The technique used by Solver is widely known as branch & bound.

We can search for an optimal solution in three steps:

- Define the cost variable.
- Link the cost variable to other variables in the problem.
- Define the search for optimal solutions.
Performance Objective

- Basically, the cost variable is a constrained variable and its value is computed by the given objective function.
- Therefore, this variable becomes the objective to be minimized.
- A constraint is used to link the cost variable to other variables.
- For example,
  \[ \text{IlcIntVar cost} = 15*W + 10*P + 7*C \]
Branch and Bound Algorithm

- The `IlcManager::setObjMin()` can be used to activate the optimization mechanism in ILOG Solver.

- To perform Branch&Bound, we have to loop to find better and better solutions.

- The loop will actually produce a series of solutions, ending with an optimal solution.
int main()
{
    IlcManager m(IlcEdit);

    IlcIntVar W(m, 0, 9), P(m, 0, 9), C(m, 0, 9);
    IlcIntVarArray vars(m, 3, W, P, C);

    for(IlcInt i=0;i < 3;i++){
        vars[i].setStorable();
    }

    m.add(4*W+3*P+2*C<=9);

    IlcIntVar cost=15*W + 10*P + 7*C;
    m.setObjMin(-cost);

    // ... To be continued
Smugglers Knapsack Problem

//...

m.add(IlcGenerate(vars));
while (m.nextSolution()){
    m.out() << "Cost: " << cost << endl;
    m.storeSolution();
}

m.restart();
m.nextSolution();
m.out() << endl << "Optimal Solution: " << vars << endl;

m.end();
return 0;
Cost: [0]
Cost: [7]
Cost: [14]
Cost: [21]
Cost: [28]
Cost: [31]
Cost: [32]

Optimal Solution: IlcIntVarArrayI[[1] [1] [1]]
storeSolution() & restart()

- The `storeSolution()` stores the current value of every constrained variable which has been marked by the member functions `setStorable`.

- The `restart()` restores the invoking manager to the state before the first call to `nextSolution`, and it puts the manager in edit mode if the manager was created in edit mode.

- It also stores the best value of any objective function that has been defined by `setObjMin`.

- If you have stored an intermediate solution, then you should call `restart` before you call `nextSolution` so that `nextSolution` will immediately produce that stored solution.
An Utility Function: \texttt{IlcMax}

\begin{itemize}
  \item \texttt{IlcIntExp IlcMax(IlcInt exp1, const IlcIntExp exp2)};
  \item \texttt{IlcIntExp IlcMax(const IlcIntExp exp1, IlcInt exp2)};
  \item \texttt{IlcIntExp IlcMax(const IlcIntExp exp1, const IlcIntExp exp2)};
  \item \texttt{IlcInt IlcMax(IlcInt exp1, IlcInt exp2)};
  \item \texttt{IlcFloatExp IlcMax(const IlcFloatExp exp1, const IlcFloatExp exp2)};
  \item \texttt{IlcFloatExp IlcMax(const IlcFloatExp exp1, IlcFloat exp2)};
  \item \texttt{IlcFloatExp IlcMax(IlcFloat exp1, const IlcFloatExp exp2)};
  \item \texttt{IlcFloat IlcMax(const IlcFloat exp1, const IlcFloat exp2)};
  \item \texttt{IlcFloatExp IlcMax(const IlcFloatVarArray array)};
  \item \texttt{IlcFloat IlcMax(const IlcFloatArray array)};
  \item \texttt{IlcIntExp IlcMax(const IlcIntVarArray array)};
  \item \texttt{IlcInt IlcMax(const IlcIntArray array)};
\end{itemize}
This function returns the maximum of its argument or arguments.

When its argument is an array of constrained expressions, it creates a new constrained expression equal to the maximum of the elements in that array.
Yet Another Map-Coloring Problem
Six countries and six colors

IlcIntVar Belgium(m, 0, 5);
IlcIntVar Denmark(m, 0, 5);
IlcIntVar France(m, 0, 5);
IlcIntVar Germany(m, 0, 5);
IlcIntVar Netherlands(m, 0, 5);
IlcIntVar Luxembourg(m, 0, 5);
IlcIntVarArray AllVars(m, 6, Belgium, Denmark, France, Germany, Netherlands, Luxembourg);
m.add(France != Belgium);
m.add(France != Luxembourg);
m.add(France != Germany);
m.add(Luxembourg != Germany);
m.add(Luxembourg != Belgium);
m.add(Belgium != Netherlands);
m.add(Germany != Netherlands);
m.add(Germany != Denmark);
int main()
{
    IlcManager m(IlcEdit);

    // Define variables and constraints

    // Search for minimal number of colors
    m.add(IlcGenerate(AllVars));
    for(IlcInt i=0;i < 6;i++)
    {
        AllVars[i].setStorable();
    }

    IlcIntVar cost = IlcMax(AllVars);
    m.setObjMin(cost);
    while(m.nextSolution())
    {
        m.out() << "Cost: " << cost << endl;
        m.storeSolution();
    }

    m.restart();
    m.nextSolution();
    m.out() << "Optimal Solution: " << AllVars << endl;
}
Cost: [3]
Cost: [2]
Optimal Solution: Color[[0] [1] [1] [0] [1] [2]]
Any Questions...