Práctica 2. Algoritmos evolutivos- QAP

María Victoria Santiago Alcalá

27-01-2017

Este documento se corresponde a la memoria explicativa de la práctica 2 de la asignatura Inteligencia Computacional del máster de Ingeniería Informática que ofrece la Unicersidad de Granada.

Contents

1 Introducción

En esta memoria se recoge todo el contenido relacionado con la realizacion de esta práctica cuyo objetivo consiste en la comprensión del funcionamiento de los algoritmos evolutivos centrandonos concretamente en los algoritmos geneticos.

Se comienza con la breve descripción de lso algoritmos geneticos que se han usado con el fin de resolver el problema de asociación cuadrática.

A continuación se mostrará las implementaciones realizadas con el analisis de los resultados obtenidos con su correspondiente representación.

2 Breve descripción

La práctica que se ha realizado consiste en resolver un problema de asignación cuadrática el cual tiene por objetivo construir x fabricas o instalaciones en x lugares los cuales mminimicen el coste de transporte de materiales. Una vez se hayan determinado las fábricas se ha de transportar todos los materiales teniendo en cuenta el coste que supone llevar el material de una instalación a otra dependiendo de su ubicación.

Como podemos apreciar, una solucion al problema es el viajante de comercio. La función coste que nos queda es la función de minimizar que se han explicado en los apuntes de la asignatura.

Las soluciones se ha obtenido mediante la implementación de tres algoritmos genéticos: Estándar, Baldwiniano y Lamarckiano los cuales se van a explicar de forma más detallada en los siguientes puntos.

Los algoritmos implementados son versiones muy básicas y simples por lo que los resultados obtenidos no llegan a ser los más óptimos. Para haber obtenido mejor resultados se hubiese necesitado mayor tiempo para determinar el valor de los parámetros y optimizar las funciones que intervienen en los algoritmos.

2.1 Parametros

Los parámetros a modificar durante las evaluaciones son las siguientes:

Individuos en el torneo:
Se ha determinado que los individuos del torneo van a tener como valor 5 ya ue al ser un numero pequeño con el fin de obtener individuos que no sean muy parecidos en la población.

Probabilidad de mutación del individuo y mutación del gen:
La mutación se realiza con el fin de obtener una población más variada en el individuo. Por otro lado, con la mutación en el gen se hace pequeña puesto que pude darnos individuos muy distintos a los originales.

Tamaño de población:

A más población implica abarcar un mayor tamaño, el mejor resultado se ha obtenido al establacer este parametro en 100. Como conclusión referente a este parámetro tenemos que a mas generaciones y mayor número de población, obtenemos individuos muy parecidos, también se observa que a más población mayor es el tiempo que se tarda en la ejecución.

Número de generaciones:
El numero de generaciones se ha ido modificado desde 100 hasta 1000 obteniendo el mejor resultado en las 200 generaciones.

2.2 Implementacion

Para la realización del proyecto me he decantado por usar tres algoritmos genéticos los cuales se van a explicar a continuación, pero en primer lugar se va a comenzar describiendo la representación de la solución la cual es por medio de una simple permutación y todos los operadores basicos que se han realizado ya que son metodos comunes a todos los algoritmos implementados.

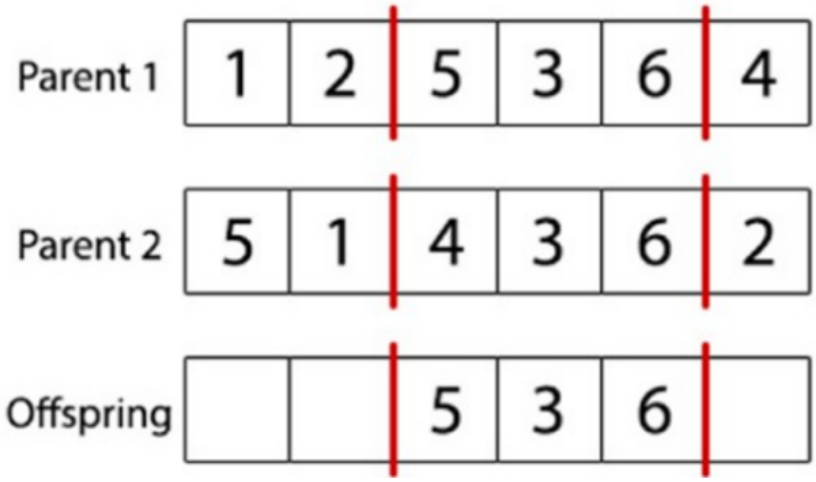
2.2.1 Operador de Selección

La selección se realiza definiendo los padres de la nueva generación de individuos todo ello usando el método de elección por torneo. Se hace una elección aleatoria de los padres y seguidamente se hace el torneo. Se eligen los de mejor coste. Seguidamente se crean los hijos (de clase Individuo) donde se va a usar para su elección el método de cruce.

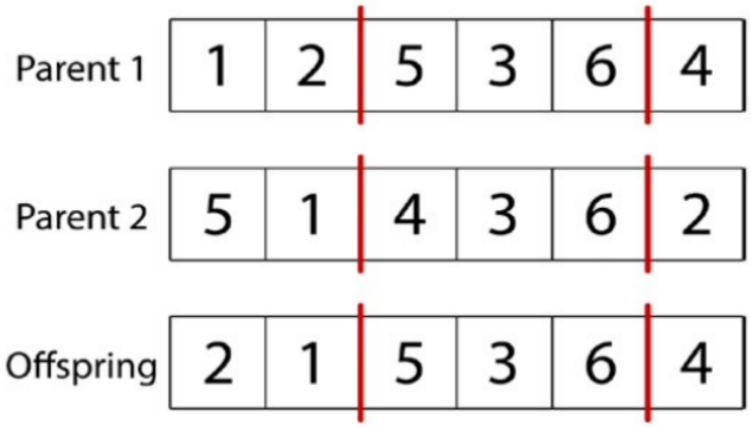
Finalmente se repite el proceso hasta crear todos los nuevos individuos.

2.2.2 Operador de Cruce

Consiste en realizar un cambio en el orden de los individuos padres en los hijos los cuales tienen que heredar el contenido central del cromosoma del padre, es decir, el hijo 1 por ejemplo heredará el rango central y el resto de posiciones del inicio hasta el centro por ejemplo los heredara del padre y del centro al final de la madre asi obtenemos un nuevo individuo e igual con el siguiente hijo.



Selección central de los cromosomas comunes a los padres.



Rellenando el restante.

Se elegirá un rango central de elementos de cromosoma que se conservan de los padres a sus hijos. Seguidamente el resto de elementos a los extremos de los hijos se van introduciendo con la información del otro padre rellenando las posiciones de principio a fin en el mismo orden.

2.2.3 Operador de Mutación

La mutación básicamente consiste en que se definen dos valores los cuales almacenan la probabilidad de que pueda mutar un hijo y además se define otro valor en el cual se determina la probabilidad de que mute un gen.

Si resulta que se da el caso de que un hijo debe de mutar se han de recorer

cada uno de los genes mirando si han de mutar o no, si se da el caso de que mutan, el gen de mutar se debe de cambiar aleatoriamente por otro gen.

2.3 Técnica de Optimización Local - Greedy 2-opt

Para la optimización local se ha usado el algortimo facilitado en la documentación de la práctica el cual se muestra a continuación:

S = candidato inicial con coste c (S)
do
mejor = S
for i =1.. n
for j = i +1.. n
T = S tras intercambiar i con j
if c (T) ¡ c (S)
S = T

while (S != mejor)

Este algoritmo se usa en el método de recalcular el fitness el cual se va a usar únicamente en los algoritmos baldwiniano y lamarckiano.

2.4 Algoritmo genetico estándar

Se ha implementado este algoritmo en primer lugar siendo la base de partida de los otros dos.

Consiste en ul algoritmo genético básico el cual utiliza los operadores de selección, de cruce y de mutación. Indicando el número de generaciones, el tamaño de población y demás, nos dará la solucíón. Este algoritmo finaliza cuando termina de ejecutarse el número de generaciones indicado.

2.5 Algoritmo genetico con variante baldwiniana

Como se ha mencionado anteriormente, este algoritmo parte de la base de la implentación del algoritmo básico donde además de aplicar los metodos de selección, cruce y mutación, se aplica la optimización local a cada generación. Ésta se aplica después de haber obtenido la nueva generación.Seguidamente se guarda el mejor individuo y las soluciones optimizadas obtenidas no se usan para la obtenr la siguiente generación.

2.6 Algoritmo evolutivo con variante lamarckiana

En este algoritmo también se parte de la versión estándar a la que también se le aplica la optimización local anteriormente descrita. Se aplica en cada generación una vez obtenida la nueva generación, seguidamente se guarda el mejor individuo obtenido despues de realizar esta optimización. Finalmente, a diferencia del algoritmo Baldwiniano, se usan las soluciones optimizadas para generar la siguiente generación.

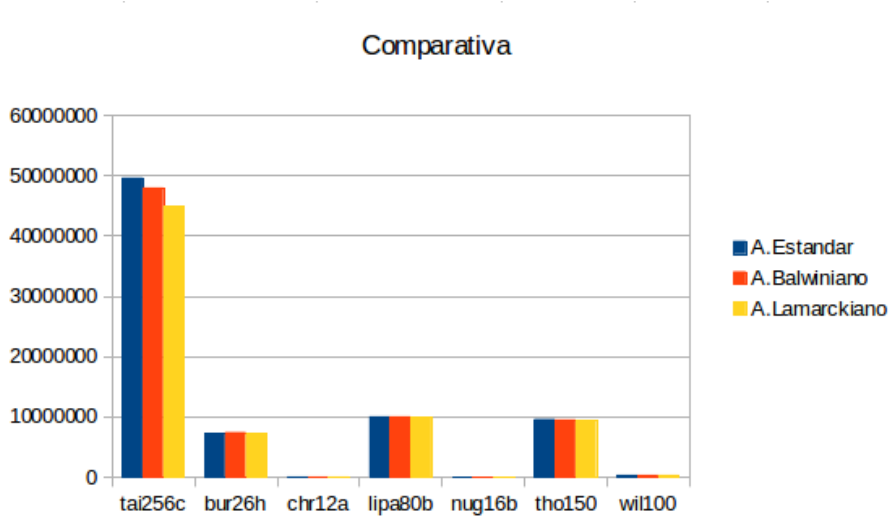
3 Resultados

En este apartado se va a comentar, tras detallar la información referente al problema y la descripción de las soluciones dadas, los resultados obtenidos en las distintas ejecuciones:

Los experimentos o pruebas realizadas nos han permitido ver como va cada uno de los parametros y la relación que hay entre ellos los cuales se han explicado en el apartado de parámetros.

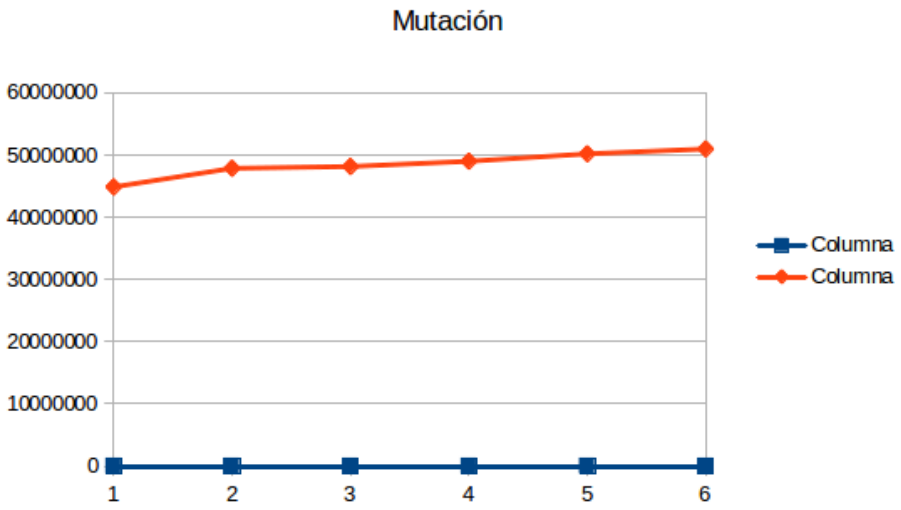
Fichero	A.Estandar	A.Balwiniano	A.Lamarckiano
tai256c	49555084	47921548	44873262
bur26h	7278393	7364255	7227007
chr12a	14178	16784	12638
lipa80b	9997450	9992339	9938918
nug16b	1436	1432	1390
tho150	9542534	9473572	9423820
wil100	294674	291430	284928

La anterior tabla muestra los resultados obtenidos tras ejecutarlo sobre distintos ficheros de datos de donde observamos que el que mejor resultados aporta en general es el algoritmo lamarckiano, seguido por el baldwiniano y el standar los cuales depende del fichero son mejor o peor que el otro.



Tras ejecutar los algoritmos conlas primeras bases de datos, se ha pasado a ejecutar la base de datos más grande que es la tai256c la cual se ha entregado los distintos resultados por la plataforma web. Se han obtenido múltiples resultados de evaluar sobre la base de datos tai256c. Principalente las obtenidas por el algoritmo básico y por la versión Baldwiniana son altas, el estándar da 49555084 y el básico da 47921548 mientras que el lamarckiano es menor.

Se ha ejecutado sobre poblaciones de tamaños: 10, 20, 40, 70, 100.
Sobre generaciones de : 100, 200, 500 y 1000.
Mutacion de individuo desde 0.3 hasta 0.75.
Mutacion de genes entre 0.01 y 0.5.



Siendo el eje X la probabilidad de mutación y el eje Y.

% Mutación	A.Lamarckiano
0,015	44873262
0,1	47874274
0,2	48153671
3	48991562
0,4	50195246
0,5	50969843

Las mejores soluciones se han obtenido tras tener los parametros tamaño de poblacio en 70 y numero de generaciones en 100, mutacion del gen en 0.01 y probabilidad de mutación del individuo en 0.7

En definitiva, estos algoritmos aportan buenas soluciones al problema de asignación tratado y si hubiera que quedarse con alguno de los dos escogería el algoritmo genético estacionario ya que los demás.

4 Conclusiones

Durante la realización de la práctica, se han ido adquiriendo conocimientos de como es el funcionamiento con las distintas herramientas y realizando distintas pruebas obteniendo finalmente unos resultados aceptables.

Se ha usado material proporcionado por la web de la asignatura, memoria de practicas y por información que se referencia en el ultimo apartado.

La implementación que se ha realizad ha sido de los algoritmos estándar, lamarckiano y baldwiniano con el fin de resolver el problema de asignación cuadrática sobre el conjunto de datos que se nos ha proporcionado.

Después de implementarlo se han ejecutado modificando los parámetros con el fin de encontrar la solución más optima para cada uno de los algoritmos y finalmente se ha comprobado y llegado a la conclusión de que el mejor algoritmo es el lamarckiano, seguido del baldwiniano y el estándar. Se ha apreciado que estos dos muestran resultados muy similares y que dependiendo de la base de datos en algunas ocasiones puede ser el estándar mejor que el baldwiniano.

El mejor resultado que se ha obtenido ha sido el que ha proporcionado el algoritmo lamarckiano. Se ha de notar también que la media de tiempo de ejecución ha oscilado entre el minuto en generaciones y poblaciones pequeñas hasta 50 minutos o una hora.

5 Referencias

http://eddyalfaro.galeon.com/geneticos.html

http://www.theprojectspot.com/tutorial-post/creating-a-genetic-algorithm-for-beginners/3

http://jgap.sourceforge.net/doc/tutorial.html

http://www.elguille.info/colabora/2007/lafbegMSI$_A lgoritmoGenetico.htm$

$http : //robologs.net/2015/09/01/como-programar-un-algoritmo-genetico-parte-ii-implementacion-en-python/$

$GeneticAlgorithmsinJavaBasics : SolveClassicalProblemslike : TheTravellingSalesmanwithGA|LeeJac$