Track 2

Solving Post Enrolment based Course Timetabling Using Differential Evolution

Differential evolution (DE) is a relatively new evolutionary algorithm proposed by Storn and Price 1995, DE is applied successfully to solve a wide range of optimization problems. The suitability of applying Differential evolution toward solving the course scheduling problem is investigated in solving post enrolment based course timetabling problem.

Differential evolution is the relative new evolution algorithm that has been proposed by Storn and Price (1995). DE is similar to Genetic Algorithms (GAs) in that a population of individuals are used to search for an optimal solution (Feoktistov and Janaqi 2004). The main difference between GAs and DE is that, in GAs, mutation is the result of small perturbations to the genes of an individual while in DE mutation is the result of arithmetic combinations of individuals. DE uses the differences between randomly selected vectors (individuals) as the source of random variations for a third vector (individual), referred to as the target vector. Trial solutions are generated by adding weighted difference vectors to the target vector. This process is referred to as the mutation operator where the target vector is mutated. A recombination, or crossover step is then applied to produce an offspring which is only accepted if it improves on the fitness of the parent individual.

Explanation for our method:

At the beginning a random floating-point represents a priority will be given for each individual recourse (i.e. rooms, time slots).

A suitable room and time slot will be assigned to each event (e) as follows:

Step1: Let V`=∅, with V`⊆ V, where V is the array for all rooms.

Step2: For All j∈ V do

If j is big enough for all the attending students and satisfies all the features required by the event e , Then V`= V`∪{j}, i.e. insert j in set V`.

Step3: Determine the component *i* with highest priority $\psi_{I,}$ for all *j* in set V`.

Step4: Assigned i to the current event e.

Now selecting time slot;

Step6: Let T`=∅, with T`⊆ T, where T is the array for all Time slots.

Step7: For all *t* ∈ T do

if room i which selected in step 3 is not already occupied in t by other event, and if no student is already has event in time t (i.e. t will not produce conflict in students schedule), and if t will not violate the events' order constraint, and event e is available at time t. then T`=T`∪{j}, i.e. insert t in set T`.

Step8: Determine the component t with highest value of $\psi_{I,}$ for all *t* in set V`.

Step9: assigned t to the current event e.

Other wise if no suitable room, time is available to event e, then event e will not be schedule "i.e. assign -1 for room and time slto).

Steps 1 to 10 will be repeated for all event. Moreover, in each generation, new mutant priority will be assigned for each time slot and room. By applying a convention mutation, crossover operations as follows:

Step 1: for Timeslot vector:

For each element $t_{i,j}$ in T vector, a perturbed (offspring), $v_{i,j}$

is generated according to following;

If (U(0, 1) < Pr)

$v_{i,j} = t_{r_1,j} + F(t_{r_2,j} - t_{r_3,j})$

Otherwise;

$v_{i,j} = t_{i,j}$

Where $r_1, r_2, r_3 \in (1,..., n)$,n the number of parent "schedule", and $r_1 \neq r_2 \neq r_3 \neq i$. F is a real and constant factor $\in$ [0, 2] which controls the amplification of the differential $(t_{r_2,j} - t_{r_3,j})$, where p is the number of timeslots available, Pr is the probability of reproduction (with Pr $\in$ [0, 1]).

Step 2: for room vector:

For each element $f_{i,j}$ in F vector, a perturbed (offspring), $v_{i,j}$

is generated according to following;

If (U(0, 1) < Pr)

$v_{i,j} = f_{r_1,j} + F(f_{r_2,j} - f_{r_3,j})$

Otherwise;

$v_{i,j} = f_{i,j}$

where $r_1, r_2, r_3 \in (1,..., n)$,n the number of parent "schedule", and $r_1 \neq r_2 \neq r_3 \neq i$. F is a real and constant factor $\in$ [0, 2] which controls the amplification of the differential $(f_{r_2,j} - f_{r_3,j})$, where f is the number of rooms available, Pr is the probability of reproduction (with Pr $\in$ [0, 1]). With each new generation, an offspring schedule only will be accepted if it improves on the fitness of the parent individual.

Selection: a new schedule $m_i(t)$ will be generated for each parent schedule using the mutated value of priority. The generated offspring, $m_i(t)$, replaces the parent, $x_i(t)$, only if the fitness of the offspring is better than that of the parent, where fittnes calculated by the total number of violating soft constraints and the distance to feasibility. The offspring is in better fitness if violate a smaller number of soft contrarians that its parent.

Mutation, crossover, and selection operations will be repeated till max number of generations is reached or if the executed time is finished. A best schedule obtained in all generation will be saved.