Una primera aproximación al estudio experimental de procedimientos de optimización

A veces un hombre desea conocer el resultado de una acción, y entonces piensa en una acción parecida y en los resultados sucesivos a que ésta dio lugar, en la suposición de que acciones semejantes se seguirán de resultados semejantes.

Thomas Hobbes. *Leviathan.*
(Ilustración: Grabado El sueño de la razón produce monstruos de Francisco de Goya)

Carlos Andrés Romano
Catedrático de Universidad
Universitat Politècnica de València
Valencia, 2022

Este texto es un extracto inspirado en el material publicado por el mismo autor en el libro “OPTIMIZACIÓN METAHEURÍSTICA PARA INGENIEROS: Aplicaciones en Organización Industrial” Ed. Delta 2017

1. Introducción

Este texto pretende guiar tus primeros pasos a la hora de evaluar el funcionamiento de cualquier método que programes para resolver problemas de optimización combinatoria relacionados con la Organización Industrial. Dada la naturaleza de estos procedimientos de optimización, siempre va a ser inevitable que, cuando los diseñes e implementes, tengas dudas de hasta qué punto estás obteniendo buenas soluciones y si tu procedimiento es mejor o peor que otros ya existentes.

Para poder responder adecuadamente a estas cuestiones debes de analizar los resultados de ejecutar tu programa usando un **método** basado en la **experimentación**. Del mismo modo que, para demostrar que cierto tipo de medicamento es eficaz contra una bacteria, es necesario realizar experimentos para determinar las concentraciones del fármaco, su frecuencia de administración, contra qué cepas de la bacteria es mejor, etc., así también nosotros tenemos que usar el método científico y realizar experimentos para profundizar en el conocimiento del comportamiento de nuestras implementaciones. Con ello trataremos de determinar (entre otras cosas) si su diseño es correcto, cuántas veces hemos de ejecutarlos, los parámetros que vamos a usar (si nuestros algoritmos tuvieran parámetros), los tipos de problemas que pueden resolver, etc…

En nuestro caso, el objeto de los experimentos no es un fenómeno físico o químico sino un constructo artificial que hemos creado nosotros. En este sentido tenemos cierta ventaja porque conocemos los detalles del fenómeno a estudiar (al menos conocemos y controlamos lo que hemos programado) y podemos interpretar los resultados que obtengamos a la luz de los experimentos. Sin embargo, también existen muchos factores que no controlamos (las decisiones de diseño a la hora de establecer la manera en cómo funciona nuestra propuesta, la aleatoriedad que aparecen en algunas estrategias de optimización, ciertas características de los problemas a resolver que hacen que algunas veces sean fáciles de optimizar y otras veces sean terriblemente complejos, nuestra habilidad en programar las instrucciones que deseamos que se ejecuten, etc.).

Por todo esto, realizar un estudio experimental de un algoritmo es una tarea compleja que exige unos conocimientos importantes en estadística (diseño de experimentos, inferencia estadística, técnicas de regresión) algunos de los cuales se van a abordar en este texto, pero otros exigen una profundización a la que no todo el mundo está dispuesto a llegar. Por ello, me contentaré con darte unas ideas sencillas para que puedas evaluar tus algoritmos de manera muy preliminar e intentaré despertar tu interés por esta apasionante fase del desarrollo de herramientas de optimización. En todo caso, al final de este texto te dejaré algunas referencias bibliográficas útiles si crees necesario profundizar más para evaluar tus desarrollos de una manera más rigurosa. Espero que todo lo que vas a leer a continuación te sea de ayuda.

2. ¿Cuál es tu objetivo?

Un método de resolución se puede analizar desde muchas perspectivas. Lo primero que podemos plantearnos es que sea capaz de producir **soluciones de alta calidad** (lo que llamamos bondad del algoritmo). Sin embargo, en términos de computación, es frecuente que incorporemos el factor **tiempo** en esta pregunta por lo que cambiamos nuestro enfoque a estudiar si nuestro algoritmo produce **soluciones de alta calidad más rápidamente que otros**. Por ello, no solamente tendremos que estudiar el valor de las soluciones sino el tiempo que nos cuesta obtenerlas.

Otros enfoques pueden ser que nuestros procedimientos **ofrezcan más soluciones de alta calidad que otros métodos** a lo largo de sucesivas ejecuciones (no solo una), estos es lo que se conoce como exactitud. También podemos pensar en que nuestros algoritmos sean robustos, esto es, que **sean poco sensible a diferencias en las características** de los problemas (exactitud de los datos o variación de los valores de los parámetros que se usan). Finalmente, muchos usuarios agradecerán que hagan un **uso eficiente** de la memoria de los ordenadores utilizados o incluso que los algoritmos planteados sean simples de implementar y comprender.

En todo caso, nosotros nos vamos a focalizar en el objetivo más común cuando hablamos de evaluar el rendimiento de métodos de optimización y estas es la bondad, esto es, encontrar una solución lo más cercana al óptimo posible (o al menos a la mejor solución conocida).

Dada la naturaleza de los problemas que abordamos, sabemos que es prácticamente imposible asegurar que una solución es la óptima para la mayoría de problemas a los que tratamos de encontrar una respuesta. Sin

embargo, hay ciertos problemas (TSP, VRP, Equilibrado de líneas, Scheduling, etc…)[1] que han sido estudiados por los investigadores desde hace más de medio siglo y para los que se han conseguido encontrar soluciones óptimas. Lamentablemente, para un problema real, concreto y particular nuestro, lo más seguro es que no sepamos nunca cuál es la solución óptima. En estos casos podemos proceder comparando nuestros resultados con la mejor solución encontrada (bien sea por nosotros o por otros). Para ello, en vez de trabajar con términos absolutos, os recomiendo trabajar con esta expresión cuando analicemos si una solución es mejor que otra:

$$RPD = \frac{Some_{sol} - Best_{sol}}{Best_{sol}}$$

Donde RPD se conoce como **Relative Percentage Deviation** (desviación porcentual relativa) y mide el % de desviación de nuestra solución ($Some_{sol}$) respecto a la mejor conocida ($Best_{sol}$).

Inicialmente, cuando no disponemos de ningún valor de $Best_{sol}$ podemos generar aleatoriamente soluciones a nuestro problema y usar la mejor como primera estimación del valor de $Best_{sol}$. Esto permite obtener una distribución de las soluciones y empezar a conocer un poco mejor las características del problema (restricciones y variables) así como determinar qué nos interesa del mismo (función objetivo).

Para terminar de entender el concepto anterior, pensemos en un determinado problema de TSP que tiene una solución óptima de 200 y supongamos que nuestro algoritmo ha encontrado una solución que vale 250. Siempre será mejor indicar que nuestro algoritmo ha obtenido un RPD = 25% peor que el óptimo que decir que hemos encontrado una solución que vale 250.

Además de lo anterior, mi recomendación es que, cuando analices el funcionamiento de herramientas de resolución de cualquier problema, te focalices en resolver las siguientes cuestiones:

- ¿Cuál de las herramientas/algoritmos/programas/métodos es mejor para resolver un cierto problema?
- Para implementar un algoritmo, ¿cuál alternativa de diseño es mejor?
- En caso que mi procedimiento tenga parámetros, ¿qué valores de los mismos son los más adecuados?
- ¿Para qué instancias[2] del problema la herramienta propuesta funciona bien (o mal)?

Todo esto se puede concretar en que, cuando analicemos nuestros algoritmos, abordemos estos tres objetivos principales:

- Objetivos de diseño: Elegir bien aquellas características que hacen que mi algoritmo funcione bien.
- Objetivos de análisis: ¿En qué instancias mi algoritmo funciona bien?
- Objetivos de comparación: En el caso que estemos comparando diferentes algoritmos, ¿cuál de ellos es el mejor y en qué condiciones.

Todos estos objetivos se pueden formular como **preguntas de investigación o hipótesis**.

Por ejemplo, un objetivo de diseño podría ser formulado como “En un Algoritmo Genético ¿Es mejor usar un operador de cruce por un punto o por dos puntos?”, “¿Un tamaño de lista tabú mayor que 3 es mejor que uno más pequeño?

Lo mismo podemos hacer para los objetivos de análisis: ¿Mi algoritmo funciona igual de bien cuando resolvernos un TSP de 20 ciudades que uno de 100?

Y, por supuesto, los objetivos de comparación: ¿Mi implementación de un algoritmo de Recocido Simulado es mejor que un algoritmo puramente aleatorio?

En el siguiente apartado, comentaré brevemente cómo deberíamos contestar rigurosamente a cualquiera de las preguntas de investigación anteriores.

[1] Por ejemplo, si buscas en google el nombre del problema (TSP, Scheduling, etc…) junto con “benchmark dataset” podrás acceder a librerías con ejemplos de esos problemas y sus soluciones óptimas (o mejores conocidas hasta la fecha)

[2] Se denomina instancia o ejemplar de un problema al conjunto de valores numéricos del problema. Por ejemplo, una instancia de un problema de TSP de nueve ciudades sería los valores concretos de la matriz de distancias. Otra instancia diferente implicaría que los valores fueran distintos.

3. Demostrar si es cierta o no una pregunta de investigación

Para responder a cualquiera de las preguntas de investigación que podemos formularnos, tendremos que ejecutar nuestro procedimiento de optimización en unas condiciones controladas. El número de veces que lo hagamos será limitado y por ello, a partir de los resultados que obtengamos (una muestra) tendremos que inferir unas conclusiones generales. Es por ello que habrá que hacer uso de la Estadística para que no cometamos errores a la hora de generalizar resultados de una muestra.

Por ejemplo, pensemos que estamos formulando una hipótesis que dice que un algoritmo A da mejores soluciones en promedio que uno B durante el mismo tiempo de cálculo ($\mu 1 = \mu 2$ donde μi representa la media de los resultados de ejecutar el algoritmo i, es lo que en estadística se llama **hipótesis nula** o H_0). Para demostrar si esta hipótesis H_0 es cierta o no, ejecutamos ambos algoritmos sobre las mismas instancias y durante el mismo tiempo. Cada vez se guarda la solución que obtienen, esto es, una muestra de todas las veces que los podríamos ejecutar.

A partir de la muestra se debería calcular lo que se denomina un estadístico de prueba (que es una expresión estadística que nos va permitir aceptar o rechazar H_0 la hipótesis nula). Cualquier análisis estadístico se basa en **calcular la probabilidad P de que el estadístico de prueba tenga un valor que sea al menos igual al valor que debería tener si H_0 fuera verdadera**.

El valor P de esta probabilidad podría ser cualquiera, pero en estadística se suele usar 0.05. De una manera más sencilla, P es la probabilidad de que estemos equivocados rechazando la hipótesis nula. Así si P es más grande que 0.05 tu riesgo en equivocarte rechazando la hipótesis nula se considera que es muy grande y NO debes rechazarla (en nuestro caso recordamos que esto significa que ambos algoritmos dan el mismo valor en media) mientras que si es menor, entonces has de rechazar la hipótesis nula porque tu riesgo de equivocarte al hacerlo es pequeño (o lo que es lo mismo, aceptar la hipótesis alternativa que es que ambos algoritmos tienen distinto valor de media).

Entender todas las técnicas estadísticas que se usan para evaluar esta probabilidad P va más allá de estos apuntes, pero, en este texto os voy a presentar la más extendida que es el Análisis de la Varianza (ANOVA). Para ello, voy a usar un ejemplo sencillo basado en analizar mediante la hoja de cálculo EXCEL y el complemento Real Statistics[3] si **existen diferencias en el promedio de los resultados de aplicar tres algoritmos** A, B y C al mismo problema usando tanto **ANOVA** (usada cuando queremos saber si existen diferencias en el promedio de una variable en dos o más grupos) y el denominado **test de Tukey** (usado cuando sí que hay diferencias y queremos saber si entre qué algoritmos las hay).

El motivo de usar un complemento como el Real Statistics es que Excel no cuenta con un gran número de funciones estadísticas de análisis y diseño de experimentos (salvo el ANOVA y pocos más) por lo que va a ser necesario siempre usar algún módulo añadido a EXCEL o usar otros paquetes estadísticos como STATGRAPHICS[4], SPSS[5] o R[6]. Yo os sugiero el paquete Real Statistics porque es gratuito y tiene una página web con ayuda sobre las diferentes funciones que incorpora. En el anexo os indico como os lo podéis descargar e instalar en vuestra EXCEL.

Hay que remarcar que la técnica ANOVA es una técnica paramétrica y se basa en suponer que la variable a analizar se distribuye de manera normal y tiene varianza idéntica (cosa que puede no suceder y por eso te remito a cualquier texto en un curso de nivel medio de estadística donde se expliquen técnicas de inferencia estadística (paramétrica o no-paramétrica) que te permitirán extraer conclusiones en el caso que no se cumplan las hipótesis anteriores.

Para empezar, vamos a suponer que ejecutamos tres algoritmos A, B y C sobre la misma instancia, sin variar sus parámetros y con el mismo tiempo de cálculo, durante 10 veces cada uno. Obtenemos los siguientes resultados:

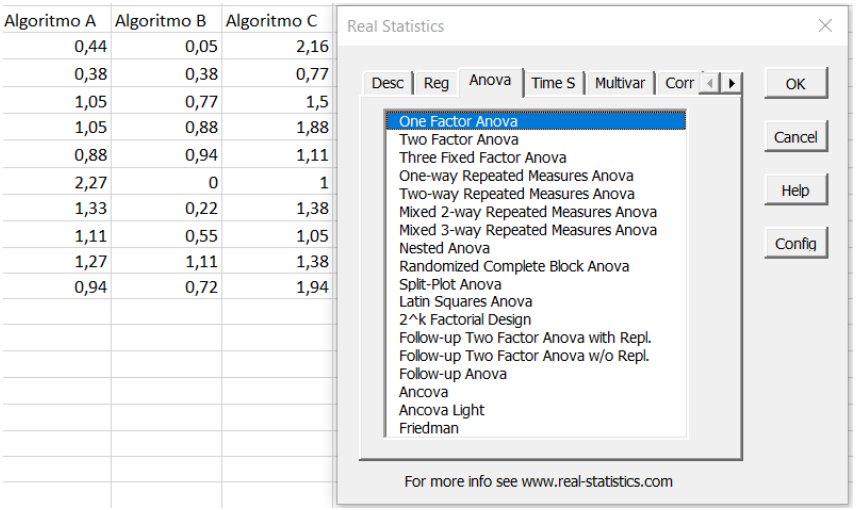
[3] Puedes descargarte este complemento de Excel de manera gratuita de https://www.real-statistics.com/
[4] https://www.statgraphics.com/
[5] https://www.ibm.com/es-es/spss
[6] https://www.r-project.org/

Algoritmo A	Algoritmo B	Algoritmo C
26	19	57
25	25	32
37	32	45
37	34	52
34	35	38
59	18	36
42	22	43
38	28	37
41	38	43
35	31	53

Lo primero que hemos de hacer para empezar a trabajar es calcular la RPD de cada ejecución de cada algoritmo. Como el mejor valor encontrado en todas las ejecuciones es 18, tenemos que calcular para cada valor la diferencia entre los valores de la tabla anterior y 18, y después dividir todo eso por 18.

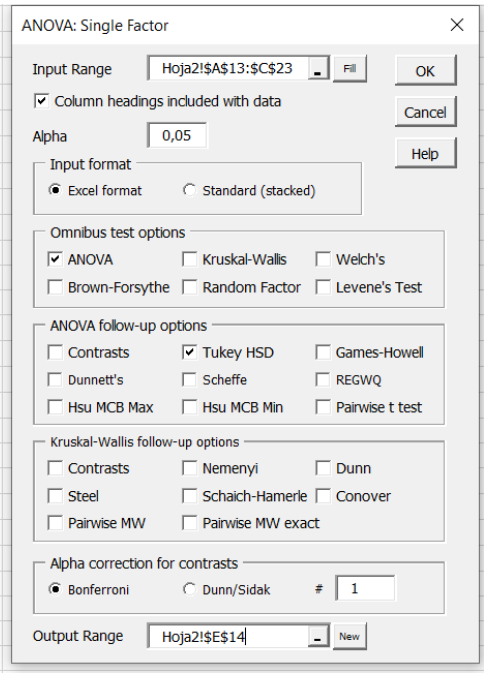
Algoritmo A	Algoritmo B	Algoritmo C
0,44	0,05	2,16
0,38	0,38	0,77
1,05	0,77	1,5
1,05	0,88	1,88
0,88	0,94	1,11
2,27	0	1
1,33	0,22	1,38
1,11	0,55	1,05
1,27	1,11	1,38
0,94	0,72	1,94

A continuación, abriremos EXCEL con el complemento Real Statistics según se indica en el anexo y accederemos a la pestaña ANOVA eligiendo la opción One Factor ANOVA:



Seleccionaremos el rango de entrada (incluyendo los títulos de las columnas) y, seleccionando el test de Tukey HSD tal y como se muestra en la figura, pulsaremos aceptar:

Algoritmo A	Algoritmo B	Algoritmo C
0,44	0,05	2,16
0,38	0,38	0,77
1,05	0,77	1,5
1,05	0,88	1,88
0,88	0,94	1,11
2,27	0	1
1,33	0,22	1,38
1,11	0,55	1,05
1,27	1,11	1,38
0,94	0,72	1,94



El resultado se muestra a continuación:

ANOVA: Single Factor								
DESCRIPTION					Alpha	0,05		
Group	*Count*	*Sum*	*Mean*	*Variance*	*SS*	*Std Err*	*Lower*	*Upper*
Algoritmo A	10	10,72	1,072	0,27510667	2,47596	0,14514016	0,77419699	1,36980301
Algoritmo B	10	5,62	0,562	0,14897333	1,34076	0,14514016	0,26419699	0,85980301
Algoritmo C	10	14,17	1,417	0,20789	1,87101	0,14514016	1,11919699	1,71480301
ANOVA								
Sources	*SS*	*df*	*MS*	*F*	*P value*	*Eta-sq*	*RMSSE*	*Omega Sq*
Between Gro	3,7005	2	1,85025	8,7832492	0,00115291	0,39416376	0,93718991	0,34162156
Within Group	5,68773	27	0,21065667					
Total	9,38823	29	0,32373207					

El valor que más nos interesa de esta tabla es el que aparece debajo del valor de **P value** (0,00115291). Este valor representa la probabilidad de equivocarnos si rechazamos que las medias de los resultados de cada algoritmo son iguales y como es menor que el umbral del 0,05 (que se toma como referencia para P) podemos asumir que la probabilidad de equivocarnos si rechazamos la hipótesis nula de la igualdad de medias es muy pequeña por lo que asumiendo ese mínimo riesgo la rechazamos. En resumen, aceptamos que hay diferencia entre las medias de la ejecución de los tres algoritmos.

Para terminar de evaluar **entre qué algoritmos** hay diferencia significativa de una manera estadísticamente rigurosa, se suele **usar el test de HSD de Tukey**. En la otra tabla que hemos generado podemos leer lo siguiente:

TUKEY HSD/KRAMER			alpha	0,05					
group	*mean*	*n*	*ss*	*df*	*q-crit*				
Algoritmo A	1,072	10	2,47596						
Algoritmo B	0,562	10	1,34076						
Algoritmo C	1,417	10	1,87101						
		30	5,68773	27	3,506				
Q TEST									
group 1	*group 2*	*mean*	*std err*	*q-stat*	*lower*	*upper*	*p-value*	*mean-crit*	*Cohen d*
Algoritmo A	Algoritmo B	0,51	0,14514016	3,51384477	0,00113859	1,01886141	0,04943882	0,50886141	1,11117528
Algoritmo A	Algoritmo C	0,345	0,14514016	2,37701264	-0,16386141	0,85386141	0,2307716	0,50886141	0,7516774
Algoritmo B	Algoritmo C	0,855	0,14514016	5,89085741	0,34613859	1,36386141	0,00080848	0,50886141	1,86285268

Aquí, lo que más nos interesan son los valores que hay en la columna p-value de la tabla Q test para cada comparación entre dos algoritmos. Vemos que son valores inferiores a 0,05 para la pareja Algoritmo A- Algoritmo B (0,04943882) y la pareja Algoritmo B-Algoritmo C (0,00080848) por lo que podemos concluir que hay diferencia entre usar el algoritmo A y el B, y también entre usar el B y el C, pero no la hay entre el Algoritmo A y el Algoritmo C. En esta tabla, bajo la columna de diferencia de medias (mean) se observa la diferencia en valor absoluto entre las medias de los resultados de cada algoritmo, se observa que el algoritmo A en media tiene una diferencia de 0,51 mayor que el algoritmo B. Del mismo modo, la diferencia de medias en valor absoluto entre el Algoritmo B y el C es de 0,855. ¿Cómo podemos saber cuál de ellos es el mejor?

Para responder a esta pregunta, observamos las medias de aplicar cada algoritmo en la tabla TUKEY HSD/KRAMER (columna mean). Se ve que es menor en el algoritmo B (0,562) que en el resto por lo que, al dar valores de la RPD más bajos en media es mejor significativamente que los otros dos (esto también se podría corroborar haciendo a partir de los datos una gráfica de caja y bigotes ya que está disponible entre las gráficas de Excel).

Como ves, este proceso es relativamente sencillo cuando queremos analizar una hipótesis de igualdad entre varios algoritmos, pero se puede convertir en muy tedioso si queremos comparar varios algoritmos u otros factores que creamos que influyen en los resultados. Para agilizar este proceso y asumiendo que solo hacemos comparaciones entre dos alternativas, existe una técnica estadística muy potente que nos permite estructurar el proceso de investigación y obtener conclusiones de la manera más eficiente posible, esta técnica la vamos a ver en el siguiente apartado y se conoce como Diseño de Experimentos (DDE).

4. Diseño de experimentos

A la hora de evaluar nuestros algoritmos podemos plantear diferentes factores que pueden afectar a su rendimiento. Estos factores se pueden clasificar en dos grandes grupos: **factores intrínsecos al algoritmo y factores relacionados con las instancias**.

Al primer grupo pertenecen todos los **parámetros** que deben definirse a la hora de **diseñar y ejecutar** el algoritmo. Pueden ser parámetros de diseño (tipo de algoritmo, tipo de movimiento, tipo de criterio de terminación, método para evaluar soluciones, etc.) o parámetros ajustables por el usuario (tamaño de población, tamaño de la lista tabú, valor α en un algoritmo GRASP, etc…).

En el segundo grupo de factores extrínsecos podemos encontrar todo aquello que define las **instancias** del problema que abordamos (número de ciudades en un TSP, valores de peso en un problema de mochila, tamaño del tablero en un problema de n reinas, etc.).

En cualquier caso, deseamos por un lado conocer qué decisiones de diseño y parametrización son las más adecuadas, así como saber para qué instancias son más adecuados nuestros procedimientos de resolución.

Para poder analizar de manera estructurada el efecto de todos los factores es necesario usar un conjunto de técnicas conocidas como diseño de experimentos. En nuestro caso, un experimento será el conjunto de pruebas de nuestro algoritmo que haremos bajo unas condiciones controladas.

Teniendo en cuenta que los factores que influyen en el resultado de nuestro algoritmo serán cualquier variable controlable (p. ej. tipo de algoritmo, tamaño instancia, parámetros…), lo que haremos será determinar para cada factor, el nivel o valor que deberá tomar para que el algoritmo ofrezca los mejores resultados posibles respecto

a una variable de respuesta (normalmente será la RPD, el tiempo, el número de veces que encuentra una solución cercana al óptimo, etc…).

La combinación de los niveles de los diferentes de los diferentes factores definirá lo que conocemos como tratamiento y la ejecución de un tratamiento específico para obtener un cierto valor resultante de la variable de respuesta será un experimento o experiencia.

El diseño de experimentos consiste en establecer cómo vamos a realizar dichos experimentos (en nuestro caso, cómo ejecutaremos el algoritmo considerando los factores que creemos que influyen y fijando sus niveles niveles) para obtener conclusiones estadísticamente válidas con el menor esfuerzo. Aunque hay muchos tipos de diseños de experimentos y muchas técnicas asociadas, nosotros plantearemos lo que se conoce como Diseño Factorial Completo a dos niveles y usaremos Excel con Real Statistics para plantearlo y analizarlo.

El Diseño Factorial Completo de dos niveles se basa en generar todos los experimentos en los que cada experimento consta de dos o más factores y, a su vez, cada factor solo tiene dos niveles posibles. Se generan pues, todas las posibles combinaciones de dichos niveles en todos los factores permitiendo de esta manera estudiar el estudio del efecto de cada factor sobre la variable respuesta.

Como ejemplo, supongamos que estamos analizando el comportamiento de un algoritmo de Recocido Simulado en el que queremos evaluar la influencia de tres factores: Temperatura inicial (T0), Coeficiente de temperatura (α) y número de iteraciones para cada valor de temperatura (A). Lo primero que debemos establecer son los valores de cada factor (si son factores cuantitativos definiremos su valor numérico, pero si fueran cualitativos usaríamos una etiqueta para nombrarlos, como sería si estuviéramos probando dos tipos de operadores para generar vecinos y uno de ellos fuera “swap” y otro “inserción”). En nuestro caso los tres son numéricos y son:

FACTOR	Nivel bajo	Nivel alto
Temperatura inicial (T_0)	100	200
Coeficiente de temperatura (α)	0.80	0.95
Iteraciones a una temperatura dada (A)	5	20

En DDE se reemplazan los nombres o valores por un símbolo –1 o +1 dependiendo si se trata del nivel bajo o alto en el caso de factores cuantitativos (en el caso de factores cualitativos, la asignación es arbitraria). Así, la tabla experimental quedaría:

FACTOR	Nivel bajo	Nivel alto
Temperatura inicial (T_0)	-1	+1
Coeficiente de temperatura (α)	-1	+1
Iteraciones a una temperatura dada (A)	-1	+1

Por lo tanto, un diseño factorial completo consistiría en definir todas las combinaciones de los factores (en este caso son 2^3=8).

Tratamiento	T0	α	A
1	-1	-1	-1
2	+1	-1	-1
3	-1	+1	-1
4	+1	+1	-1
5	-1	-1	+1
6	+1	-1	+1
7	-1	+1	+1
8	+1	+1	+1

Como puedes ver, cada tratamiento consiste en ejecutar el algoritmo con la combinación de factores que corresponda (p ej. el tratamiento 1 consistirá en ejecutar el Recocido Simulado con unos niveles bajos de T_0, α y A (100, 0,8 y 5).

Una pregunta que podemos plantearnos en este punto es ¿cuántas veces repetir cada tratamiento? La respuesta dependerá, por supuesto, de la variabilidad de los resultados y puede ser calculada estadísticamente. Sin embargo, para evitar complicar más este tutorial y teniendo en cuenta que el propio diseño factorial completo

permite no tener que hacer muchas repeticiones de cada tratamiento, recomendamos que al menos se realicen cuatro réplicas de cada tratamiento.

Es de sentido común que cada tratamiento y réplica se deben hacer intentando que no varíen el resto de factores que puedan influir en el resultado. En concreto, todos deberían de hacerse usando el mismo ordenador, permitiendo el mismo tiempo de cálculo, usando el mismo código y usando la misma instancia.

Así, pues, en la siguiente tabla se muestran los resultados de hacer una serie de experimentos en los que se ha ejecutado el algoritmo de Recocido Simulado durante 30 segundos sobre la misma instancia del TSP y en un ordenador portátil. Cada tratamiento se ha repetido 4 veces y el valor de la distancia total de la mejor solución obtenida en cada tratamiento y repetición, se muestra en cada celda:

Tratamiento	Rep 1	Rep 2	Rep 3	Rep 4
1	27	42	38	36
2	42	62	53	42
3	47	47	60	72
4	72	64	59	65
5	60	59	54	60
6	53	50	47	57
7	75	68	65	78
8	52	57	67	71

Antes de aplicar cualquier análisis, vamos a transformar los valores absolutos en RPD teniendo en cuenta que el menor valor obtenido en todos los experimentos es 27.

Tratamiento	Rep 1	Rep 2	Rep 3	Rep 4
1	0	0,55	0,4	0,33
2	0,55	1,29	0,96	0,55
3	0,74	0,74	1,22	1,66
4	1,66	1,37	1,18	1,4
5	1,22	1,18	1	1,22
6	0,96	0,85	0,74	1,11
7	1,77	1,51	1,4	1,88
8	0,92	1,11	1,48	1,62

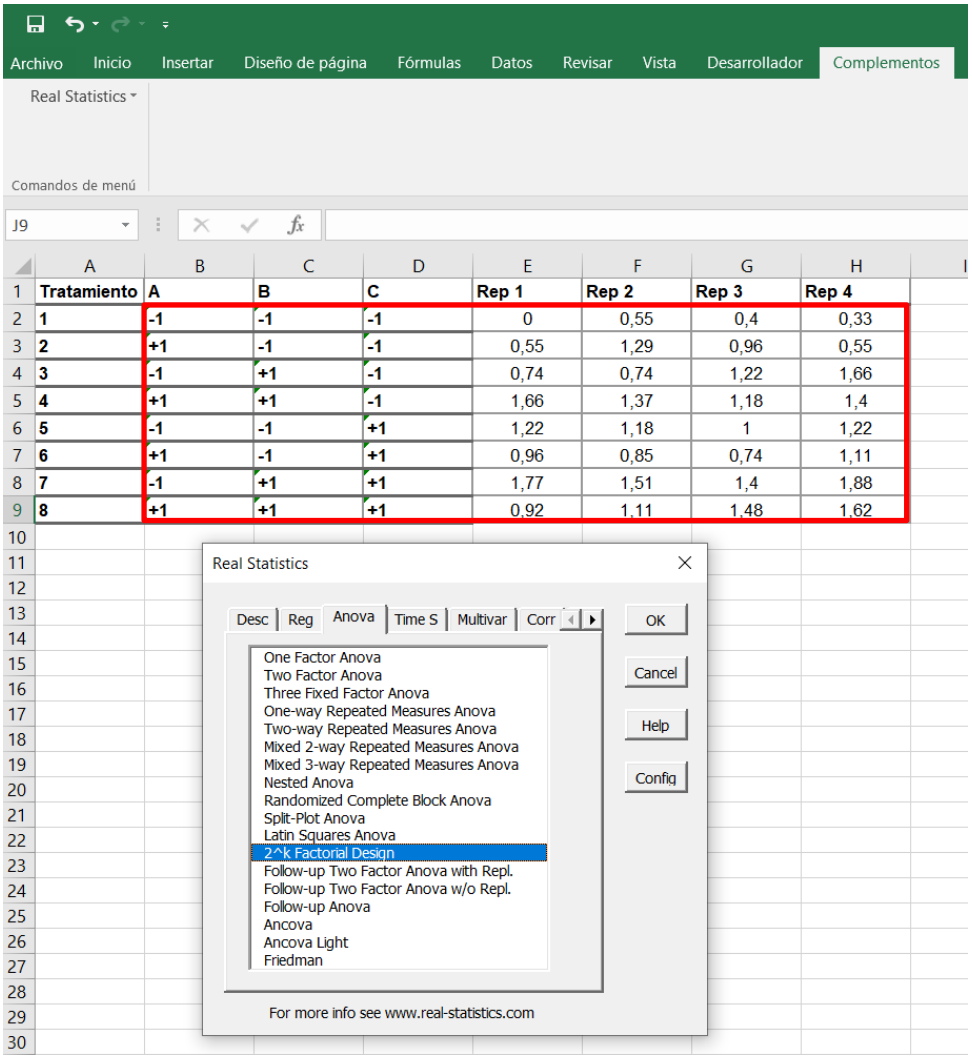
Ahora vamos a usar las técnicas basadas en Análisis de la Varianza de manera similar a lo que lo hemos hecho en los apartados anteriores para determinar si hay diferencias estadísticamente significativas de los resultados cuando cambiamos los niveles de los diferentes factores.

Para poder realizar el análisis la tabla de tratamientos y datos deberán configurarse de la siguiente manera:

Tratamiento	A	B	C	Rep 1	Rep 2	Rep 3	Rep 4
1	**-1**	**-1**	**-1**	0	0,55	0,4	0,33
2	**+1**	**-1**	**-1**	0,55	1,29	0,96	0,55
3	**-1**	**+1**	**-1**	0,74	0,74	1,22	1,66
4	**+1**	**+1**	**-1**	1,66	1,37	1,18	1,4
5	**-1**	**-1**	**+1**	1,22	1,18	1	1,22
6	**+1**	**-1**	**+1**	0,96	0,85	0,74	1,11
7	**-1**	**+1**	**+1**	1,77	1,51	1,4	1,88
8	**+1**	**+1**	**+1**	0,92	1,11	1,48	1,62

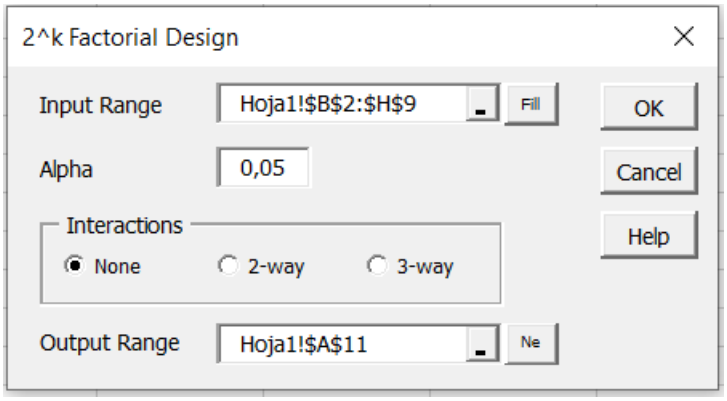
Nótese que se han cambiado los nombres de los factores por letras para poder interpretar mejor los resultados de la tabla ANOVA que vamos a generar con el paquete Real-statistics.

Una vez escrita la tabla en Excel, abriremos Real-statistics pulsando Ctrl y m o directamente en la pestaña Complementos del Menú:



	A	B	C	D	E	F	G	H
1	**Tratamiento**	**A**	**B**	**C**	**Rep 1**	**Rep 2**	**Rep 3**	**Rep 4**
2	**1**	**-1**	**-1**	**-1**	0	0,55	0,4	0,33
3	**2**	**+1**	**-1**	**-1**	0,55	1,29	0,96	0,55
4	**3**	**-1**	**+1**	**-1**	0,74	0,74	1,22	1,66
5	**4**	**+1**	**+1**	**-1**	1,66	1,37	1,18	1,4
6	**5**	**-1**	**-1**	**+1**	1,22	1,18	1	1,22
7	**6**	**+1**	**-1**	**+1**	0,96	0,85	0,74	1,11
8	**7**	**-1**	**+1**	**+1**	1,77	1,51	1,4	1,88
9	**8**	**+1**	**+1**	**+1**	0,92	1,11	1,48	1,62

En el menú del paquete buscaremos la pestaña Anova y elegiremos 2^k Factorial Design. El rango de entrada serán las celdas que contienen los valores -1 y +1 de los niveles, así como las de los resultados (como se puede ver en la figura anterior):



:

Para facilitar el análisis no vamos a tener en cuenta las interacciones (esto quiere decir que no vamos a estudiar si hay alguna relación entre varios factores entre sí). En un análisis más detallado puede tener sentido estudiar estas interacciones. El resultado se mostrará en la celda A11:

2^k Factorial Design					
	SS	*df*	*MS*	*F*	*p-value*
A	0,02702813	1	0,02702813	0,25119596	0,6201519
B	2,39257813	1	2,39257813	22,2363175	6,0354E-05
C	0,90115313	1	0,90115313	8,37520278	0,00728908
Err	3,0127375	28	0,10759777		
Tot	6,33349688	31	0,20430635		

Para identificar los factores o interacciones que son tienen influencia estadística sobre los resultados hemos de fijarnos en los valores de la columna p-value que muestra (del mismo modo que en la sección anterior cuando analizábamos dos algoritmos diferentes) la probabilidad de equivocarnos si rechazamos la hipótesis de que los resultados son iguales cuando cambia el factor.

Todos aquellos que son inferiores a 0,05 indican que la probabilidad de equivocarnos es muy baja cuando decimos que cambiar el factor no influye en el resultado.

O lo que es lo mismo, podemos tener una gran probabilidad de acertar cuando decimos que cambiar el nivel del factor, influye significativamente sobre los resultados. En nuestro caso, se han remarcado en amarillo los que influyen en los resultados y son los correspondientes a α y A. En cambio, el factor de temperatura inicial no influye. Ahora podemos utilizar directamente la otra tabla que se nos muestra, para identificar la importancia de cada factor y cómo influyen sobre el resultado:

				alpha	0,05	
	effect	*s.e.*	*t-stat*	*p-value*	*lower*	*upper*
A	0,058125	0,11597293	0,50119454	0,6201519	-0,17943478	0,29568478
B	0,546875	0,11597293	4,71554	6,0354E-05	0,30931522	0,78443478
C	0,335625	0,11597293	2,89399426	0,00728908	0,09806522	0,57318478

En esta tabla nos fijamos en la columna effect solo para los factores e interacciones significativas (en amarillo). En ella se muestra la diferencia entre las medias de los resultados cuando cada factor tiene su valor superior (+) menos las medias de los resultados cuando tiene su valor inferior (-). Por ejemplo, se nos indica que, para B, la media de los resultados cuando α=0,95 (valor superior) es (0,74+0,74+…+1,48+1,42)/16=1,35375 mientras que la media cuando α=0,80 (valor inferior) es (0+0,55+…+0,74+1,11)/16=0,806875 y por lo tanto la diferencia entre ambas es 0,546875.

Como estamos minimizando, cualquier efecto positivo implica empeorar en media el valor de la función objetivo (en el ejemplo del factor B, en media el valor superior aumenta en 0,546875 la RPD) por lo que la conclusión es que debemos elegir el nivel bajo cuando los efectos sean positivos y el nivel alto si los efectos fueran negativos.

En resumen, en nuestro caso es irrelevante el valor de T0 que elijamos para los experimentos mientras que fijaremos α=0,8 y A=5 ya que, siendo significativos, sus niveles bajos permiten obtener unas RPD menores.

Este procedimiento es relativamente rápido y podemos usar otros niveles de los factores anteriores para ir afinando las conclusiones. También se usa de la misma manera cuando estamos evaluando la influencia de factores relacionados con las instancias. Al nivel en el que estamos, este método es suficiente, aunque presenta limitaciones que se comentarán en el apartado de conclusiones.

Para terminar, en el siguiente apartado vamos a definir el método general que sugerimos que se utilice para analizar los algoritmos que programemos.

5. Propuesta de método de experimentación y análisis de algoritmos.

En resumen, los pasos a seguir para analizar los algoritmos que programemos se pueden plantear en tres fases:

Fase 0. Identificar qué queremos analizar.

Normalmente analizaremos su RPD respecto a la mejor solución conocida, pero también podríamos analizar su tiempo de cálculo, el tiempo que necesita para converger a una buena solución, etc…

Fase 1. Ajuste de los parámetros de los diferentes algoritmos.

En esta fase plantearemos un ajuste de los parámetros de diseño y funcionamiento de cada algoritmo. Por ello, para cada algoritmo que planteemos tenemos que:

- Elegir los factores de diseño que creamos que funcionen mejor. P. ej. Determinar cuál es el mejor entre varios operadores de cruce, de generación de vecindario, de representación de soluciones, etc..
- Ajustar los parámetros de los algoritmos que hagan que funcionen mejor (% mutación, α, tiempo de ejecución, etc..). Por ello, tendremos que identificar los factores y niveles que deseamos ajustar.

Para esta fase usaremos todo lo visto en el apartado anterior y su resultado serán algoritmos que "en principio" están ajustados para resolver el tipo de problemas que deseamos. Hay que remarcar que las instancias que usaremos para calibrar los parámetros deben ser similares a las que desearemos resolver en la fase 3, pero no deben ser las mismas para evitar que el algoritmo se diseñe solamente para una instancia específica.

Fase 2. Comparación de algoritmos entre sí para instancias de ejemplo.

Podremos usar lo visto en los apartados 3 y 4, en función del número de algoritmos que deseemos comparar (si son dos, podemos considerar el tipo de algoritmo como factor de dos niveles y usar lo visto en el apartado 4, pero si son más algoritmos o analizamos un factor a más niveles podremos usar lo visto en el apartado 3). En todo caso, uno de los algoritmos con los que podríamos comparar sería el que generase soluciones de manera totalmente aleatoria y, por supuesto, usaríamos instancias diferentes a las de la fase 1 pero tratando de que todos los algoritmos que comparemos se ejecuten en el mismo ordenador y durante el mismo tiempo.

Podemos también analizar los algoritmos comparados con diferentes instancias, esto consiste en variar los parámetros de las instancias (número de ciudades, distribución de tiempos, de distancias, etc…) para determinar si existen características de las instancias que produzcan diferencias entre qué algoritmo las resuelve mejor.

Fase 3. Aplicación del algoritmo elegido a la instancia (o instancias) que queramos resolver.

Una vez calibrado el mejor algoritmo mediante las fases anteriores, lo aplicaríamos a la instancia o instancias a resolver y obtendríamos las medias y desviaciones típicas de las funciones objetivo de las soluciones (en el caso que el algoritmo obtenga diferentes soluciones cada vez que se ejecute).

Conclusiones

Como se ha visto en las páginas anteriores, el análisis experimental de algoritmos de optimización se basa en una buena **aplicación de técnicas estadísticas** bien conocidas. Muchas de ellas tienen limitaciones a la hora de aplicarse (por ejemplo, ANOVA se basa en que los residuos[7] tengan una distribución normal, que su varianza

[7] Residuo en estadística es la diferencia entre el valor observado de la variable y el valor previsto por el modelo de regresión lineal subyacente

sea constante y que haya independencia de los resultados respecto al tiempo) por lo que se deben comprobar que se pueden aplicar las técnicas a partir de los resultados que se obtengan.

Para verificar la normalidad de los resultados se suelen descartar de las muestras aquellos valores que se separan mucho de las medias, aunque se ha comprobado experimentalmente que el test ANOVA es muy robusto frente a la falta de normalidad (sobre todo cuando se repiten mucho los experimentos)[8].

En cuanto a la independencia de las muestras, esta se comprueba fácilmente representando los resultados y comprobando que se distribuyen de manera aleatoria en torno a la media. Afortunadamente, esto debería suceder siempre en los experimentos de algoritmos que planteamos puesto que la ocurrencia de cada ejecución de cada experimento no influye sobre la probabilidad de ocurrencia de otro experimento siempre que tengamos la precaución de usar semillas aleatorias elegidas a su vez de manera aleatoria (por ejemplo, mediante el uso del reloj interno del ordenador).

La homocedasticidad o igualdad de varianzas entre los experimentos requiere un análisis de la dispersión de los residuos con respecto a los diferentes niveles de cada factor.

En el caso que no se cumplan las condiciones para poder hacer un Análisis de la Varianza, es necesario usar otros tipos de tests que requieren un nivel mayor de conocimiento estadístico (como pueden ser los test no paramétricos).

Como complemento a los análisis estadísticos podemos usar representaciones gráficas sencillas que nos ayuden a entender mejor su comportamiento (p. ej. gráficas de la evolución de la función objetivo de la mejor solución o la solución actual en cada iteración).

Sin embargo, cualquier conclusión sólida que queramos obtener debe estar basada en **método** como el que ha sido descrito en las páginas anteriores. Por supuesto que podemos complicarlo y analizar más exhaustivamente los factores (usando más de dos niveles, suponer su no linealidad, etc…) pero la manera en general de hacer el análisis será similar. Por ello, esta primera aproximación al estudio de algoritmos debe ser el punto de partida para todos aquellos que deseen desarrollar de forma rigurosa cualquier procedimiento de optimización.

Bibliografía

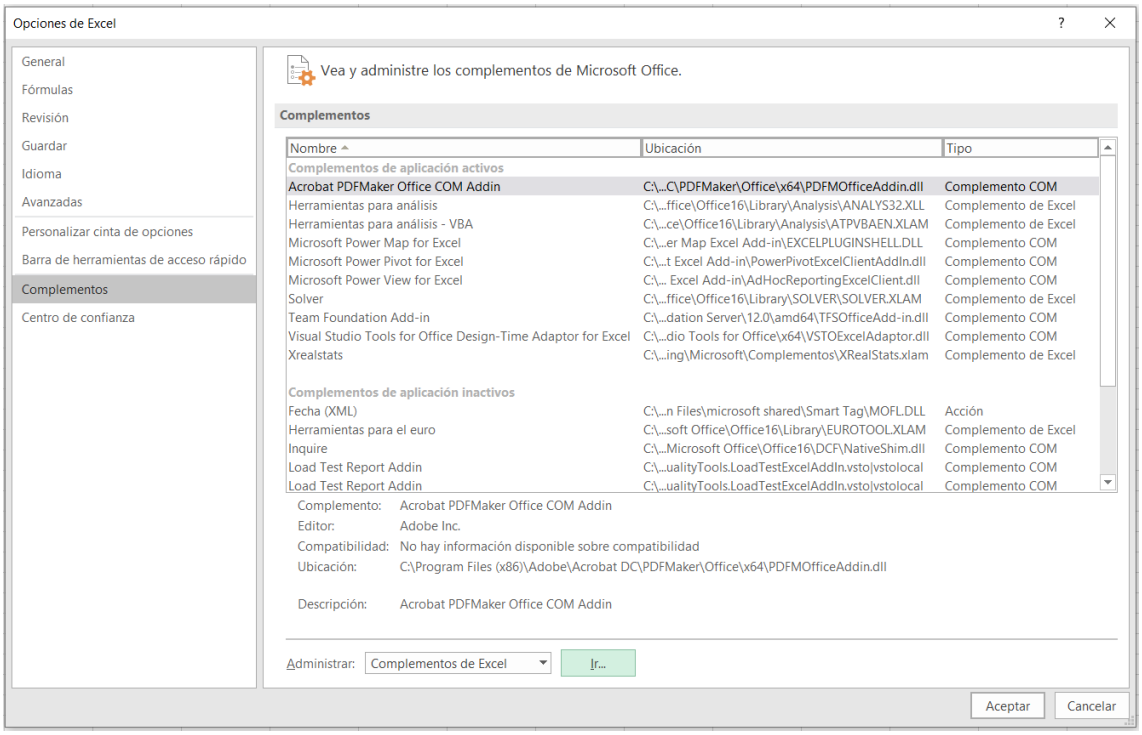
ANDRÉS, C. (2017) *Optimización metaheurística para ingenieros. Aplicaciones en Organización Industria*" Ed. Delta
BARTZ-BEIELSTEIN, TH., CHIARANDINI, M., PAQUETE, L. y PREUSS, M. (2010). *Experimental Methods for the Analysis of Optimization Algorithms*. Springer
MONTGOMERY, D. (2005). *Diseño y Análisis de Experimentos.* Ed. Limusa
ZAIONTZ, C. (2020) *Real Statistics Using Excel.* www.real-statistics.com [Fecha del último acceso: 17-10-2022]

[8] En todo caso, la manera rigurosa de comprobar la normalidad en este tipo de experimentos es usando los test de Kolmogorov-Smirnov que compara la distribución acumulada de los datos observados con la distribución acumulada esperada por una distribución Normal, obteniendo el valor de p basándose en la discrepancia entre ambas. En cuanto a la comprobación de la homocedasticidad se suele hacer con el test de Levene. Los detalles de estas pruebas se pueden encontrar en cualquier texto de estadística.

ANEXO: Instalando Real Statistics

Real Statistics es un módulo gratuito que se puede instalar en Excel para realizar análisis estadísticos que ha sido desarrollado por el Doctor Charles Zaiontz. Este paquete de software amplía las capacidades estadísticas integradas de Excel permitiendo realizarlo de manera sencilla. A través de la página web[9] se puede descargar el fichero XRealStats.xlam (para usuarios de Windows se sugiere hacerlo en el directorio C:\Usuarios\nombre-usuario\AppData\Roaming\Microsoft\AddIns).

Además, tienes que tener instalado Solver (mira la ayuda de Windows para saber cómo se hace). A continuación, abre una hoja en blanco de EXCEL y selecciona en la pestaña Archivos/Opciones/ para abrir la siguiente pantalla de la opción Complementos:



En esta pantalla, ve al botón de “Ir…” que aparece al lado del desplegable y púlsalo. Te aparecerá una ventana para instalar el nuevo complemento. Solo tienes que usar el botón “Examinar” y elegir el archivo XRealStats.xlam en su ubicación.

Ahora ya tienes instalado el complemento, al que podrás acceder pulsando las teclas **Ctrl y m** simultáneamente o en una pestaña denominada complementos que se habrá generado en la barra de herramientas.

[9] https://www.real-statistics.com/free-download/real-statistics-resource-pack/