Trabajo Individual para curso "Transporte y Tráfico"

1.- Introducción

En este trabajo se practicará de forma completa la realización de un análisis experimental completo para el problema VRP. La elección de este problema se decidió debido a dos factores:

- Ejecución más rápida
- Código más independiente de elementos externos (lo que facilita por ejemplo su ejecución desde otro sistema operativo diferente a Linux).

El código se ofrece ya completo (ver siguiente apartado) por lo que no será necesario programar nada. El objetivo por tanto es configurar de forma adecuada los algoritmos, ejecutarlos para obtener los resultados y analizarlos, plasmando los resultados en un informe (véase la sección Informe).

La organización de este documento es la siguiente:

- En la Sección Código se ofrece información de cómo obtener el código y la forma de utilizarlo.
- En la Sección Experimentos se indica que experimentos se esperan que se realicen.
- En la Sección Informe se describe qué elementos deben aparecen en el informe sobre el análisis de los resultados obtenidos.

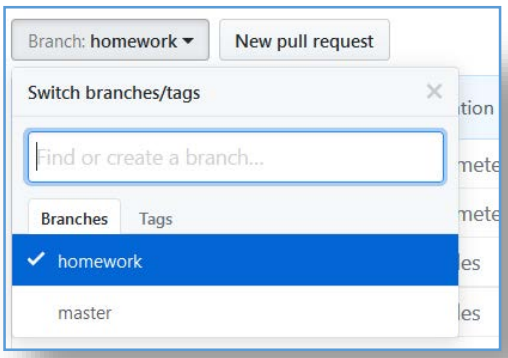
2.- Código

En este apartado mostraremos de dónde obtener el código y cómo se usa

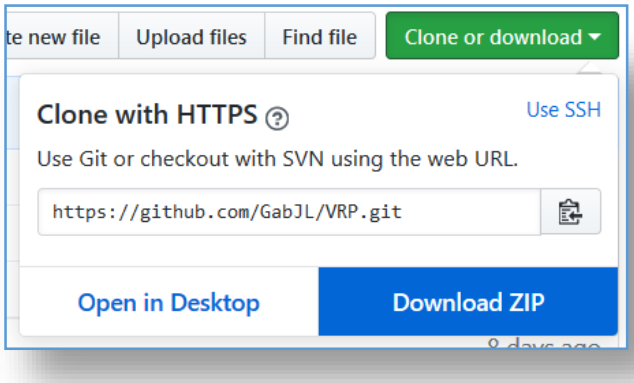
2.1.- Descarga del código

Para descarga el código acceda a la URL: https://github.com/GabJL/VRP.

Asegúrese que está marca la rama (brach) *homework*:



Descárguela pulsando el botón "*Clone or download*" y luego "*Download ZIP*":



Descomprima el fichero y obtendrá un directorio llamado “VRP-homework” donde estará el código necesario para el trabajo.

2.2.- Requisitos

Para utilizar el código necesitará un compilador de C++ que soporte el estándar del 2011 (cualquier compilador debería soportarlo). Existe una variedad bastante amplia de compiladores (e IDEs) gratuitos para todos los sistemas operativos.

Adicionalmente para los posteriores análisis de los datos necesitará algún software que permita realizar los tests estadísticos que vimos durante las sesiones presenciales. En dichas sesiones utilizamos R que es gratuito y está disponible para todos los sistemas operativos.

Aunque todo debería funcionar en Windows sin problemas, la forma de hacerlo funcionar es diferente (no mucho más complicado pero depende mucho del IDE/compilador utilizado). Por ello, quizás lo más sencillo es instalar una máquina virtual con cualquier Linux más o menos reciente y trabajar en la máquina virtual.

En Mac todo debería funcionar igual que en Linux. En el resto de apartados daré la información de cómo se hacen los pasos usando Linux/Mac.

2.3.- Compilación del código

Abra un terminal y sitúese en el directorio “VRP-homework” y ahí ejecute el comando:

```
make
```

Eso generará los ejecutables de todos los algoritmos y los copiará a la carpeta actual para ser utilizados.

2.4.- Ejecución de un algoritmo

Suponiendo que ya ha compilado el código, abra un terminal y sitúese en el directorio “VRP-homework” y ahí ejecute el comando:

```
./Algoritmo ficheroCfg
```

Donde **`Algoritmo`** puede ser **`RS`**, **`HC`**, **`VNS`** y **`ssGA`** y en **`ficheroCfg`** debe indicar el fichero donde está la configuración del algoritmo (véase siguiente apartado). Por organización recomiendo que el fichero de configuración se almacene en la carpeta **`cfg`** y su nombre tenga cierta relación con el algoritmo al que configura. Por ejemplo:

```
./HC cfg/HC1.cfg
```

Como resultado escribirá diferentes ficheros atendiendo a la configuración (véase último apartado de esta sección).

2.5.- Configuración del algoritmo

Para facilitar la parametrización de los algoritmos y los experimentos, todos los parámetros se escribirán en un fichero de configuración que será leído por el algoritmo. El formato de ese fichero son múltiples líneas con pares <atributo valor>. Si un atributo no aparece se tomará un valor por defecto y si aparece varias veces, únicamente se considerará el último. Los posibles atributos y en qué algoritmos funcionan se muestran en la Tabla 1.

Se recomienda que los ficheros de configuración se tengan en el directorio **`cfg`** y que su nombre esté relacionado con el algoritmo/experimento realizado. Además en vez de reutilizar el mismo para varias pruebas, se recomienda crear uno nuevo (copiando un antiguo y renombrándolo) por cada experimento.

Ya hay un fichero de ejemplo por cada algoritmo que puede ser usado como base para el resto de experimentos.

El atributo **experiment** influirá donde se generarán los resultados y se recomienda que empiecen por res/ (para que se generen en ese directorio) seguido por una palabra (sin espacios) relacionada con el experimento (quizás la misma con la que nombró el fichero de configuración). Por ejemplo puede ver los ficheros de configuración **HC1.cfg**, **HC2.cfg** y **HC3.cfg**.

Atributo	Dominio	Defecto	RS	HC	VNS	ssGA	Explicación	Ejemplo
runs	Natural [1,inf)	30	X	X	X	X	Número de ejecuciones independientes del algoritmo	**runs 10**
evaluations	Natural [1,inf)	100000	X	X	X	X	Número máximo de soluciones evaluadas por ejecución	**evaluacions 30000**
instance	Texto	instances/vrpnc1.txt	X	X	X	X	Nombre del fichero donde está la instancia	**instance instances/vrpnc7.txt**
experiment	Texto	res/RS	X	X	X	X	Nombre base para los guardar los experimentos (se recomienda empezar por res/ para que se almacene el dicho directorio y quede todo ordenado)	**experiment res/ssGA1-**
verbose	{0,1}	1	X	X	X	X	Indica si debe escribir cómo evoluciona el algoritmo o no.	**verbose 1**
neighbor	Insertion Inversion Interchange	Insertion		X			Operador utilizado para generar un vecino	**neighbor Interchange**
convergence	Natural [1,inf)	5			X		Número de intentos sin mejora en los que se aplica un vecindario antes de cambiar	**convergence 50**
popsize	Natural [2,inf)	10				X	Tamaño de la población del ssGA.	**popsize 50**
selection	Random BinTour	Random				X	Operador utilizado para seleccionar las soluciones que serán cruzadas/mutadas: aleatoria (Random) o torneo binario (BinTour	**selection BinTour**
mutationP	Real [0,1]	0.8				X	Probabilidad de que una solución sea mutada	**mutationP 0.0**
mutationOp	Insertion Inversion Interchange	Insertion				X	Operador de mutación	**mutationOp Interchange**
recombinationP	Real [0,1]	0.8				X	Probabilidad de que una solución sea cruzada	**recombinationP 0.0**
recombinationOp	CX OX	OX				X	Operador de recombinación	**recombinationOp CX**

Tabla 1.- Atributos para el fichero de configuración.

2.6.- Resultados de la ejecución:

Como resultado siempre ofrecerá un fichero llamado **Xg.txt** donde **X** es el nombre del experimento indicado en la configuración que tendrá varias líneas (una por ejecución independiente realizada) con el mejor valor encontrado durante la ejecución.

Adicionalmente, si en la configuración se indicó que fuese detallada (**verbose 1**) generará un fichero adicional por ejecución (**X1.txt**, **X2.txt**, **X3.txt**, …) con el valor actual en cada paso. Tenga en cuenta los siguientes aspectos:

- En RS y HC tendrá tantas líneas como evaluaciones se estableció en el fichero de configuración
- En VNS puede tener menos porque cuando converge en todos los vecindarios posibles para.
- En ssGA tendrá (evaluaciones – popsize) + 1 líneas debido que a que en la población inicial genera popsize soluciones pero solo escribe la mejor.

3.- Experimentos

Se van a realizar los siguientes experimentos:

- Análisis del mejor operador para explorar el vecindario HC: (3 pruebas)
  - Use la instancia vrpnc1.txt para estos experimentos y 100000 evaluaciones (30 ejecuciones).
  - Pruebe los tres operadores.
  - Elija el mejor operador (o si hay varios mejores – sin diferencia significativa entre ellos- elija el que obtuvo el mejor valor en alguna ejecución).
  - Para este caso debe ejecutar el algoritmo **HC** con los ficheros de configuración **HC1.cfg, HC2.cfg** y **HC3.cfg** (que tienen la configuración elegida).
- Análisis del mejor criterio de convergencia para VNS: (3 pruebas)
  - Use la instancia vrpnc1.txt para estos experimentos y 100000 evaluaciones (30 ejecuciones).
  - Pruebe los valores de convergencia 5, 50, 100
  - Elija el mejor valor (o si hay varios mejores – sin diferencia significativa entre ellos- elija el que obtuvo el mejor valor en alguna ejecución).
  - Para este caso debe ejecutar el algoritmo **VNS** con el fichero de configuración **VNS.cfg** pero debe adaptar el valor de **convergence** a los valores pedidos.
- Análisis de la mejor configuración para ssGA: (10 pruebas)
  - Use la instancia vrpnc1.txt para estos experimentos y 100000 evaluaciones (30 ejecuciones).
  - Pruebe 10 configuraciones diferente (puede cambiar los parámetros que desee, pero al menos cambie tres atributos diferentes)
  - Elija el mejor valor (o si hay varios mejores – sin diferencia significativa entre ellos- elija el que obtuvo el mejor valor en alguna ejecución).
  - Para este caso debe ejecutar el algoritmo **ssGA** con el fichero de configuración **ssGA.cfg** pero debe adaptar el valor de los atributos elegidos.
- Comparación de los mejores: (20 pruebas)
  - Ejecute las mejores configuraciones de **HC**, **VNS** y **ssGA** y además el **RS** en 5 instancias (vrpnc{2-6}.txt) con 100000 evaluaciones (30 ejecuciones).
  - Compare los resultados (véase el Informe).

4.- Informe

Como resultado de los anteriores experimentos debe crease un fichero comprimido con los siguientes ficheros y enviarse a gabriel@lcc.uma.es:

- Todos los ficheros de configuración (use uno diferente por ejecución).
- Todos los ficheros de resultados (tenga cuidado en no sobrescribir los ficheros).
- Informe (documento PDF) de análisis con el contenido que se describe a continuación.

En el informe debe añadirse:

- En los tres primeros análisis (los de buscar la mejor configuración):
  - A cada variante debe ponerle un nombre y describir sus parámetros (solo aquellos que cambien)

- Una tabla con la media/desviación estándar resultante de las 30 ejecuciones independientes. Además debe añadirse otra columna indicando el ranking del algoritmo (tendrán el mismo ranking los resultados que no sean significativamente diferentes).
- Gráfica con los boxplots
- Un breve comentario explicando los resultados

- En el último análisis (comparación entre los mejores):
  - Una tabla algoritmos x instancia (4 x 5) donde en cada celda aparezca la mejora la media de sus 30 ejecuciones. Además ese valor estará en negrita si es el mejor (o no hay diferencia significativas con el mejor) para esa instancia.
  - Un texto explicando los resultados y cuál es el mejor algoritmo y cómo lo ha decidido.
  - Una gráfica donde muestre la evolución del fitness para la instancia vrpnc5.txt (elija una ejecución representativa para cada algoritmo – tenga cuidado en ajustar los valores ya que no en todos están iguales como se indica en 2.6).
  - Texto explicativo para esa gráfica.