GIIE Course: 546U4030 SOFT COMPUTING METHODS AND APPLICATIONS

Assignment 09 (2020)
Binary- and Permutation-Encoded GAs for Job Assignment Problem

Continued with your pervious project of designing a generic GA solver and a derived binary solver, you are asked to complete the implementation of the binary encoded GA solver and implement a permutation-encoded GA solver to solve the job assignment optimization problems. You are asked to design the computation logic of all of the crossover operators of the permutation encoded GA and display the logic in pseudo code first, before implementing them. **Submit the pseudo code in Word or PowerPoint file**. For example: the pseudo code of the Partial Map Crossover Operation is:

```
// i1, i2 cut locations
// m[] partial map, m[i] => the mapping target of i
PMX( p1, p2, c1, c2)
i1←Random(n); i2←Random(n)
if i1 > i2 swap i1 and i2
m[i] ←-1, for all i, i = 0, 1, …, n-1
for i ← i1 to i2-1
   if p1[i] = p2[i] then next i // no mapping
   if m[p1[i]] = -1 and m[p2[i]] = -1
       m[p1[i]] ← p2[i]; m[p2[i]] ← p1[i]
   else if m[p1[i]] = -1
       m[p1[i]] ← m[p2[i]]; m[m[p2[i]]] ← p1[i]; m[p2[i]] ← -2
   else if m[p2[i]] = -1
       m[p2[i]] ← m[p1[i]]; m[m[p1[i]]] ← p2[i]; m[p1[i]] ← -2
   else
       m[m[p2[i]]] ←m[p1[i]]; m[m[p1[i]]] ← m[p2[i]]
       m[p1[i]] ← -3; m[p2[i]] ← -3
for i ← 0 to n-1
   if i1 <= i and i < i2
       c1[i] ← p2[i]; c2[i] ← p1[i]
   else
       if m[p1[i]] < 0 then c1[i] ← p1[i]
       else c1[i] ← m[p1[i]]
       if m[p2[i]] < 0 then c2[i] ← p2[i]
       else c2[i] ← m[p2[i]]
```

You are asked to submit pseudo code for the rest of crossover operators and mutation operators in the file. You can add any data structure to facilitate the computation logic of your algorithms; however, do not dynamically allocate memory within the code, since it is repeatedly called. Instead, allocate required memory only once in advance.

Refer to Assignment 07 to design your UIs to read in a benchmark file of the Job Assignment Problem and select an encoding type of GA solver to solve the problem. Note that your BinaryGA uses *byte* type to store either value 1 or 0 for easier implementation and efficient computation. User should be able to set number of cuts for crossover operations.

Job Assignment Problems Solved by the Binary-encoded GA:

Company A has *n* machines and *n* jobs to be processed. Each machine must be assigned with exactly one job. The time required to set up each machine for processing each job is different and given in a time matrix *C*. Company A wants to minimize the total setup time needed to complete the jobs. Find the solution using a binary-encoded GA model.

$$C = \left[c_{ij} \right]_{nxn} = \begin{matrix} Machines \\ Jobs \end{matrix} \begin{bmatrix} \\ c_{ij} \end{bmatrix}$$

$$\min \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} c_{ij} x_{ij}$$

s.t.

$$\sum_{i=0}^{n-1} x_{ij} = 1;\ j = 0,1,...,n-1$$

$$\sum_{j=0}^{n-1} x_{ij} = 1; i = 0,1,...,n-1$$

$$x_{ij} = \begin{cases} 0 \\ 1 \end{cases}$$

There will be n^2 0-1 genes subject to *n* machine-wise constraints and *n* job-wise constraints. Evaluate the violation amount of these hard constraints as penalties on the objective function. The objective function is then

$$\min f(\mathbf{x}) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} c_{ij} x_{ij} + M \cdot \left(\sum_{j=0}^{n-1} \left| \left(\sum_{i=0}^{n-1} x_{ij} \right) - 1 \right| + \sum_{i=0}^{n-1} \left| \left(\sum_{j=0}^{n-1} x_{ij} \right) - 1 \right| \right),$$

where *M* is a user-specified positive penalty multiplier. In your solving system, provide a user interface control to let user specify the weight *M* of penalty. Note that the variables defined in the above equations are in matrix format, while our variables should be a list of binary variables stored in a segment of chromosomes. Therefore, you need to transform the two dimensional variables into a one-dimensional binary variable array. For example,

$$\mathbf{x}' = \left[x'_0 x'_1 \cdots x'_{n^2-1} \right]$$

$$\min f'(\mathbf{x}') = \sum_{k=0}^{n^2-1} c_{\left(\left\lfloor \frac{k}{n} \right\rfloor \right)(k \bmod n)} x'_k + M \cdot \left(\sum_{j=0}^{n-1} \left| \left(\sum_{k=jn}^{jn+n-1} x'_k \right) - 1 \right| + \sum_{j=0}^{n-1} \left| \left(\sum_{k=0}^{n-1} x'_{kn+j} \right) - 1 \right| \right)$$

or

$$\min f'(\mathbf{x}') = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} c_{i,j} x'_{(i \cdot n+j)} + M \cdot \left(\sum_{j=0}^{n-1} \left| \left(\sum_{i=0}^{n-1} x'_{(i \cdot n+j)} \right) - 1 \right| + \sum_{i=0}^{n-1} \left| \left(\sum_{j=0}^{n-1} x'_{(i \cdot n+j)} \right) - 1 \right| \right)$$

You need to define the objective evaluation function correctly to return the objective value that has aggregated penalties of the hard constraint violations.

Job Assignment Problems Solved by the Permutation-encoded GA:

Assume that $\mathbf{x} = [x_0, x_1, \cdots, x_{n-1}], x_j \in \{0,1,\cdots,n-1\} \wedge x_j \neq x_{j'}$ is a solution of the problem, where job x_j is assigned to machine *j*. Therefore, the goal of the problem is to

$$\min\ f(\mathbf{x}) = \sum_{j=0}^{n-1} c_{x_j,j}$$

Deriving from your generic GA solver to implement your permutation-encoded GA solver, say `PermutationGA : GenericGA<int>`. You are encouraged to implement all 6 canonical crossover (Partial Map, Order, Position-based, Order-based, Cycle, Sub-tour) and 5 mutation operators (Inversion, Swapped, Insertion, Displacement, Reciprocal Exchange). Implement as many operators as possible to earn high scores and polish your programing skills as well. To add properties for user to select different types of operators provided in your new GA solvers, exercise yourself to define `enum` (enumeration) types. Carefully implement the overridden crossover and mutation operators, where no `new` operations should be executed in these repeatedly called functions. Instead, define data for the operations as data member of the class that are instantiated only once in advance and repeatedly used by these functions.

Provide the objective evaluation function for the permutation-encoded model to solve the job assignment problem.

Appendix: Extended File format of the Job Assignment Problem with extension “aop”.

7	*n*
14 5 8 7 6 10 8	c_{00}~$c_{0(n-1)}$
2 12 6 5 9 4 3	c_{10}~$c_{1(n-1)}$
7 8 3 9 8 11 4	c_{20}~$c_{2(n-1)}$
2 4 6 10 7 9 3	c_{30}~$c_{3(n-1)}$
5 8 10 14 3 5 9	c_{40}~$c_{4(n-1)}$
4 8 7 6 10 8 7	c_{50}~$c_{5(n-1)}$
3 5 7 8 6 4 7	c_{60}~$c_{6(n-1)}$
26	minimal setup time
1 0 2 5 4 6 3	optimal solution (if available)