
      Generate offspring $i_c = \text{crossover} (i_a \text{ and } i_b)$;
    Else If mutation;
      Select one chromosome $i$ at random;
      Generate an offspring $i_c = \text{mutate} (i)$;
    End if;
  Calculate the fitness of the offspring $i_c$;
  If $i_c$ is better than the worst chromosome then replace the worst chromosome by $i_c$;
  Next $i$;
  Check if termination = true;
End;
Appendix II. Pseudocode for a MA Procedure

Begin;
  Generate random population of $P$ solutions (chromosomes);
  For each individual $i \in P$: calculate fitness ($i$);
  For each individual $i \in P$: do local-search ($i$);
    For $i = 1$ to number of generations;
      Randomly select an operation (crossover or mutation);
      If crossover;
        Select two parents at random $i_a$ and $i_b$;
        Generate on offspring $i_c = \text{crossover} (i_a \text{ and } i_b)$;
        $i_c = \text{local-search} (i_c)$;
      Else If mutation;
        Select one chromosome $i$ at random;
        Generate an offspring $i_c = \text{mutate} (i)$;
        $i_c = \text{local-search} (i_c)$;
      End if;
      Calculate the fitness of the offspring;
      If $i_c$ is better than the worst chromosome then replace the worst chromosome by $i_c$;
    Next $i$;
  Check if termination = true;
End;

Appendix III. Pseudocode for the Memetic Local Search

Begin;
  Select an incremental value $d = a \times \text{Rand} ()$, where $a$ is a constant that suits the variable values;
  For a given chromosome $i \in P$: calculate fitness ($i$);
  For $j = 1$ to number of variables in chromosome $i$;
    Value ($j$) = Value ($j$) + $d$;
    If chromosome fitness not improved then Value ($j$) = Value ($j$) - $d$;
    If chromosome fitness not improved then retain the original value ($j$);
  Next $j$;
End;

Appendix IV. Pseudocode for a PSO Procedure

Begin;
  Generate random population of $N$ solutions (particles);
  For each individual $i \in N$: calculate fitness ($i$);
  Initialize the value of the weight factor, $\omega$;
    For each particle;
      Set $pBest$ as the best position of particle $i$;
      If fitness ($i$) is better than $pBest$;
        $pBest (i) = \text{fitness} (i)$;
      End;
    Set $gBest$ as the best fitness of all particles;
    For each particle;
      Calculate particle velocity according to Eq. 3;
      Update particle position according to Eq. 4;
    Update the value of the weight factor, $\omega$;
    Check if termination = true;
End;

Appendix V. Pseudocode for an ACO Procedure

Begin;
  Initialize the pheromone trails and parameters;
  Generate population of $m$ solutions (ants);
    For each individual ant $k \in m$: calculate fitness($k$);
    For each ant determine its best position;
    Determine the best global ant;
    Update the pheromone trail;
  Check if termination = true;
End;
Appendix VI. Pseudocode for a SFL Procedure

Begin;

   Generate random population of $P$ solutions (frogs);
   For each individual $i$ in $P$ calculate fitness ($i$);
   Sort the population $P$ in descending order of their fitness;
   Divide $P$ into $m$ memeplexes;
   For each memeplex;
      Determine the best and worst frogs;
      Improve the worst frog position using Eqs. 4 or 5;
      Repeat for a specific number of iterations;
   End;

   Combine the evolved memeplexes;
   Sort the population $P$ in descending order of their fitness;
   Check if termination = true;

End;

References

Asset management

While the civil infrastructure is the foundation for economic growth, a large percentage of its assets are rapidly deteriorating due to age, aggressive environment, and insufficient capacity for population growth. In 2003, the American Society of Civil Engineers released a report card on the infrastructures in the USA that gave failing grades to many infrastructure systems, and identified the need for $1.6 trillion (US) to bring the assets to acceptable condition (ASCE 2003). Similarly, the environmental, social, and transportation infrastructure systems in Canada require huge investments that amount to approximately $10 billion (US) annually for 10 years (Federation of Canadian Municipalities 1999). Since the environmental, social, and transportation sectors represent about 25% of the Canadian infrastructure expenditures (Figure 1, Statistics Canada 1995), it can be assumed that the infrastructure system as a whole requires an investment of about $40 billion per year for ten years. Despite this large need, the Infrastructure Canada Program allocated only $2 billion (US) for the year 2000 to all infrastructure sectors (Federation of Canadian Municipalities 2001), thus covering only about 5% of the need. With the non-residential buildings being largest sector of the infrastructure (approximately 40%), such sector is expected to suffer the largest shortfall in expenditures on rehabilitation and repair.

Average yearly expenditures by type of infrastructure

Important Questions:
- What assets do you own?
- What is it worth of each asset?
- What is its current condition of asset components?
- What is the remaining service life of the asset?
- What is the predicted condition in the future?
- What do you fix first?
- How do you fix it?
- What is the condition after the repair?
- How will you execute the many repairs?
Scheduling Repetitive & Linear Projects

- Problems with CPM & PDM
- Resource-Driven Scheduling
- Crew Work Continuity
- Learning Phenomenon

Integrated CPM & LOB Calculations:

**New Representation:**

![Diagram of crew synchronization calculations]

**Crew Synchronization Calculations:**

Crews \((C) = (D) \times (R)\)

**Calculating a Desired Progress Rate \((R)\):**

![Diagram showing the calculation of \((R)\)]
Example:
For this small project, the work hours and the number of workers for each activity are shown. If you are to construct these tasks for 5 houses in 21 days, calculate the number of crews that need in each activity. Draw the schedule and show when each crew enters and leaves the site.

**Step 1: CPM Calculation**

**Step 2: LOB Calculations**  
Deadline $T_L = 21$; $T_1 = ____$; $n = 5$

<table>
<thead>
<tr>
<th>Activity</th>
<th>Duration (D)</th>
<th>Total Float (TF)</th>
<th>Desired Rate (R) ((n-1)/(T_L-T_1+TF))</th>
<th>Min. Crews ((C) = D \times R)</th>
<th>Actual Crews ((C_a))</th>
<th>Actual Rate ((R_a) = C_a / D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Step 3: Draw the Chart**

Draw the critical path

Assume:
- Same no. of crews
- Activity A in unit 2 has double the duration
- Unit 4 does not need excavation.
Basic scheduling for repetitive projects

- Activate the **BAL-1.mpp** file. Look at the activities of this one unit of a small project. The duration of this unit is _______ days. Now, we have 5 units to construct within 45 days. Go ahead and activate the macro Auto_Open by selecting **Tools-Macro-Macros**.

- Change the number of crews in activity D into 4 crews. Do you still meet the deadline? If not, try to meet it without changing the crews. Try using interruption.

More Advanced Linear scheduling Model

Flexible features for scheduling the activities include: color-coded or pattern-coded crews; varying quantities; productivity impact; crew interruption time; crew staggering; crew work sequence; and activities' progress speeds (slopes of lines). It is noted that the schedule is efficiently arranged with crew work continuity maintained. Also, overlapping is avoided by simply showing the activities of each path in the work network separately. In addition:

1. Activities are not necessarily repeated at all sections.
2. Activities can proceed in an ascending or descending flow. This provides work flow flexibility and provides for a way to fast-track projects;
3. Each activity has up to 3 methods of construction (e.g., normal work, overtime, or subcontractor) with associated time, cost, and crew constraints. The model can then be used to select the proper combination of methods that meet the deadline, cost, and crew constraints;
4. Activities can have non-standard durations and costs at selected sections;
5. Work interruption (layoff period) can be specified by the user at any unit of any activity; and
6. Conditional methods of construction can be specified by the user.

![Diagram](image-url)
LINEAR SCHEDULING: Highway Example

A three-kilometer highway stretch is divided to ten sections for planning purposes. Each section is 300 meters. The cross section is shown below along with activities’ details.

For the highway project discussed earlier, let’s develop an optimum schedule considering different realistic options of crews and how they move among the ten stations. The data are as follows:

<table>
<thead>
<tr>
<th>Activity</th>
<th>Applicable Sections</th>
<th>Max. Crews</th>
<th>Cost ($)</th>
<th>Time (days)</th>
<th>Cost ($)</th>
<th>Time (days)</th>
<th>Cost ($)</th>
<th>Time (days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Excavation, East</td>
<td>1 to 5</td>
<td>2</td>
<td>21,000</td>
<td>3</td>
<td>30,000</td>
<td>2</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>2. Sub-base, East</td>
<td></td>
<td>2</td>
<td>7,800</td>
<td>2</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>3. Base, East</td>
<td></td>
<td>3</td>
<td>72,000</td>
<td>10</td>
<td>80,000</td>
<td>8</td>
<td>100,000</td>
<td>5</td>
</tr>
<tr>
<td>4. Binder, East</td>
<td></td>
<td>1</td>
<td>30,000</td>
<td>1.2</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>5. Asphalt, East</td>
<td></td>
<td>1</td>
<td>14,400</td>
<td>1</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>6. Curbs, East</td>
<td></td>
<td>1</td>
<td>31,200</td>
<td>2</td>
<td>38,000</td>
<td>1</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>7. Lighting, East</td>
<td></td>
<td>2</td>
<td>19,245</td>
<td>2</td>
<td>25,000</td>
<td>1</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>8. Sidewalks, East</td>
<td></td>
<td>2</td>
<td>10,950</td>
<td>2</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>9. Paint, East to West</td>
<td>1 to 10</td>
<td>1</td>
<td>198</td>
<td>0.2</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>10 to 17. Same as 1-8 but at West</td>
<td>10 to 6</td>
<td></td>
<td>Same as activities 1 to 8.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The logical relationships within each section are the same, but the deadline for finishing the whole highway is 30 days. Seasonal productivity factors are also as shown below.

How will your plan differ if you start the whole project from one side as opposed to employing different sets of crews from both sides?

Activate the BAL-2.mpp file, which will run the macro Auto_Open from the Tools-Macro-Macros.
Infrastructure Networks with Distributed Sites: A Bigger Challenge
Buildings, Hospitals, Schools, Highway Spots, Bridges

Effect of Site order

From the BAL-3 directory, activate BAL-DEMO.Exe. Go to the resource bank, view all sites, and change the productivity factors only for the first four sites to: January 0.7, February 0.8 and March 0.9, and leave all others as 1.0s. In BAL main screen, activate the BAL-Schedule button to access MS Project. Use Project-Information to change project start data to Jan. 2, 2002. Use the BAL Schedule toolbar button, then the Project Data button. Change the deadline to March 25, 2002. Go to the Activities tab and scroll through the activities. Change the maximum number of crews for activity “Subbase” to 3. Save and proceed. Try to meet the deadline. Use optimization options. After every trial notice the arrangement of the sites, the number of crews used, and the method of construction used. Try manually to shift the sites that take long durations later in the order. Notice the effect on time and cost. Best duration obtained is ____ days and minimum cost is_______.

Delivery Approaches for Infrastructure MR&R Programs

- Using in-House Resources
  - Emergency projects
  - No time to accurately define scope, write contract, & evaluate bids
  - Internal expertise available for design and execution

- Outsourcing to contractors
  - Risky projects
  - Specialized work
  - Enough time to accurately define scope, write contract, & evaluate bids
  - No internal expertise available

- Combination of Both
  - Suits various types of MR&R programs
  - Suits large organizations with diverse programs

Effect of Site order

Scheduling of crews along multiple sites

From the BAL-3 directory, activate BAL-DEMO.Exe. Go to the resource bank, view all sites, and change the productivity factors only for the first four sites to: January 0.7, February 0.8 and March 0.9, and leave all others as 1.0s. In BAL main screen, activate the BAL-Schedule button to access MS Project. Use Project-Information to change project start data to Jan. 2, 2002. Use the BAL Schedule toolbar button, then the Project Data button. Change the deadline to March 25, 2002. Go to the Activities tab and scroll through the activities. Change the maximum number of crews for activity “Subbase” to 3. Save and proceed. Try to meet the deadline. Use optimization options. After every trial notice the arrangement of the sites, the number of crews used, and the method of construction used. Try manually to shift the sites that take long durations later in the order. Notice the effect on time and cost. Best duration obtained is ____ days and minimum cost is_______.

Delivery approaches for MR&R programs