Support to Sprint C of ALGAV Practical Work Delivery Planning with Electric Trucks

Curricular Unit: Advanced Algorithms
Programme: BSc Informatics Engineering (ISEP)

Basic Genetic Algorithm

Generate P_0 ← Initial Population

$t \leftarrow 0$ ← Iteration (1st)

Avaliar P_t

while not Final Condition of P_t **do** ← Solutions' Population in iteration **t**

Generate new individuals

$P'_t \leftarrow$ Select P_t ← Select the most fit

$P''_t \leftarrow$ Apply recombination operators P'_t

$P'''_t \leftarrow$ Apply mutation operators P''_t

Evaluate P'''_t ← Evaluate the fitness of solutions

$P_{t+1} \leftarrow$ Select the survivors from P_t and P'''_t

$t \leftarrow t + 1$ ← New iteration

endwhile

return Better Global Solution

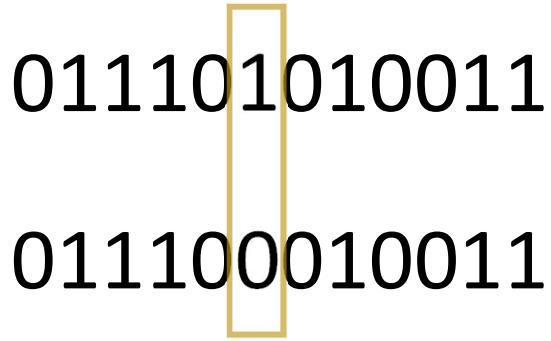
Crossover

Order 1 crossover

	A	B
	123**4567**89	452**187**693
Copy the genes in the middle of A, the remainder will be replaced by H →	HHH**4567**HH	
Shift B 2 positions (the ones after the 2nd crossover point) →		934521876
Remove genes already existant in A →		93218
Replace the H's by the previous after the 2nd crossover point →	218**4567**93	

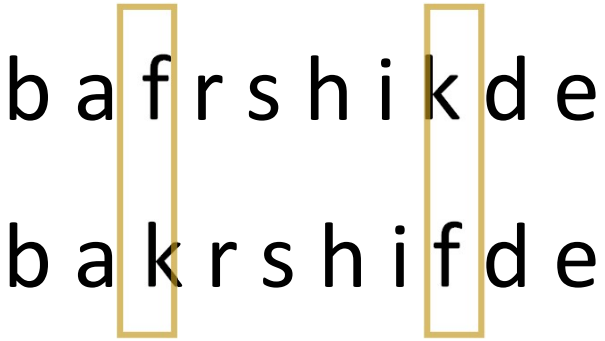
Mutation

Binary Genes



Replace a 0 by 1 or 1 by 0)

Genes sequences



Swap elements between two positions

Example – Task sequencing

schedule 5 tasks in one machine

- For each task **j** (j=1, …, 5),
  - $\mathbf{p_j}$ processing time,
  - $\mathbf{d_j}$ due date
  - $\mathbf{w_j}$ weight of the penalty in case of delay
- Goal: minimize the weighted sum of delays $\mathbf{\Sigma w_j T_j}$, where $\mathbf{T_j}$ is task j delay

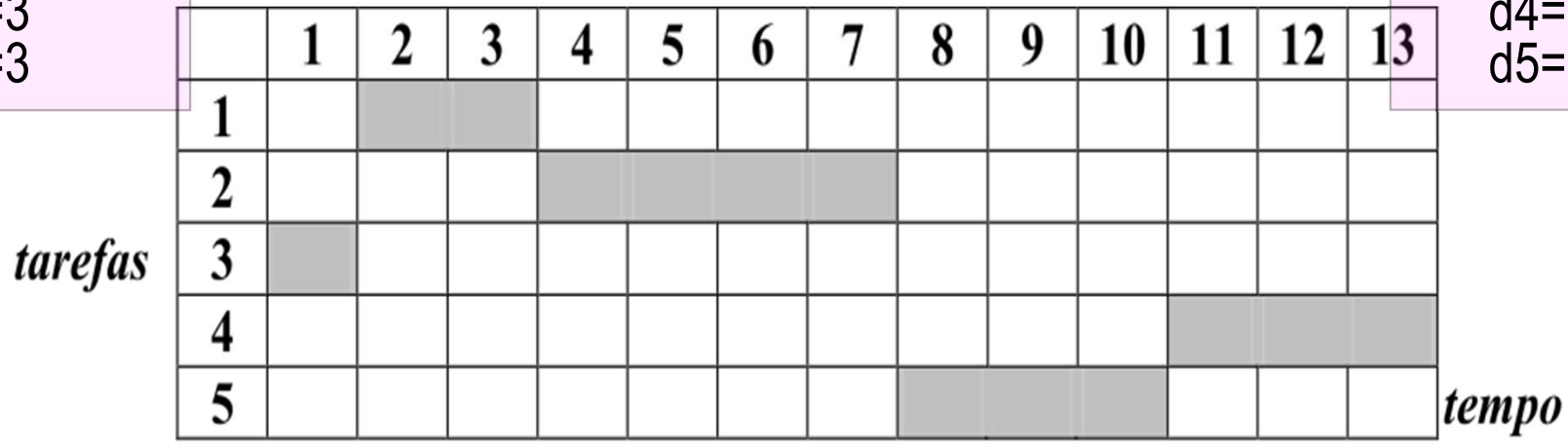
Example (cont)

For visualizing a sequence Gantt Diagrams are often used *tasks x time*

For the tasks' sequence [3,1,2,5,4] we obtain the scheduling below

Processing Time
p1=2
p2=4
p3=1
p4=3
p5=3

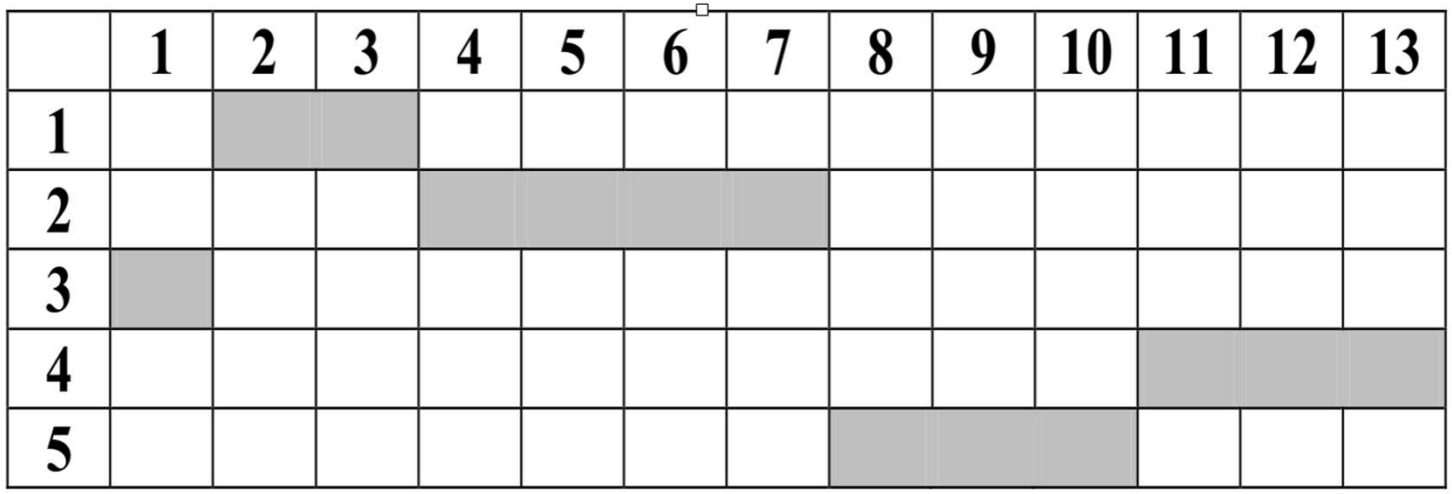
Due date
d1=5
d2=7
d3=11
d4=9
d5=8



Example – Task sequencing

GANTT chart for one possible solution [t3,t1,t2,t5,t4]

Processing Time	Due date	Weight
p1=2	d1=5	w1=1
p2=4	d2=7	w2=6
p3=1	d3=11	w3=2
p4=3	d4=9	w4=3
p5=3	d5=8	w5=2



$$\Sigma w_j T_j = 0+0+0+(13-9)x3+(10-8)x2=4x3+2x2=16$$

Example (cont)

Recombination Method

Order 1 crossover

1. Assuming the following pair of individuals, 2 crossing points are selected (4th and 7th).

2. The genes located between the two crossing points are even copied to their descendants, with the remaining positions filled with an H character.

A=123**4567**89 A'=HHH**4567**HH

B=452**1876**93 B'=HHH**1876**HH

Example (cont)

Recombination Method (cont.)

3. Then, and starting at the second crossing point of father B, a new order for the genes is defined:

[9 3 4 5 2 1 8 7 6]

B was 452**1876**93

4. After removing genes 4, 5, 6 and 7 already defined in child A', we are left with the genes [9 3 2 1 8]. The positions at A' containing H will be filled by this sequence starting at the second crossing point:

A'=218**4567**93

A' was HHH**4567**HH

Example (cont)

Recombination Method (cont.)

5. In the same way to generate the second descendant B ', a new sequence is defined starting from A:

[8 9 1 2 3 4 5 6 7].

A was 123**4567**89

6. Eliminating the genes already defined in B ', we obtain the sequence: [9 2 3 4 5]. Then the sequence obtained is replaced in the gaps (H) of B ', starting at the 2nd crossing point, obtaining the descendant B'.

B'=345**1876**92

Genetic Algorithm in PROLOG

```
% tarefa(Id,TempoProcessamento,TempConc,PesoPenalizacao). // task
tarefa(t1,2,5,1).
tarefa(t2,4,7,6).
tarefa(t3,1,11,2).
tarefa(t4,3,9,3).
tarefa(t5,3,8,2).
```

Tasks definition

```
% tarefas(NTarefas).//tasks number
tarefas(5).

% parameterização / parameters initialization
inicializa:-write('Numero de novas Geracoes: '),read(NG),
            (retract(geracoes(_));true), asserta(geracoes(NG)),
            write('Dimensao da Populacao: '),read(DP),
            (retract(populacao(_));true), asserta(populacao(DP)), %population size
            write('Probabilidade de Cruzamento (%):'), read(P1), % crossover probability
            PC is P1/100,
            (retract(prob_cruzamento(_));true),     asserta(prob_cruzamento(PC)),
            write('Probabilidade de Mutacao (%):'), read(P2), %mutation probability
            PM is P2/100,
            (retract(prob_mutacao(_));true), asserta(prob_mutacao(PM)).
```

GA parameter initialization

One possible implementation in PROLOG

```
gera:- %generate
        inicializa,
        gera_populacao(Pop),
        write('Pop='),write(Pop),nl,
        avalia_populacao(Pop,PopAv),
        write('PopAv='),write(PopAv),nl,
        ordena_populacao(PopAv,PopOrd),
        geracoes(NG),
        gera_geracao(0,NG,PopOrd).

gera_populacao(Pop):- %generate_population
        populacao(TamPop),
        tarefas(NumT),
        findall(Tarefa,tarefa(Tarefa,_,_,_),ListaTarefas),
        gera_populacao(TamPop,ListaTarefas,NumT,Pop).

gera_populacao(0,_,_,[]):-!. %generate_population
gera_populacao(TamPop,ListaTarefas,NumT,[Ind|Resto]):-
        TamPop1 is TamPop-1,
        gera_populacao(TamPop1,ListaTarefas,NumT,Resto),
        gera_individuo(ListaTarefas,NumT,Ind),
        not(member(Ind,Resto)).
gera_populacao(TamPop,ListaTarefas,NumT,L):-
        gera_populacao(TamPop,ListaTarefas,NumT,L).
```

PopOrd is the list with the initial generation
Each element of the list is of type LT * Av where LT is a task list (individual), for example [t3, t1, t5, t2, t4], and Av the respective assessment in terms of the heavy sum of the delays, an element PopOrd could be
[t3, t1, t5, t2, t4] * 16, where * is a mere separator

gera_população/1 creates a population of individuals

gera_individuo/3 creates an individual
with all tasks, each task is a gene and the individual corresponds to the chromosome

```
gera_individuo([G],1,[G]):-!. %generate_iindividuaçl

gera_individuo(ListaTarefas,NumT,[G|Resto]):-
                NumTemp is NumT + 1, % para usar com random
                random(1,NumTemp,N),
                retira(N,ListaTarefas,G,NovaLista),
                NumT1 is NumT-1,
                gera_individuo(NovaLista,NumT1,Resto).

retira(1,[G|Resto],G,Resto).% removes
retira(N,[G1|Resto],G,[G1|Resto1]):- N1 is N-1,
                                        retira(N1,Resto,G,Resto1).
```

```
avalia_populacao([],[]). % evaluate_population (according with the fitness function
avalia_populacao([Ind|Resto],[Ind*V|Resto1]):-
        avalia(Ind,V),
        avalia_populacao(Resto,Resto1).

avalia(Seq,V):- avalia(Seq,0,V).

avalia([ ],_,0).
avalia([T|Resto],Inst,V):-
        tarefa(T,Dur,Prazo,Pen),
        InstFim is Inst+Dur,
        avalia(Resto,InstFim,VResto),
        ((InstFim =< Prazo,!, VT is 0) ; (VT is (InstFim-Prazo)*Pen)),
        V is VT+VResto.
```

avalia_população/2 will evaluate all individuals in the population (each individual is a list of all tasks) according to the heavy sum of delays V and creates a list (second argument) with elements in the format individual * evaluation

```
ordena_populacao(PopAv,PopAvOrd):-
        bsort(PopAv,PopAvOrd).

bsort([X],[X]):-!.
bsort([X|Xs],Ys):-
        bsort(Xs,Zs),
        btroca([X|Zs],Ys).


btroca([X],[X]):-!.

btroca([X*VX,Y*VY|L1],[Y*VY|L2]):-
        VX>VY,!,
        btroca([X*VX|L1],L2).

btroca([X|L1],[X|L2]):-btroca(L1,L2).
```

Sorting population elements in ascending order
of evaluations by the heavy sum of delays

```
gera_geracao(G,G,Pop):-!, %generate_generation
        write('Geração '), write(G), write(':'), nl, write(Pop), nl.
gera_geracao(N,G,Pop):-
        write('Geração '), write(N), write(':'), nl, write(Pop), nl,
        cruzamento(Pop,NPop1),
        mutacao(NPop1,NPop),
        avalia_populacao(NPop,NPopAv),
        ordena_populacao(NPopAv,NPopOrd),
        N1 is N+1,
        gera_geracao(N1,G,NPopOrd).
```

gera_geração/3 –the new generations of the population are created after the crossover, mutations and evaluation of the new individuals of each population

```
gerar_pontos_cruzamento(P1,P2):- gerar_pontos_cruzamento1(P1,P2). %generate crossover points

gerar_pontos_cruzamento1(P1,P2):-
        tarefas(N),
        NTemp is N+1,
        random(1,NTemp,P11),
        random(1,NTemp,P21),
        P11\==P21,!,
        ((P11<P21,!,P1=P11,P2=P21);P1=P21,P2=P11).
gerar_pontos_cruzamento1(P1,P2):-
        gerar_pontos_cruzamento1(P1,P2).
```

Generate crossover points P1 (start) e P2 (end), if, for example P1 = 2 and P2 = 4 the cut points will be between the 1º and the 2º gene, and between the 4º and the 5º gene
Note that as it is implemented, there are no cuts that only have 1 gene in the middle, because P11 is different from P21

```
cruzamento([ ],[ ]). %crossover
cruzamento([Ind*_],[Ind]).
cruzamento([Ind1*_,Ind2*_|Resto],[NInd1,NInd2|Resto1]):-
                gerar_pontos_cruzamento(P1,P2),
                prob_cruzamento(Pcruz),random(0.0,1.0,Pc),
                ((Pc =< Pcruz,!,
        cruzar(Ind1,Ind2,P1,P2,NInd1),
                  cruzar(Ind2,Ind1,P1,P2,NInd2))
                ;
                (NInd1=Ind1,NInd2=Ind2)),
                cruzamento(Resto,Resto1).

preencheh([ ],[ ]).

preencheh([_|R1],[h|R2]):-
                preencheh(R1,R2).
```

Crossover is attempted on successive individuals 2 to 2 of the population, which may be a limitation

To find out if the crossing takes place, a random number between 0 and 1 is generated and compared with the parameterized crossing probability, if it is lower, the crossing is made.

Auxiliary predicates for order crossover crossing, which is suitable for task sequencing

```
sublista(L1,I1,I2,L):-I1 < I2,!,
                sublista1(L1,I1,I2,L).

sublista(L1,I1,I2,L):-sublista1(L1,I2,I1,L).

sublista1([X|R1],1,1,[X|H]):-!, preencheh(R1,H).

sublista1([X|R1],1,N2,[X|R2]):-!,N3 is N2 - 1,
                sublista1(R1,1,N3,R2).

sublista1([_|R1],N1,N2,[h|R2]):-N3 is N1 - 1,
                                N4 is N2 - 1,
                                sublista1(R1,N3,N4,R2).

rotate_right(L,K,L1):- tarefas(N),
                T is N - K,
                rr(T,L,L1).

rr(0,L,L):-!.

rr(N,[X|R],R2):- N1 is N - 1,
                append(R,[X],R1),
                rr(N1,R1,R2).
```

Auxiliary predicates for order crossover suitable for task sequencing

```
elimina([],_,[]):-!.

elimina([X|R1],L,[X|R2]):- not(member(X,L)),!,
                           elimina(R1,L,R2).

elimina([_|R1],L,R2):-
        elimina(R1,L,R2).

insere([],L,_,L):-!.
insere([X|R],L,N,L2):-
        tarefas(T),
        ((N>T,!,N1 is N mod T);N1 = N),
        insere1(X,N1,L,L1),
        N2 is N + 1,
        insere(R,L1,N2,L2).


insere1(X,1,L,[X|L]):-!.
insere1(X,N,[Y|L],[Y|L1]):-
        N1 is N-1,
        insere1(X,N1,L,L1).
```

Auxiliary predicates for order crossover suitable for task sequencing

Auxiliary predicates for order crossover suitable for task sequencing

```
cruzar(Ind1,Ind2,P1,P2,NInd11):-
        sublista(Ind1,P1,P2,Sub1),
        tarefas(NumT),
        R is NumT-P2,
        rotate_right(Ind2,R,Ind21),
        elimina(Ind21,Sub1,Sub2),
        P3 is P2 + 1,
        insere(Sub2,Sub1,P3,NInd1),
        eliminah(NInd1,NInd11).


eliminah([],[]).

eliminah([h|R1],R2):-!,
        eliminah(R1,R2).

eliminah([X|R1],[X|R2]):-
        eliminah(R1,R2).
```

```
mutacao([],[]). %mutation
mutacao([Ind|Rest],[NInd|Rest1]):-
        prob_mutacao(Pmut),
        random(0.0,1.0,Pm),
        ((Pm < Pmut,!,mutacao1(Ind,NInd));NInd = Ind),
        mutacao(Rest,Rest1).

mutacao1(Ind,NInd):-
        gerar_pontos_cruzamento(P1,P2),
        mutacao22(Ind,P1,P2,NInd).

mutacao22([G1|Ind],1,P2,[G2|NInd]):-
        !, P21 is P2-1,
        mutacao23(G1,P21,Ind,G2,NInd).
mutacao22([G|Ind],P1,P2,[G|NInd]):-
        P11 is P1-1, P21 is P2-1,
        mutacao22(Ind,P11,P21,NInd).

mutacao23(G1,1,[G2|Ind],G2,[G1|Ind]):-!.
mutacao23(G1,P,[G|Ind],G2,[G|NInd]):-
        P1 is P-1,
        mutacao23(G1,P1,Ind,G2,NInd).
```

The mutation is attempted on each individual in the population

To find out if the mutation takes place, a random number between 0 and 1 is generated and compared with the probability of a parameterized mutation, if lower, the mutation is performed.

Running the Genetic Algorithm

?- gera.

Number of new generations: 6.

Population size: |: 8.

Crossover probability(%):|: 50.

Mutation Probability(%):|: 25.

Pop=[[t5,t3,t2,t1,t4],[t4,t1,t2,t5,t3],[t1,t2,t3,t5,t4],[t2,t1,t5,t4,t3],[t5,t1,t2,t4,t3],[t4,t3,t2,t5,t1],[t3,t4,t1,t2,t5],[t4,t5,t2,t1,t3]]

PopAv=[[t5,t3,t2,t1,t4]*23,[t4,t1,t2,t5,t3]*24,[t1,t2,t3,t5,t4]*16,[t2,t1,t5,t4,t3]*16,[t5,t1,t2,t4,t3]*25,[t4,t3,t2,t5,t1]*20,[t3,t4,t1,t2,t5]*29,[t4,t5,t2,t1,t3]*29]

Geração 0:

[[t1,t2,t3,t5,t4]*16,[t2,t1,t5,t4,t3]*16,[t4,t3,t2,t5,t1]*20,[t5,t3,t2,t1,t4]*23,[t4,t1,t2,t5,t3]*24,[t5,t1,t2,t4,t3]*25,[t3,t4,t1,t2,t5]*29,[t4,t5,t2,t1,t3]*29]

Geração 1:

[[t1,t2,t3,t5,t4]*16,[t2,t1,t5,t4,t3]*16,[t4,t2,t1,t5,t3]*16,[t2,t1,t5,t4,t3]*16,[t2,t3,t5,t1,t4]*17,[t4,t3,t2,t5,t1]*20,[t3,t4,t1,t2,t5]*29,[t4,t5,t2,t1,t3]*29]

Geração 2:

[[t2,t1,t4,t5,t3]*13,[t1,t2,t3,t5,t4]*16,[t2,t1,t5,t4,t3]*16,[t4,t2,t1,t5,t3]*16,[t2,t3,t5,t1,t4]*17,[t4,t3,t2,t5,t1]*20,[t4,t1,t2,t5,t3]*24,[t3,t4,t1,t2,t5]*29]

Geração 3:

[[t4,t2,t3,t5,t1]*14,[t2,t4,t1,t5,t3]*16,[t2,t3,t5,t1,t4]*17,[t4,t3,t2,t5,t1]*20,[t4,t1,t2,t5,t3]*24,[t5,t1,t2,t4,t3]*25,[t3,t4,t1,t2,t5]*29,[t3,t1,t4,t5,t2]*38]

Geração 4:

[[t1,t2,t4,t3,t5]*10,[t2,t4,t1,t3,t5]*14,[t1,t2,t5,t4,t3]*15,[t2,t3,t5,t1,t4]*17,[t3,t2,t1,t5,t4]*18,[t4,t3,t2,t5,t1]*20,[t5,t1,t3,t4,t2]*36,[t3,t1,t4,t5,t2]*38]

Geração 5:

[[t2,t4,t1,t3,t5]*14,[t2,t3,t5,t4,t1]*14,[t1,t2,t5,t4,t3]*15,[t4,t3,t2,t5,t1]*20,[t1,t3,t4,t2,t5]*28,[t5,t1,t3,t2,t4]*30,[t3,t1,t4,t5,t2]*38,[t3,t4,t1,t5,t2]*39]

Geração 6:

[[t1,t3,t2,t4,t5]*13,[t2,t4,t1,t3,t5]*14,[t2,t3,t5,t4,t1]*14,[t5,t2,t1,t4,t3]*17,[t4,t5,t2,t3,t1]*26,[t5,t1,t4,t2,t3]*34,[t3,t1,t4,t5,t2]*38,[t3,t4,t1,t5,t2]*39]

true .

ALGAV Sprint C has two directions concerning the Genetic Algorithm (GA)

- Improve the given GA for allowing better solutions
- Adapt the GA to the problem in hands (planning/sequencing deliveries), considering an adequate evaluation of the population elements and considering several trucks
- Additionally, for teams of 5 or more students: from a solution obtained by the GA, allow to obtain dynamic changes (change the load to deliver to the warehouse, delete a delivery, include a new delivery), we can start with the solution obtained by the GA before the dynamic change adapting for the initial population to run the GA again

Improving GA

- Ensure that at least the best individual between the current population and the new population passes to the next population
- Avoid that the crossover sequence always takes place between 1st and 2nd chromosome of the population, then between 3rd and 4th, 5th and 6th
- Apply a selection method that is not purely elitist, giving some likelihood that an individual with the worst assessment can pass to the next generation, even though they are not among the PS (population size) best evaluated from the previous population and their descendants
- The stopping condition may be different (consider other possibilities such as time, solution stabilization, etc.)

Keep Best individuals for the next generation

- The GA provided replaces the current population, pair by pair, with their descendants and there is no guarantee that the best individual will pass to the next generation
- If it is intended that the best, or some of the best, individuals go to the next generation then just joint in the same list the elements of the previous population with their descendants
- Subsequently, the selection of the remaining individuals from the list can be done by some method that is not purely elitist, we will suggest a method

Avoid that the crossover sequence always takes place between 1st and 2nd chromosome of the population, then between 3rd and 4th, 5th and 6th

- In the GA provided, the crossover sequence always takes place between 1st and 2nd chromosome of the population, then between 3rd and 4th, 5th and 6th
- A simple way to avoid the problem is to perform a random permutation between the elements of the list before the crossover
- How to make a random permutation between individuals of a population?
  - Use random_permutation/2 from SWI Prolog

```
?-
random_permutation([[t2,t5,t4,t3,t1]*11,[t2,t1,t4,t5,t3]*13,[t2,t4,t3,t5,t
1]*14,[t2,t5,t3,t4,t1]*14,[t2,t3,t4,t1,t5]*15,[t2,t1,t5,t4,t3]*16,[t4,t3,t1,t2
,t5]*29,[t1,t4,t5,t2,t3]*34],LRP),nl,write('LRP='),write(LRP),nl.


LRP=[[t2,t5,t4,t3,t1]*11,[t4,t3,t1,t2,t5]*29,[t2,t5,t3,t4,t1]*14,[t1,t4,t5,t2
,t3]*34,[t2,t4,t3,t5,t1]*14,[t2,t1,t5,t4,t3]*16,[t2,t3,t4,t1,t5]*15,[t2,t1,t4,t
5,t3]*13]
```

Apply a selection method that is not purely elitist

- Join the N individuals of the current population with their descendants obtained by crossing and mutation and remove repeated individuals (let's assume that we are left with T individuals)
- Order them (ascending order because the objective is a minimization) according to the evaluation of each one and choose the first P (P greater than or equal to 1 to ensure that the best P passes, values like 20% or 30% of N are suitable)
- Remove these elements from the list (go to the next generation) and create a new list with the remaining T-P individuals associating the product of the evaluation to each one by a random number generated between 0 and 1 and then sort the individuals of that new list in ascending order according to that product. Pass the first N-P elements from that list to the next generation. Note that they should only keep the associated evaluation value (and not the product of the evaluation value by the number generated randomly between 0 and 1)

Stopping Condition

- The GA runs for NG generations, but stopping condition may be different
- It can depend on a time, for example assuming a relative time for which you will run or assuming an absolute time at the end of which you will end, in the end the best solution is presented
- You can run until you reach a value with a specific rating, for example run until you find a solution with a rating of a specific value or less
- Bear in mind that the indicated value may be less than the optimal solution of the problem, so another way of ending should be allowed to avoid an infinite cycle
- You can run until the population stabilizes
  - Stabilizing means that the population for G generations did not change
  - Population stabilization is easier to achieve if the selection criterion is purely elitist
- Note that the fact that a population remains the same in two or more generations does not mean that it has stabilized
  - Crossings may be different in the following steps
  - The mutations may not have occurred and may then occur
- It is also necessary to provide a way to end the GA if stability is not achieved
- Implement the termination considering at least one of the following two possibilities (until reaching a solution with a cost below C, until the population stabilizes in G consecutive generations). In both cases it is better to have a termination dependent of time as well.

Adapt the GA to the problema of delivery sequencing for 1 truck

- Sequencing deliveries to warehouses is quite similar to sequencing tasks to machines
- The only thing that changes is the evaluation of the solution, that in the illustrated case is the sum of weighted delays and in the case of deliveries' sequencing has been done in the previous sprint, considering different types of time (movement from one warehouse to another, unloading the delivery, charging the batteries and charging at the middle of the trajectories)
- We just need to change the part of evaluation of population individuals

Include good solutions in the innitial populations

- Use one or more heuristics of the previous sprint to generate the initial population
- Remember our 3 heuristics:
  - The next warehouse to be visited is the one not visited more close of for witch we arrive more rapidly
  - The next warehouse to be visited is the one in which we can deliver a larger load, from those not visited
  - Na heuristic combining aspects from the previous two heuristics
- Include one solution obtained by a heuristic can be better than the generation a random individual

And how to adapt to several trucks?

- Reasons for a company to have several trucks?
  - The sum of loads to deliver in one day may be more the the load capacity of the truck (more than 4300 kg)
  - Allow the deliveries to be done more rapidly
  - Trucks have operations of maintainance and faults may happen
  - A better use of the drivers of the company

- One solution may be to have an algorithm to divide the deliveries by the trucks achieving a certain balance (of the transported load and of the number of deliveries) and then cal the GA several times, one for each truck
  - One possibility is to sort at the decreasing mode of the delivery loads, assigning the largest for the first truck, the second largest to the second truck, the third larger for the third truck, then the fourth larger for the third truck again, the fifth for the second truck, the sixth to the first truck. This solution tries to balance the number of loads and the total load of the trucks

And how to adapt to several trucks?

- Another solution is to use a geographical distribution, for example aggregating deliveries for the North of Matosinhos, for the South of Matosinhos, or for the litoral cities, or for the interior. There are some algorithms adequated to create clusters when the number of K clusters is defined. An exemple is the K-Means algorithm. We suggest that each warehouse is tagged with 2 tags (from north, south, interior and litoral). After that the trucks will be assigned to one or two tags. Then the deliveries can go from one truck to another.
- Note that several deliveries to the same warehouse are possible, for exemple if the sum of two loads is more than the load capacity of the truck
- Remember that it is not possible to overcome the load capacity of the truck (4300 kg in our exemple)

And how to adapt to several trucks?

- The suggested is to use the GA with a sequence with all deliveries, for example, from genes 1 to 5 to truck 1, from gene 6 to 11 to truck 2, and from 12 til the end to truck 3. Note that the evaluation function needs to consider that (when we enter in the genes of a new truck initial time needs to be placed to 0 again). And from the several trucks what is required is the larger time of all the trucks (when all deliveries are complete with the trucks returning to Matosinhos)
- This solution will allow all the loads to be exchanged among trucks, allowing better solutions compared to previously assigned deliveries to each truck. Remember again the limit of the load capacity of the trucks

Just for groups of 5 or more students

- For groups with 5 or more students, from a solution obtained from the GA it is envisaged to be able to allow dynamic changes (change in the load of the delivery for a warehouse, removing a certain delivery, including a new delivery)
  - From the solution obtained by the GA before the change we can adapt that solution to the change, considering this adaptation as one of the initial elements of the population, and then we run the GA again

Relatório

Report of Sprint C ALGAV (weeks 12 to 15)

What it will contains:

- **title**

-**identification** of the class, group number and students (with the number and complete name)

1) Introduction including the identification of the achieved goals in Sprint C

2) Creation of the initial population of the Genetic Algorithm (GA) considering 2 individuals (sequence of the deliveries) generated by different heuristics able to create good solutions according the previous sprint. If the generated individuals are the same you can transform one of them with a mutation (e.g. exchange two consecutive genes). Other elements from the population can be randomly generated. Notice that repeated individuals in the population must be avoided.

Relatório

3) Random Crossover between individuals of the population, avoiding that a sequence of crossovers is always between the first and second, trird and fourth, and so on.

4) Selection of the new generation of the population taking into account the elements of the previous population and their descendentes after crossover and mutation. Take into account that the mutation could change a descendent after crossover that could be the better element til that moment. The two better individuals of all elements (previous generation and descendents) must go to the next generation. The remaining elements to considerate for the next generation must result from Tounaments where any element/individual will have the opportunity to proceed, However individuals with better evaluation have more probability to proceed.

5) Efficacy Analysis comparing the better individual of the created GA compared with the better from the initial version of the GA (supposing the same number of generations). For problems with dimension 6 to 10 or 11 compare the obtained solution by the GA with the optimal solution from sprint B. Express the comparison with a table with 6 columns (number of deliveries – from 6 to 10 or 11, time for deliveries from the optimal solution; time for deliveries for the better solution obtained by the modified GA; time for deliveries for the better solution obtained by the original GA; medium value of the time for deliveries for the last generation of the modified GA; medium value of the time for deliveries for the last generation of the original GA.

6) Parametrization of the ending condition of the AG according two conditions: number of generations **and** { obtaining na individual with evaluation lower or equal to a certain value **or** stabilization of the population}.

Relatório

7) Use of the GA to handle several trucks, representing in the same cromossome the deliveries of the several trucks. At least two elements of the initial population must be generated by appplying a balancing method (numer of deliveries and loads; number of deliveries and geographical aspects). Take care to avoid unfeasible solutions (with more load than the truck capacity). We suggest to increase the crossover rate

8) Study of methods of Machine Learning / Copmuter Vision / Intelligent Robotics (just one) applied to the problem of goods distribution and/or use of electric vehicles

9) Conclusions

References (scientific papers, books, links)

Groups with 5 or more students must include a section for the additional request (dynamic changes).

The report must not be a copy of the developed code, but can/should incorporate one or another excerpt to help with the explanation

The report should not only say what the group did, but how how it was done, what options were taken

It makes no sense to explain the Genetic Algorithm provided, but it does make sense to explain the changes made (how it was extracted the knowledge from the initial data, what heuristics were used to generate the 2 good individuals from the initial population; how it was created the new population and ran the tournament ; how it was created the other termination condition, etc.)

The study of effectiveness must take into account that, due to randomness during the process, GA in different executions with the same initial data and same parameterizations will give different results. It is advisable that for each dimension of tasks (from 6 to 10 or 11) you run the initial GA and the one you created several times, because if you run it only once, it may coincide with a “happy” or “unfortunate” execution of the GA, in this way may instead place a specific value, such as, for example, the best solution created by the GA for a given number of deliveries

Indicate the GA parameterization, how many generations, population size, crossover and mutation rates, etc. The parameters must be the same for a comparison between the original GA and the developed GA. Choose suitable values for the parameters.