Introducció

El present informe exposa els procediments i resultats d'execució de la **PAC4** del curs d'Intel·ligència Artificial Avançada. L'objectiu d'aquesta pràctica és aplicar els algorismes genètics per tal d'optimitzar la distribució d'un conjunt de mercaderies en un conjunt de contenidors. Per a la realització de la pràctica utilitzarem els mètodes proporcionats per la llibreria **pyevolve**.

Les dades a treballar són les masses de 16 paquets de mercaderies. La massa de cada paquet és, respectivament,

[20, 40, 25, 10, 30, 45, 80, 120, 110, 70, 85, 35, 60, 100, 90, 130] kg.

L'objectiu de la pràctica és distribuir aquestes masses en 5 contenidors, de manera que cada contenidor tingui una massa total no major a la seva capacitat, de 200kg, i la massa total carregada en els 5 contenidors sigui la màxima possible. Els paquets són indivisibles.

Activitat 1: Preparació de les dades

Coneixeu una mica les dades: és possible carregar tots els paquets? Quina és la càrrega màxima possible (teòricament)? Hi ha masses repetides?

En aquest primer apartat realitzarem, únicament, un anàlisi previ de les dades, per tal de familiaritzar-nos amb les mateixes de cara afrontar els apartats següents. La llista de masses proporcionada és una llista de 16 components úniques (no repetides), tal i com es pot comprovar fàcilment realitzant l'ordenació del mateix:

[10, 20, 25, 30, 35, 40, 45, 60, 70, 80, 85, 90, 100, 110, 120, 130]

La menor massa és de 10kg, i la major de 130kg, pel que tots els paquets caben en un contenidor, i no s'ha de descartar cap `a priori`. La massa total dels paquets és de 1050kg (massa mitja 65kg). La càrrega màxima possible és de 5x200kg = 1000kg, pel que es dedueix que no podrem carregar tots els paquets, i sempre haurem de descartar com a poc 50kg. La carrega teòrica màxima és, per tant, de 1000kg. Es pot comprovar que aquesta càrrega màxima teòrica és un valor de solució possible fent una assignació manual descendent de càrregues:

- **Contenidor 1**: 130kg + 70kg = 200kg
- **Contenidor 2**: 120kg + 80kg = 200kg
- **Contenidor 3**: 110kg + 90kg = 200kg
- **Contenidor 4**: 100kg + 60kg + 40kg = 200kg
- **Contenidor 5**: 85kg + 45kg + 35kg + 25kg + 10kg = 200kg
- **Resta**: 20kg + 30kg = 50kg

El codi utilitzat per a la realització dels càlculs previs s'inclou a l'arxiu *JAM_IAA_PAC4.py*.

Activitat 2: Estratègia de solució

Trieu una codificació per representar una possible solució a aquest problema. Hi ha dues estratègies:

a) Definir un vector de 16 elements, un per paquet, amb el número de contenidor assignat a cada paquet (1..5) més un valor especial (per exemple 0) pels paquets que es descartin.

Pros: molt senzill.

Cons: donarà moltes solucions il·legals (massa contenidor > 200). Això farà que l'algorisme genètic no sigui molt adient per resoldre el problema.

b) Definir un vector de 16 elements, un per paquet, amb l'ordre amb el qual es van col·locant els paquets als contenidors. Quan un paquet no hi cap a un contenidor, es col·loca al següent. Els paquets finals que no hi càpiguen quedaran descartats.

Pros: qualsevol solució és vàlida. El creuament funciona bé.

Cons: més complicat de configurar.

Tal i com indica l'enunciat existeixen, com a mínim, dues estratègies de resolució pel problema. La primera estratègia, d'assignació de números aleatoris de l'1 al 5 que representin el contenidor objectiu, es fàcilment codificable amb el paquet pyevolve: únicament s'ha d'instanciar una llista de 16 elements enters de l'1 al 5:

```
# Genome instance
genome = G1DList.G1DList(16)
genome.setParams(rangemin=1, rangemax=5)
```

La funció objectiu només haurà de descartar les masses en aquells casos en que, al sumar-la, es superi la càrrega màxima del container, assignant-la a un contenidor 0 d'elements descartats. Aquesta solució es poc eficient, ja que no es pot configurar el balanceig de la llista d'assignacions i fàcilment s'arriba a masses superiors als 200kg per contenidor provocant moltes masses descartades mentre d'altres contenidors queden poc carregats.

La segona estratègia, molt més eficient, consisteix en assignar els paquets als contenidors segons la seva posició dins de la llista, i treballar en l'evolució d'aquesta ordenació fins que aconseguim acumular paquets de 200kg (tal i com hem fet manualment a l'apartat anterior) i deixar en la última/últimes posicions elements a descartar el pes dels quals sigui exactament 50kg. Com indica l'enunciat, aquesta alternativa genera alguns requeriments de configuració de l'algorisme genètic de pyevolve que assegurin:

- Que el llistat original que recull l'algorisme és el llistat de masses de l'enunciat
- Que els nous individus es generen amb mutacions tipus SWAP, és a dir, que es generen en les successives generacions individus únicament mitjançant canvis de posició aleatoris però no s'alteren els valors del llistat original de pesos
- Que la recombinació és de tipus EDGE, és a dir, per herència d'elements comuns a ambdós progenitors sense que es repeteixin valors

En els apartats següents utilitzarem aquesta segona estratègia. La configuració de pyevolve es descriu en el següent apartat, mentre que a l'apartat 4 s'exposen els resultats amb diverses configuracions.

Activitat 3: Implementació

Definiu una funció objectiu i configureu l'algorisme genètic de la llibreria pyevolve segons la representació de les solucions triada a l'activitat anterior. L'objectiu ha de ser maximitzar la càrrega útil (o minimitzar la massa descartada).

El següent codi, recollit a l'arxiu *JAM_IAA_PAC4.py* descriu la implementació de l'algorisme genètic per a la solució del problema plantejat. La funció objectiu avalua la quantitat de massa carregada (una maximització, per tant), realitzant l'assignació als contenidors en funció de l'ordre d'aparició dels paquets en la llista. L'últim container recull els elements descartats. Els punts següents descriuen la configuració de l'algorisme per a la solució del problema amb les consideracions exposades en l'apartat anterior. Cal notar que s'ha configurat l'emmagatzematge en una base de dades (arxiu *.csv*) dels resultats per tal d'adjuntar-los a la pràctica i realitzar algunes estadístiques sobre el comportament de l'algorisme. Addicionalment, s'ha configurat la terminació de l'algorisme en l'arribada a la puntuació perfecta (1000kg), tot i que s'ha deixat comentat per tal d'avaluar els resultats amb un nombre fix d'iteracions i diferents mides de població.

```
# Target function: maximize the weight in the first 5 containers
def eval_func(ind):
   score = 0.0
   package = [0 for x in range(6)] # 5 containers + the non included weig
hts
   i = 0
   for x in ind:
       if (package[i] + x <= 200 or i == 5):
           package[i] = package[i] + x
       else:
           i = i + 1
           package[i] = package[i] + x
   for j in range(5):
       score = score + package[j]
   return score

packages = [20, 40, 25, 10, 30, 45, 80, 120, 110, 70, 85, 35, 60, 100, 90,
 130]

def nonRepeatInitializer(genome, **args):
    genome.clearList()
    random.shuffle(packages)
    [genome.append(i) for i in packages]

# Genome instance
genome = G1DList.G1DList(len(packages))

# Change the initializator to custom values
genome.initializator.set(nonRepeatInitializer)

# Change the mutator to SWAP Mutator
genome.mutator.set(Mutators.G1DListMutatorSwap)

# The evaluator function (objective function)
genome.evaluator.set(eval_func)

# Change crossover to EDGE crossover
genome.crossover.set(Crossovers.G1DListCrossoverEdge)
```

```
# Best raw score
# genome.setParams(bestrawscore=1000)

# Genetic Algorithm Instance
ga = GSimpleGA.GSimpleGA(genome)
ga.selector.set(Selectors.GRouletteWheel)
#ga.terminationCriteria.set(GSimpleGA.RawScoreCriteria)
ga.setGenerations(10000)
ga.setPopulationSize(100)

# Run the database
csv_adapter = DBAdapters.DBFileCSV(filename="pyEvolve.csv", identify="01",
                        frequency = 1, reset = True)
ga.setDBAdapter(csv_adapter)

k = 10

ga.evolve(k)

data=numpy.genfromtxt('pyEvolve.csv',delimiter=';')
generation = data[:,1]

maxData = data[:,9]

plt.plot(generation,maxData,marker='o',alpha=.5,label='Weight')
plt.title('Max. Weight')
plt.xlabel('Generation')
plt.ylabel('Max Fitness Raw')
plt.ylim(900, 1000)
```

Activitat 4: Resultats

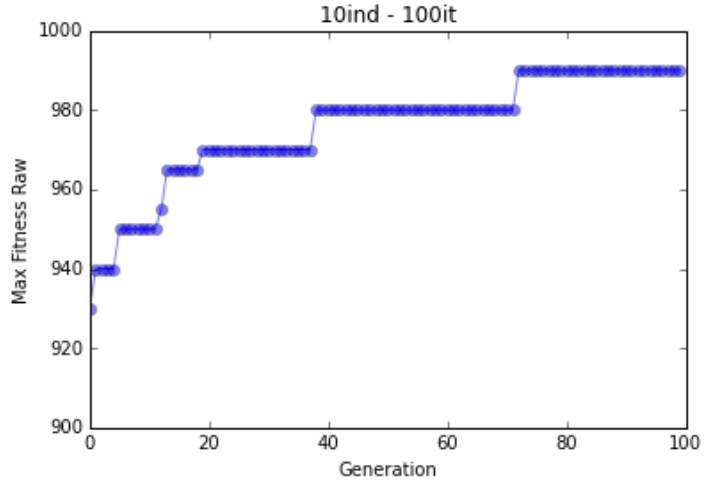
Apliqueu l'algorisme genètic per resoldre aquest problema i trobar-ne una solució. Proveu amb diferent nombre d'iteracions i tamanys de població (al vostre criteri) i estudieu la càrrega total assolida. Trobeu el màxim teòric?

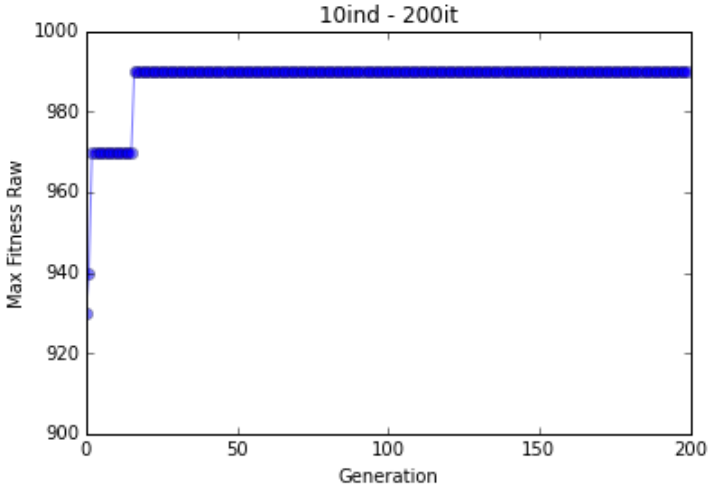
Generacions	Població	Temps	Màxima carrega assolida
100	**100**	**1,30**	**990**
200	**100**	**2,40**	**990**
500	**100**	**6**	**1000**
100	**500**	**7,8**	**990**
200	**500**	**15,30**	**995**
500	**500**	**39,50**	**1000**

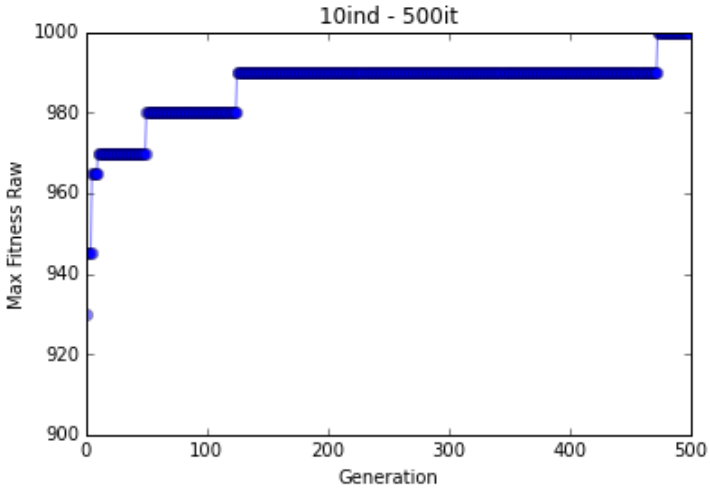
Els següents resultats corresponen a la solució fins a trobar màxim score:

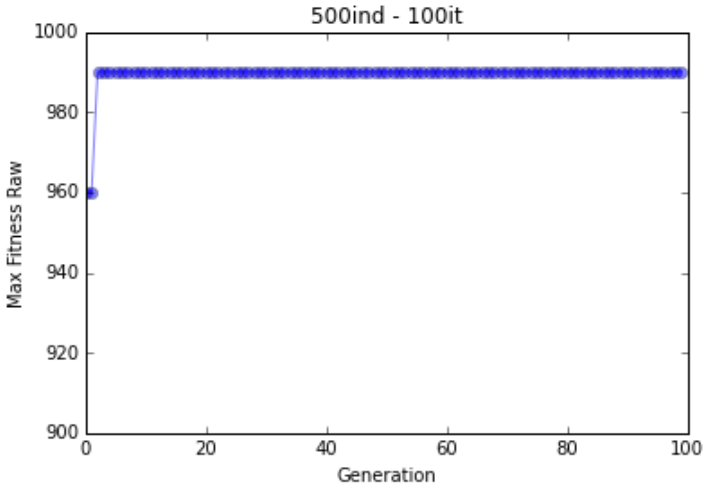
644	**100**	**9,7**	**1000**
552	**200**	**16**	**1000**
689	**500**	**53**	**1000**

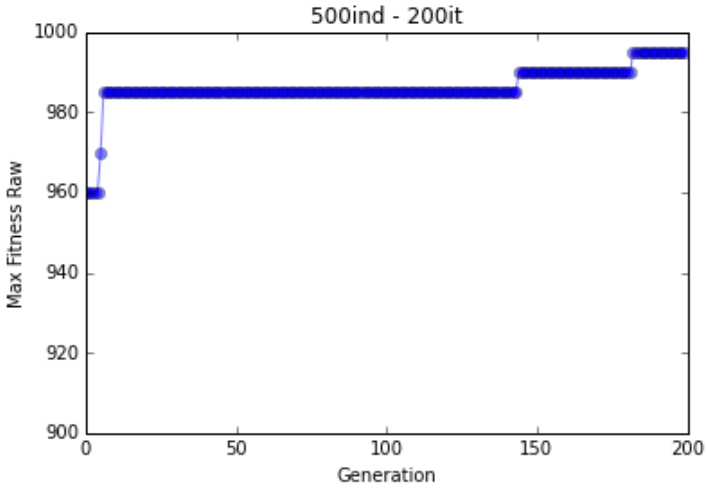
Les gràfiques següents mostren els resultats de l'execució:

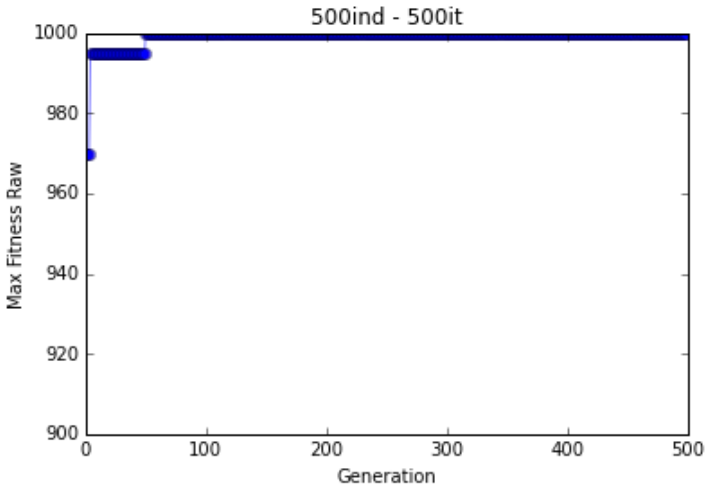


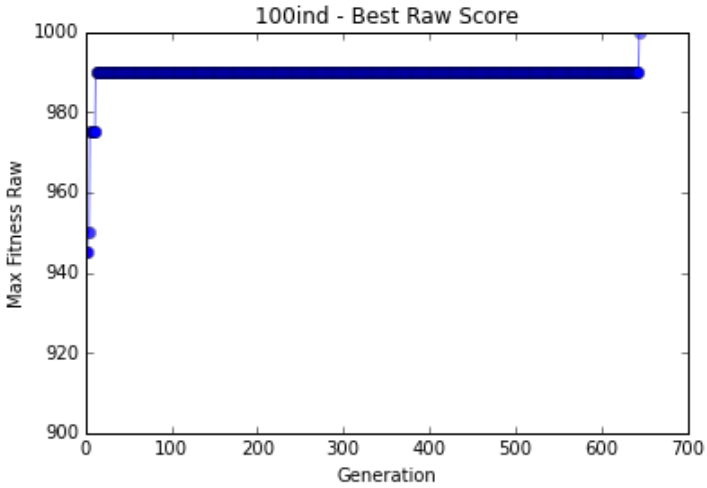


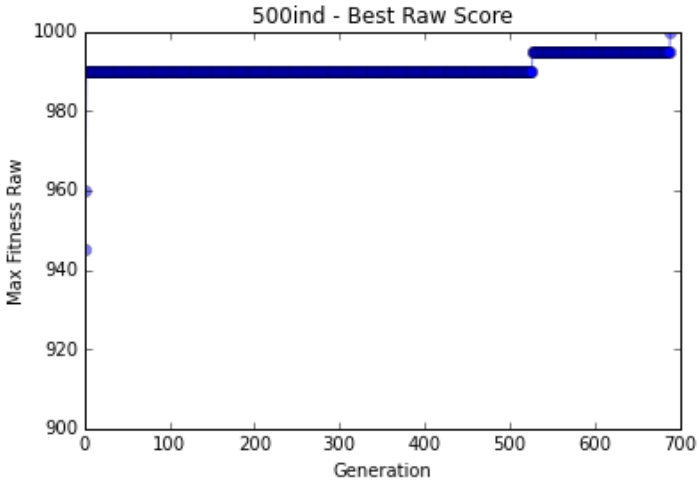



500ind - 100it
Max Fitness Raw
Generation
1000
980
960
940
920
900
0
20
40
60
80
100


500ind - 200it
Max Fitness Raw
Generation
1000
980
960
940
920
900
0
50
100
150
200


500ind - 500it
Max Fitness Raw
Generation
1000
980
960
940
920
900
0
100
200
300
400
500





Activitat 5:

Amb la codificació b), com ho faríeu si tinguessiu masses repetides a la llista de paquets?

Atès que el creuament tipus EDGE no permet valors repetits, la implementació d'aquest mètode de combinació en cas que es presentessin masses repetides requeriria adoptar, per exemple, la següent estratègia de resolució:

- Generar un vector de 16 posicions del 0 al 15 (*[x for x in range(16)]*). Aquest vector substituirà, en l'assignació, al vector original
- Generar una funció d'avaluació tal que l'assignació es realitzi en funció de l'ordre del vector generat en el punt anterior, és a dir, si el vector fos:

  [4, 2, 15, 1, 0, 6, ...]

  La primera massa a carregar seria la que ocupa la quarta posició del vector original (10, en el cas de l'enunciat). Després la que ocupa la segona posició, etc. Quan es superi la capacitat de càrrega del contenidor, es passarà al següent contenidor, tal i com hem fet en la resolució
- Avaluar la carrega total amb el mateix criteri que s'ha utilitzat a la pràctica

Òbviament, també cap la substitució de l'operador de creuament per un que permeti la combinació de dos llistes amb valors duplicats, com un creuament per 2 punts.