Práctica: Configurador de productos

Jorge Juan González

Contenido

Introducción 2
Problema 2
Individuo 3
Fitness 3
Generación del entorno 6
Evaluación 6
Algoritmo Genético 6
Diseño 6
Población inicial 6
Procesos 7
Evaluación 8
Tamaño del problema 8
Exploración de soluciones 9
Parámetros de evaluación 11
Conclusiones 15
Enfriamiento simulado 15
Diseño 15
Evaluación 16
Tamaño del problema 16
Parámetros de evaluación 17
Conclusiones 19
Comparativa 19
Conclusiones 21

Introducción

Este documento constituye la memoria de la práctica de la asignatura Técnicas de Inteligencia Artificial. En esta práctica modelaremos un problema y trataremos de resolverlo utilizando un algoritmo genético y un algoritmo de enfriamiento simulado. Una vez implementados estos algoritmos, evaluaremos el rendimiento de cada algoritmo sobre distintos tamaños del problema utilizando la parametrización por defecto. Después, evaluaremos cómo afecta cada uno de los parámetros de cada algoritmo para encontrar su parametrización óptima. Finalmente, utilizando los mejores parámetros encontrados, compararemos el rendimiento de estos dos algoritmos en la resolución del problema.

Problema

El problema que trataremos de resolver en esta práctica es la configuración automática de productos, es decir, dada una empresa que va a fabricar un producto, conocer a qué proveedor comprar cada componente del producto y cuánto aumentar el precio sobre el coste de producción de forma que se maximicen los beneficios.

Cada componente tiene una lista de posibles proveedores, ofreciendo cada uno de ellos una calidad y precios distintos. Pueden existir proveedores disruptivos que ofrezcan componentes de buena calidad a buen precio y proveedores ineficientes que ofrezcan componentes de mala calidad a un alto precio. Cada componente puede ser comprado o fabricado por proveedores distintos.

Cada posible cliente objetivo tiene un porcentaje del total del mercado de consumidores y da diferente peso a la calidad total percibida y al precio total. Pueden existir clientes que necesitan el producto y a los que les importa poco la calidad y poco el precio. De igual modo, pueden existir clientes muy exigentes que sólo estén dispuestos a comprar el producto cuando sea de muy buena calidad a muy buen precio. Con cliente nos referiremos a clase o arquetipo de cliente.

Por otro lado, cada cliente dará diferente importancia a cada componente. Por ejemplo, si el producto es un teléfono móvil, habrá clientes que valoren mucho la calidad de la batería mientras que otros valorarán mucho la calidad de la cámara.

La suma de los costes de producción de cada componente representará el coste de producción del producto al que se le añadirá un margen de beneficio. El coste de producción más el beneficio representa el precio final del producto.

Existe un precio de mercado máximo y mínimo que sirve de referencia a los clientes a la hora de percibir un producto como caro o barato. Para evitar precios abusivos, se ha establecido la restricción de que el precio final de nuestro producto no debe superar el precio máximo de mercado, aunque haya clientes dispuestos a pagarlo.

Asumiremos que se fabrican tantos productos como se venden de modo que no hay un excedente de producción que pueda representar pérdidas.

Este problema presenta las siguientes características que lo hacen atractivo y por lo que ha sido elegido para esta práctica:

- Aplicabilidad: Al trabajar en términos genéricos de productos y componentes, esta solución se puede aplicar en gran variedad de productos como hardware, construcción, automoción e incluso software, entendiendo los componentes como features y los proveedores como, por ejemplo, distintos equipos o metodologías de desarrollo.
- Flexibilidad: La complejidad del problema es fácilmente escalable escalando el número de componentes, clientes y proveedores. Además, puede aislarse o simplificarse el problema estableciendo un único cliente disponible para obtener la mejor configuración

para ese cliente o un único componente para obtener la mejor configuración si sólo podemos elegir un proveedor.

Nos referiremos al conjunto de componentes, proveedores y clientes como entorno.

El principal reto de este problema es conseguir una representación objetiva y fiable del entorno donde resulta fundamental saber medir de forma objetiva la calidad de un componente o cuánto valora un cliente la calidad de cada componente.

Existe una variante del problema por la cual la empresa sólo puede elegir un cliente al que vender y deberá encontrar el mejor cliente al que dirigirse. Esta variante sería más fiel al caso en el que sólo pudiéramos fabricar un producto y debiéramos decidir a qué cliente vendérselo y a qué precio.

Nos enfocaremos en un escenario de producción tradicional en el que todos los clientes pueden comprar nuestro producto, aunque a unos les resultará más atractivo que a otros. De igual forma, dado que la variante también ha sido implementada, se comentará a lo largo de la memoria qué implicaciones en el diseño tendría el haber escogido esta variante.

Individuo

En este problema, un individuo o solución representará una estrategia que especifica qué proveedor se elige para cada componente y qué margen de beneficio añadiremos al coste de producción.

Sabiendo esto, para Nc componentes, el genotipo de un individuo está codificado como un diccionario compuesto de las siguientes claves:

- Suppliers: Lista de longitud Nc que almacena el índice del proveedor elegido para cada componente c.
- Profit: Valor numérico que representa el margen de beneficio que se añadirá al coste de producción.

Variante: El diccionario que codifica el genotipo también incluiría una clave Client con el índice de la lista de clientes en el que se almacena el cliente al que se dirige el individuo.

Fitness

Éste es un problema de maximización, es decir, trataremos de encontrar un individuo que maximice un cierto valor de fitness o coste (Z). Este coste representa los beneficios que obtendría la empresa de aplicar la configuración que describe un individuo, por cada cliente del total del mercado de consumidores.

Si el 40% de todos los consumidores compran el producto y nos llevamos un beneficio de 1000€ por cada venta, Z = 0.4 * 1000 = 400€/potencial cliente. Para un total de 200 clientes estaríamos obteniendo un beneficio de 200 * 0.4 * 1000 = 80000€ (80 ventas).

Obtener un Z=0.001, por ejemplo, implicaría que necesitamos un mercado de consumidores de, al menos, 1000 consumidores para que nos sea rentable.

Para evaluar un individuo usamos una función de coste (fitness) que calcula dicho valor Z. Este cálculo se realiza de la siguiente manera:

- Coste de producción (Pp): Suma de los costes de producción de cada componente de acuerdo con el coste de producción del proveedor (supplier) elegido para ese componente.

$$Pprod = \sum_{comp=1}^{Ncomp} cost_{sup,comp}$$

- Precio final (Pf): Precio final resultante de añadir el margen de beneficio (profit) al Pp.

$$Pfinal = Pprod + Profit$$

- Precio normalizado (Pn): Pf normalizado al rango [0,1] relativo a los precios de mercado máximo y mínimo. Pn=1 significa que el producto es tan caro como el producto más caro del mercado. Pn=0 significa que el producto es tan barato como el producto más barato del mercado.

$$Pnorm = \frac{Pfinal - minP}{maxP - minP}$$

- Por cada cliente:
    - Percepción de calidad (Qp): Suma de la calidad que da el proveedor de cada componente ponderada por la importancia que le da el cliente a ese componente. Esta suma está normalizada respecto al número de componentes de modo que se mantenga dentro del rango [0,1].

$$Qperc_{client} = \sum_{comp=1}^{Ncomp} quality_{sup,comp} * Wquality_{client,comp}$$

    - Atractivo (D): Suma de Qp y Pn ponderada por la importancia que le da el cliente a la calidad y al precio. Si este valor da 1, significa que todos los clientes de ese tipo comprarán el producto.

$$D_{client} = Qperc_{client} * Wquality_{client} + Pnorm * Wprice_{client}$$

    - Ventas (Sc): D multiplicado por la fracción que ocupa el tipo de cliente elegido en el mercado de consumidores (market share). Este valor representa la fracción del mercado de consumidores que adquirirá el producto. Para mercados fragmentados (muchos clientes con poco market share) es esperable obtener valores de Sc muy pequeños.

$$S_{client} = D_{client} * MktShare_{client}$$

- Ventas totales (St): Sumatorio de Sc para cada cliente. Obtener un valor S=1 significa que todos los consumidores (de todos los tipos) adquirirán el producto lo cual nunca ocurrirá en un escenario realista.

$$S = \sum_{client=1}^{Nclients} S_{client}$$

- Z: St multiplicada por el margen de beneficio que obtenemos por la venta de cada producto. Si este valor es menor o igual a 0, Z valdrá 0.

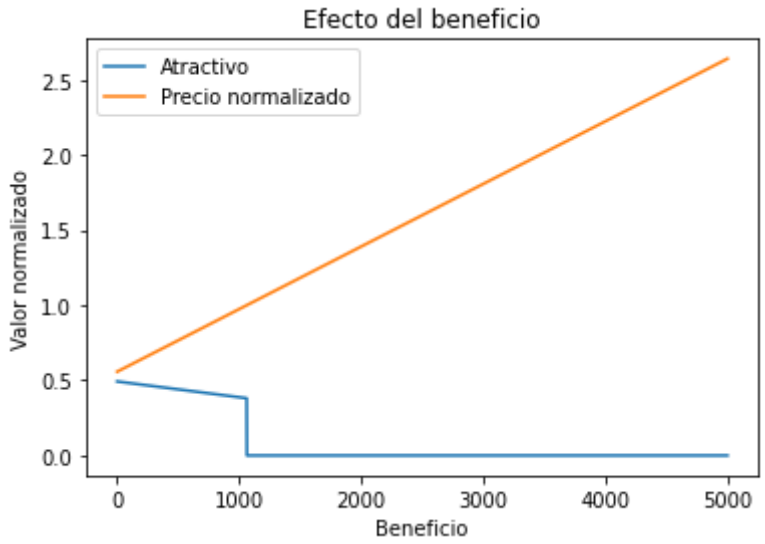
$$Z = \begin{cases} S * Profit & S > 0 \\ 0 & S <= 0 \end{cases}$$

Variante: St sería igual a S y sólo calcularíamos S para el cliente objetivo.

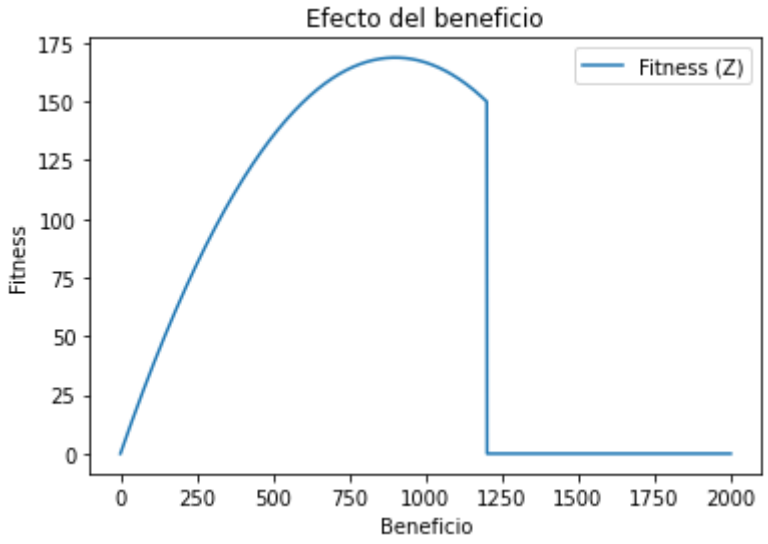
La función fitness devuelve un diccionario con todos estos valores de modo que puedan ser inspeccionados en el futuro.

La siguiente figura muestra cómo el aumento del margen de beneficio afecta en el atractivo de un producto, manteniendo los mismos proveedores y el mismo cliente. Como podemos observar, el atractivo alcanza su máximo cuando el beneficio es 0 (precio final mínimo) pero si la calidad es suficientemente buena los clientes pueden estar dispuestos a comprar el producto, aunque sea hasta 2,5 veces más caro que el producto más caro del mercado (precio normalizado = 1).

Para evitar que el algoritmo dé por válidos productos excesivamente caros (precio normalizado > 1) haremos que los clientes rechacen comprar productos con precios abusivos. La siguiente figura muestra cómo el aumento del margen de beneficio afecta en el atractivo de un producto (aplicando esta corrección), manteniendo los mismos proveedores y el mismo cliente.



En la siguiente gráfica podemos ver cómo aumentar el beneficio afecta en el fitness para un entorno simplificado con un cliente, un componente y un proveedor genéricos. Usar márgenes de beneficio bajos puede resultar en muchas ventas que no se traducen en ingresos para la empresa. En cambio, aplicar beneficios por encima de 1000 (para este ejemplo) también resulta perjudicial para la empresa porque estos beneficios no compensan la bajada en las ventas. Antes de llegar a este punto existe un pico máximo que es el que deseamos encontrar.



La caída a 0 se debe a que a partir de un beneficio cercano a 1250, el precio total del producto excede el precio máximo del mercado y se aplica la corrección que hemos visto previamente.

Generación del entorno

El entorno en el que trataremos de maximizar beneficios será generado aleatoriamente a partir de los siguientes parámetros:

- Ncomp: Número de componentes.
- Ncli: Número de clientes.
- Nsupp: Número de proveedores.
- MaxMktPrice: Máximo precio de mercado.
- MinMktPrice: Mínimo precio de mercado.

* En la implementación realizada todos los componentes tienen el mismo número de proveedores, pero esto no es una restricción del problema.

Los clientes tendrán un peso aleatorio para la calidad y para el precio. La suma de los market share de todos los clientes dará siempre 1 aunque su distribución también será aleatoria.

Para cada componente se genera una lista de Nsupp proveedores donde cada uno tendrá una calidad aleatoria entre 0 y 1 y un precio de componente aleatorio comprendido entre MinMktPrice/Ncomp y MaxMktPrice/Ncomp.

Aunque no se utiliza, cada tipo de cliente y cada componente tiene un nombre generado aleatoriamente como un código alfanumérico.

Para las evaluaciones utilizaremos los siguientes entornos que representan distintos tamaños de problema:

	Compacto	**Medio**	**Fragmentado**
Núm. clientes	5	25	50
Núm. componentes	15	82	150
Núm. proveedores por cada componente	10	55	100
Precio mínimo de mercado	300.00 €	300.00 €	300.00 €
Precio máximo de mercado	2700.00 €	2700.00 €	2700.00 €

Algoritmo Genético

Diseño

Población inicial

Para una población de tamaño Npop, la población inicial se ha creado generando aleatoriamente Npop individuos. La generación aleatoria de un individuo se realiza de la siguiente manera:

- Elección de un índice aleatorio de entre los proveedores disponibles para cada componente.
- Elección de un beneficio aleatorio dentro de un rango (de 10 a 10000).

Variante: También habría que elegir un índice aleatorio de entre los clientes disponibles.

Procesos

El algoritmo genético implementado admite los siguientes parámetros:

- Número de iteraciones o generaciones (n_iter)
- Máximo número de generaciones sin mejorar el score (n_convg)
- Lista de tamaños de población (N_pop)
- Número de individuos que participan en cada torneo (k_tnmt)
- Ratio de selección (r_sel)
- Ratio de cruce (r_cross)
- Ratio de mutación (r_mut)
- Máximo decremento del beneficio (maxProfidtDrop)
- Máximo incremento del beneficio (maxProfitIncrease)

Estos parámetros influyen en los distintos procesos del algoritmo, los cuales son explicados a continuación siguiendo el orden en el que se realizan en cada generación.

Selección

La selección de un individuo apto para emparejamiento se realiza por torneo. Se eligen k_tnmt individuos aleatorios de la población y nos quedamos con el que tenga mejor fitness.

Haremos esta selección hasta conseguir una lista de padres tan grande como la fracción r_sel de la población.

Cruce

Se eligen los pares de padres en orden siguiendo la lista de padres obtenida tras la selección. A partir de cada par de padres obtendremos un par de individuos hijos, copias de cada uno de los padres. Después se combinarán o cruzarán los genotipos de sus padres de la siguiente manera:

- Si se da la probabilidad (r_cross) de que la lista de proveedores del hijo se deba cruzar, se elegirá un índice aleatorio de la lista de índices de proveedores al que nos referiremos como puntero. Después se permutará la lista de índices de proveedores en ambos hijos, usando el puntero como pivote.
- Si se da la probabilidad (r_cross) de que el valor de beneficio deba cruzarse, ambos hijos tendrán la media del beneficio de ambos padres.

Variante: Habría que añadir que si se da la probabildiad (r_cross) de que el cliente deba cruzarse, el cliente del padre1 se asignará al hijo 2 y viceversa.

Mutación

Cada uno de los hijos, tras el cruce, se someten a un proceso de mutación que se realiza de la siguiente manera:

- Por cada índice de proveedor (tantos índices como componentes), si se da la probabilidad (r_mut) de que el proveedor deba mutar, se elegirá otro proveedor aleatorio de entre los proveedores del componente correspondiente a ese índice.
- Si se da la probabilidad (r_mut) de que el beneficio deba mutar, se sumará al beneficio actual un valor aleatorio comprendido en el rango [-maxProfitDrop, maxProfitIncrease].

Variante: Habría que añadir que si se da la probabilidad (r_mut) de que el cliente deba mutar, se escogerá un cliente aleatorio de entre los clientes disponibles.

Corrección

Después del cruce y la mutación, un hijo puede representar una solución inválida. En este problema una solución inválida es un individuo con un beneficio negativo. Si esto ocurre, se cambiará el “gen” del beneficio a 0.

Reemplazo

Una vez se han realizado los procesos anteriores para una generación, se ordenan los individuos de la población en función de su fitness en orden decreciente. Teniendo los individuos ordenados, reemplazaremos los peores individuos por los hijos obtenidos. Si la fracción de la población de la que generaremos hijos (r_sel) es menor de 0.5, se mantendrán los padres, se perderán los peores individuos y también se conservarán parte de los individuos que no fueron seleccionados.

Criterio de terminación

Si se ha establecido un máximo número de generaciones sin mejorar el score (n_convg > 0), el algoritmo terminará cuando se detecte convergencia, es decir, cuando no se obtenga una mejor solución después de n_convg generaciones.

Evaluación

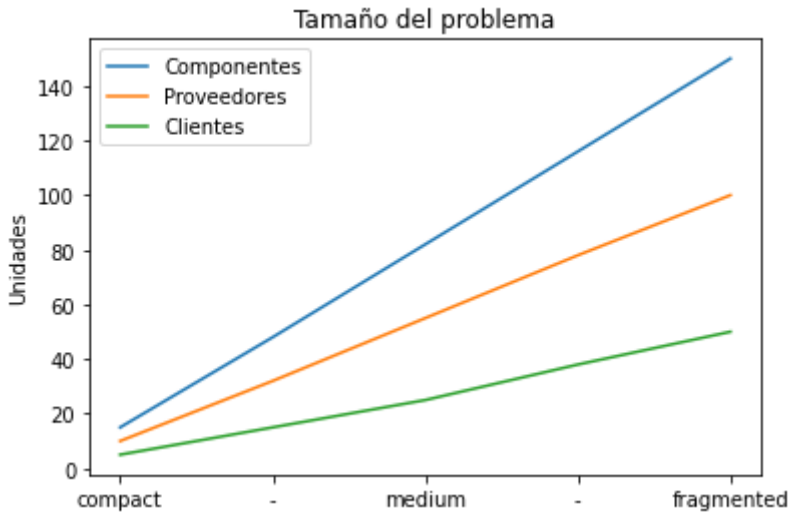
Los parámetros del algoritmo tendrán los siguientes valores por defecto:

- Número de generaciones: 500
- Máximo número de generaciones sin mejora: -1
- Tamaño de la población: 200
- Tamaño de los torneos: 3
- Ratio de selección: 40%
- Ratio de cruce: 80%
- Ratio de mutación: 5%
- Máximo decremento del beneficio: -2400
- Máximo incremento del beneficio: 2400

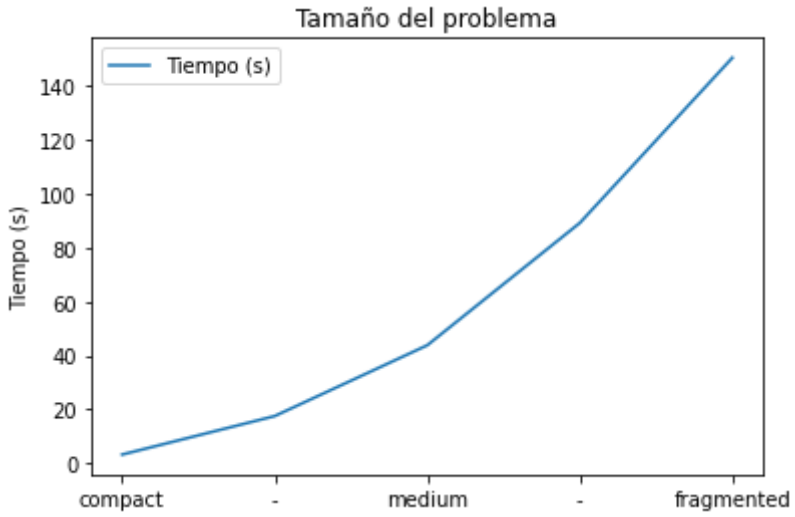
Por defecto, no activaremos el criterio de terminación para comparar resultados con el mismo número de generaciones (n_convg = -1).

Tamaño del problema

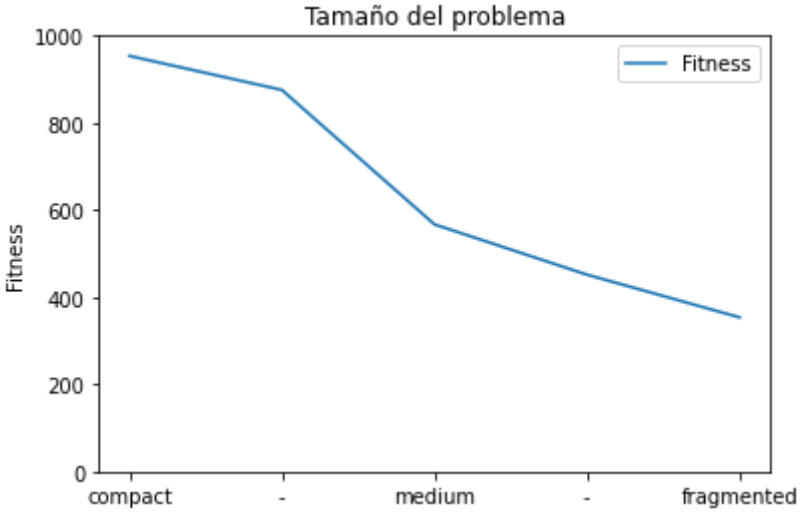
Se ha evaluado cómo afecta el tamaño del problema al rendimiento del algoritmo genético. Para ello, se ha ejecutado el algoritmo con los parámetros por defecto sobre los entornos 'Compacto', 'Medio' y 'Fragmentado'. La siguiente gráfica muestra cómo los tamaños de problema probados crecen linealmente.



El tiempo de cómputo crece de forma exponencial con el tamaño del problema como se puede ver claramente en la siguiente gráfica. Para el tamaño de problema más grande probado, la ejecución ha tenido una duración de más de 2 minutos.



Además, el fitness máximo al que logra llegar el algoritmo decrece con el tamaño del problema como podemos ver en la siguiente gráfica.



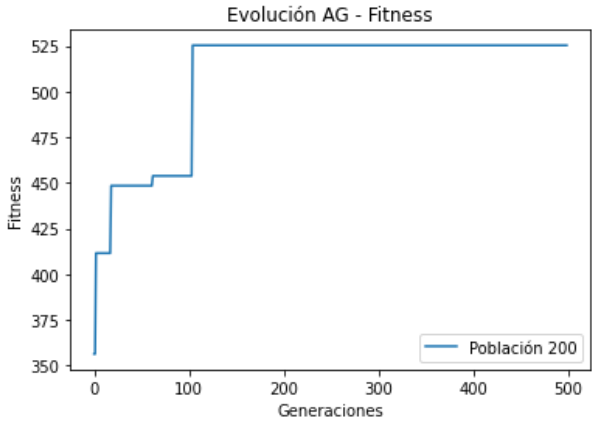
Esto se debe a que conforme aumentamos el tamaño del problema:

1. Al haber más clientes será más probable que existan clientes poco exigentes pero cada cliente influirá menos en el total de ventas.
2. Al haber más proveedores será más probable que existan proveedores disruptivos pero, al haber más componentes, encontrar un proveedor disruptivo tendrá un menor impacto en los beneficios totales.

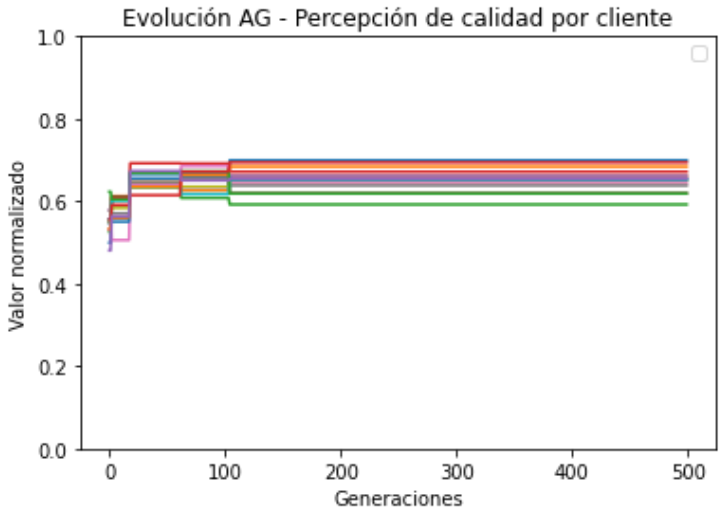
Exploración de soluciones

Podemos inspeccionar qué cambios han permitido al algoritmo encontrar un mejor individuo. Las siguientes gráficas muestran esta evolución sobre una ejecución del algoritmo usando el entorno y los parámetros por defecto.

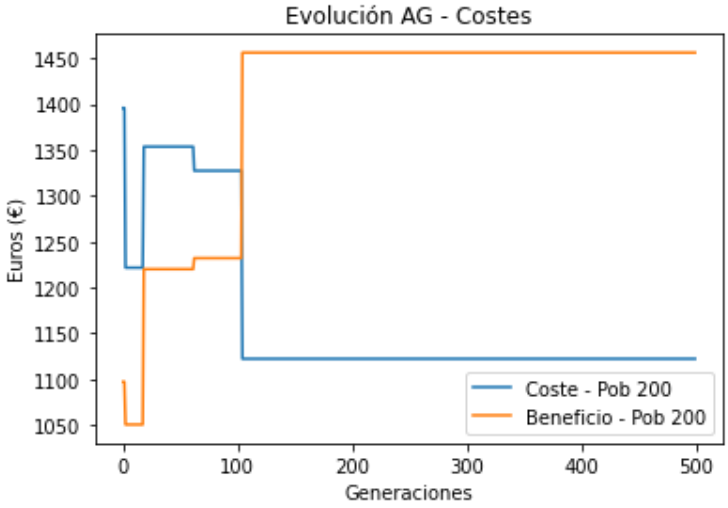
En esta ejecución se ha llegado a encontrar una solución con un Z=525 usando una población de 200 individuos. Podemos observar un gran incremento de Z después de la generación 100.



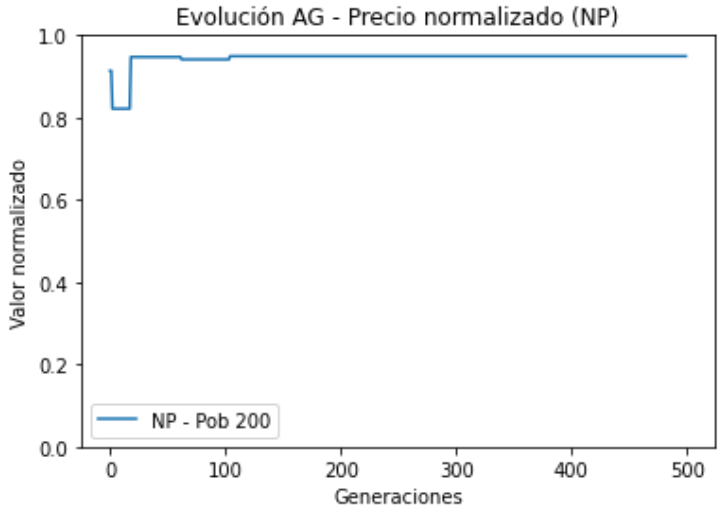
Este incremento se puede explicar por el descubrimiento por parte del algoritmo de una combinación de proveedores que, a la vez que mantiene la calidad percibida de los clientes (1), permite bajar mucho los costes de producción y aumentar los beneficios (2) sin que esto encarezca mucho el precio final del producto (3). De este modo, a la vez que obtenemos más beneficios por venta, podemos mantener las ventas (4).



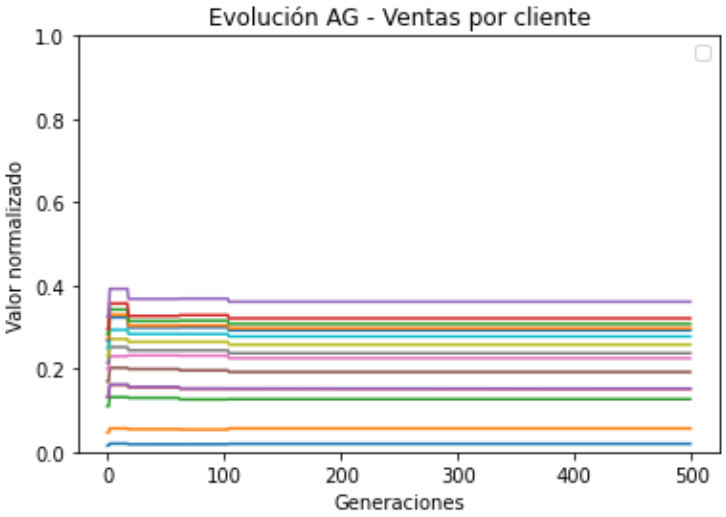
(1)



(2)



(3)



(4)

Después de estudiar cómo el algoritmo explora el espacio de soluciones podemos encontrar los siguientes patrones:

- Busca los proveedores que le permitan mantener o mejorar la calidad percibida por los clientes y a la vez bajar los costes de producción.
- Prefiere ofrecer productos de calidad media-alta con un alto margen de beneficio, aunque menos de la mitad de los clientes adquieran el producto.
- Tiende a ofrecer productos caros, cercanos al precio máximo de mercado.

Estos patrones sólo se aplican para entornos homogéneos, generados aleatoriamente. Estos patrones pueden cambiar ante entornos más realistas y heterogéneos.

Parámetros de evaluación

El algoritmo admite parámetros que permiten regular cómo el algoritmo explora el espacio de soluciones. Estos parámetros son:

- Tamaño de la población.
- Número de participantes en un torneo (k).
- Ratio de selección.

- Ratio de cruce.
- Ratio de mutación.

También podemos cambiar los precios máximo y mínimo de mercado, pero esto sólo afectará a la escala y variabilidad de los precios y del fitness.

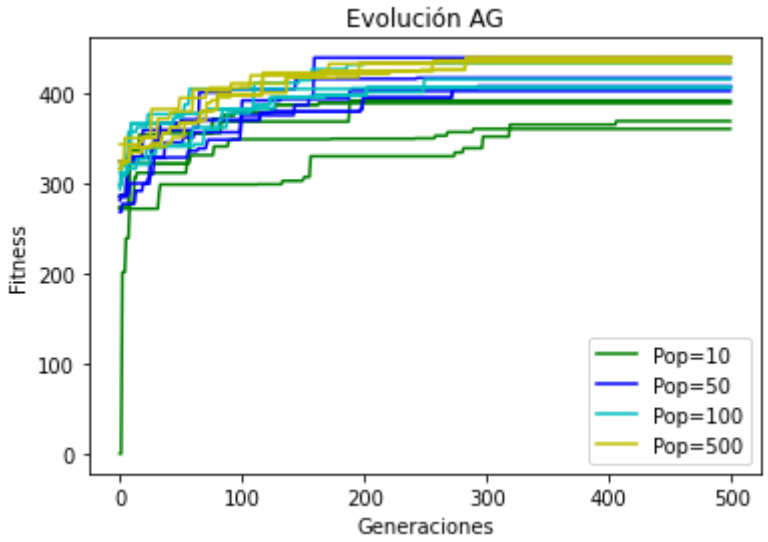
El criterio de terminación (número máximo de generaciones sin mejorar) es un parámetro de rendimiento que puede interrumpir la ejecución antes de alcanzar un óptimo y repercutir en la mejor solución final obtenida, pero no influye en cómo el algoritmo explora el espacio de soluciones.

Tamaño de la población

Se han realizado 4 ejecuciones por cada uno de los siguientes tamaños de población sobre el entorno por defecto:

$$\text{Población} = [10, 50, 100, 200, 500]$$

La siguiente gráfica muestra cómo evoluciona el algoritmo para los distintos tamaños de población probados.



De esta gráfica podemos extraer que tener una población grande hace que sea más probable que entre los individuos iniciales existan algunos cercanos a un óptimo local, lo que les permite converger hacia ese óptimo local en menos generaciones.

El mayor fitness se ha obtenido con un tamaño de población de 500 individuos (n_pop=500).

Tamaño de torneo

Se han realizado 4 ejecuciones por cada uno de los siguientes tamaños de torneo sobre el entorno por defecto:

$$K = [3, 5, 10, 20]$$

En teoría, conforme aumentamos el número de individuos que participan en un torneo, más elitista se vuelve nuestro algoritmo. Para el entorno utilizado, no hemos apreciado una gran diferencia en el rendimiento obtenido para cada valor de k.

El mayor fitness se ha obtenido con un tamaño de torneo de 3 individuos (K=3).

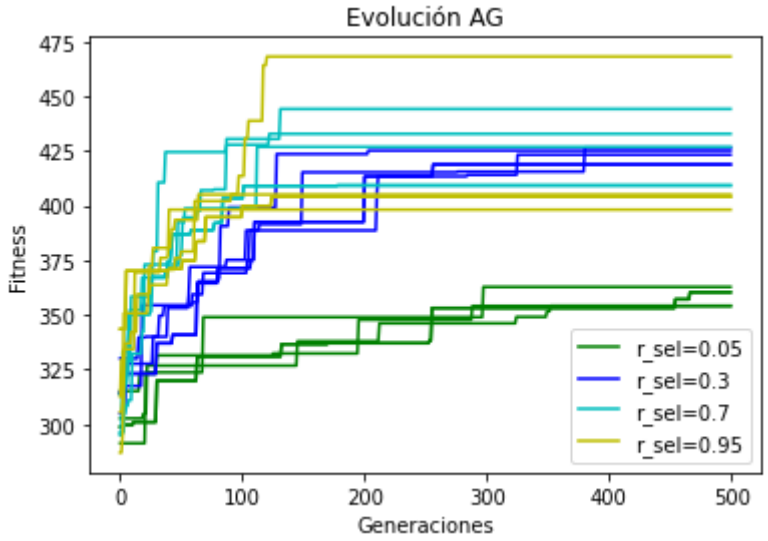
Ratio de selección

Se han realizado 4 ejecuciones por cada una de las siguientes ratios de selección sobre el entorno por defecto:

Ratio de selección = [5%, 30%, 70%, 95%]

Proporciones por debajo del 50% permitirán mantener individuos que, sin ser los peores, no han sido suficientemente buenos como para tener descendencia.

La siguiente figura muestra cómo evoluciona el algoritmo para distintas ratios de selección manteniendo el resto de los parámetros por defecto.



Podemos ver claramente como una ratio de selección muy baja impide explorar soluciones rápidamente al tener muy poca descendencia que poder cruzar y mutar. Por otro lado, tener una ratio de selección muy alto nos ha permitido llegar antes a soluciones mejores porque, al tener más descendencia que cruzar y mutar es más probable dar con una configuración que mejore mucho respecto a las demás.

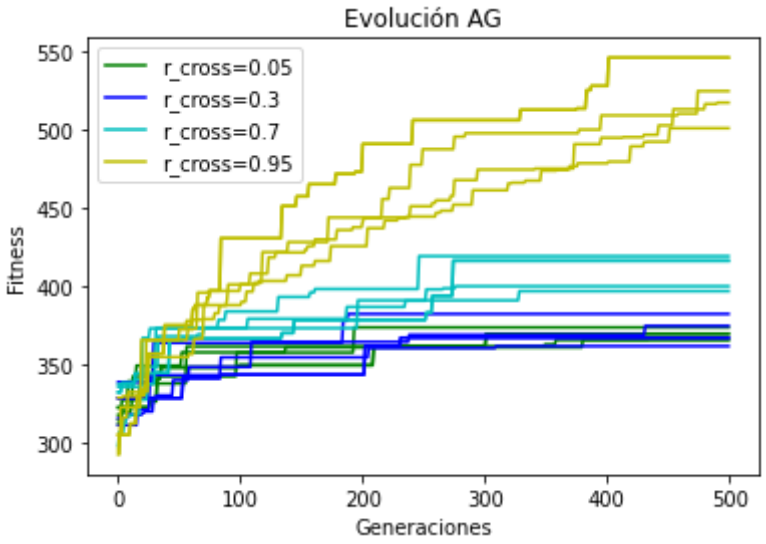
El mayor fitness se ha obtenido con una ratio de selección del 95% (r_sel=0.95).

Ratio de cruce

Se han realizado 4 ejecuciones por cada una de las siguientes ratios de cruce sobre el entorno por defecto:

Ratio de cruce = [5%, 30%, 70%, 95%]

La siguiente figura muestra cómo evoluciona el algoritmo para distintas ratios de cruce manteniendo el resto de los parámetros por defecto.



Podemos ver claramente que utilizar ratios de cruce muy altos permite conseguir soluciones mejores incluso después de la generación 300 donde el resto de las ejecuciones alcanzan la convergencia.

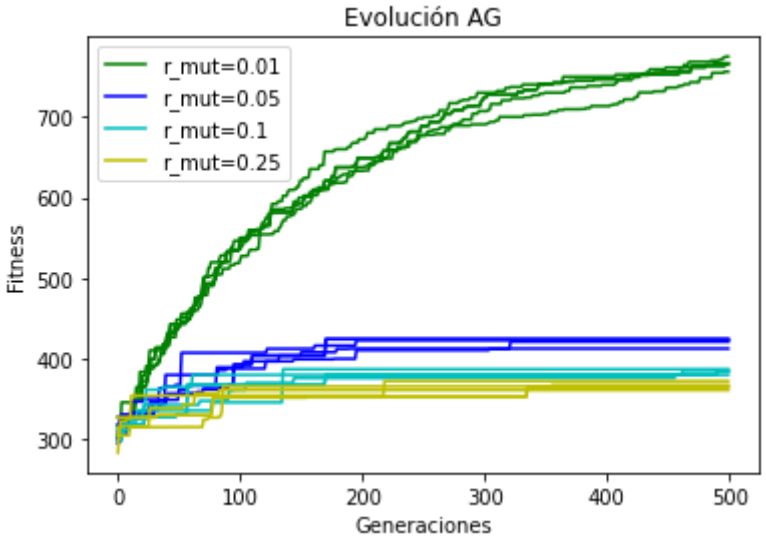
El mayor fitness se ha obtenido con una ratio de cruce del 95% (r_cross=0.95).

Ratio de mutación

Se han realizado 4 ejecuciones por cada una de las siguientes ratios de mutación sobre el entorno por defecto:

Ratio de mutación = [1%, 5%, 10%, 25%]

La siguiente figura muestra cómo evoluciona el algoritmo para distintas ratios de mutación manteniendo el resto de los parámetros por defecto.



Claramente podemos ver que emplear ratios de mutación muy pequeños (1%) mejora significativamente la exploración del espacio de soluciones permitiendo al algoritmo alcanzar óptimos que son posiblemente globales y a los que no se ha podido llegar con ratios de mutación mayores.

El mayor fitness se ha obtenido con una ratio de mutación del 1% (r_mut=0.01).

Conclusiones

La parametrización óptima que hemos encontrado para el algoritmo de enfriamiento simulado es:

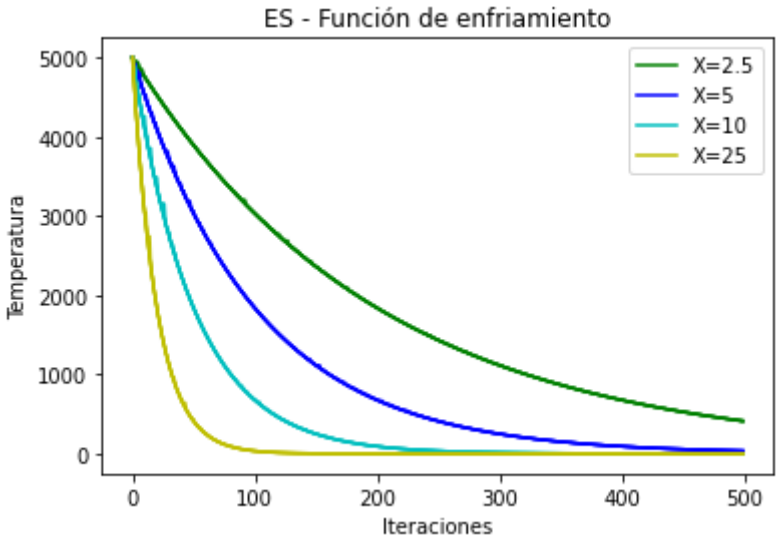
- Tamaño de la población: 500
- Número de participantes en un torneo (k): 3
- Ratio de selección: 95%
- Ratio de cruce: 95%
- Ratio de mutación: 1%

Enfriamiento simulado

Diseño

Para resolver este problema a través de un algoritmo de enfriamiento simulado buscaremos vecinos por mutación. En concreto, usaremos la misma función de mutación que hemos utilizado para el algoritmo genético, pero ligando la ratio de mutación a la temperatura de modo que esta ratio llegará a 0 cuando la temperatura llegue a 0 en la última iteración. La ratio de mutación inicial que se aplicará en la búsqueda de vecinos será un parámetro de nuestro algoritmo.

Usaremos una bajada de temperatura exponencial donde esa componente exponencial será uno de los parámetros de nuestro algoritmo. La siguiente gráfica muestra cómo la temperatura desciende en cada iteración para las diferentes intensidades de enfriamiento.



La temperatura inicial también representará un parámetro que podemos cambiar en nuestro algoritmo.

Como criterio de parada hemos establecido, de nuevo, un número máximo de iteraciones sin mejora y además la condición de que la temperatura sea mayor que 0. En el momento en el que alguna de estas condiciones no se cumpla, la ejecución terminará.

Por último, la implementación de este algoritmo soporta retroalimentación, es decir, al completarse una ejecución (temperatura mínima) se puede repetir la ejecución (de nuevo con temperatura máxima) utilizando como solución inicial la mejor solución encontrada en la ejecución anterior. El número de retroalimentaciones del algoritmo es un parámetro que podemos controlar.

Evaluación

Los parámetros del algoritmo tendrán los siguientes valores por defecto:

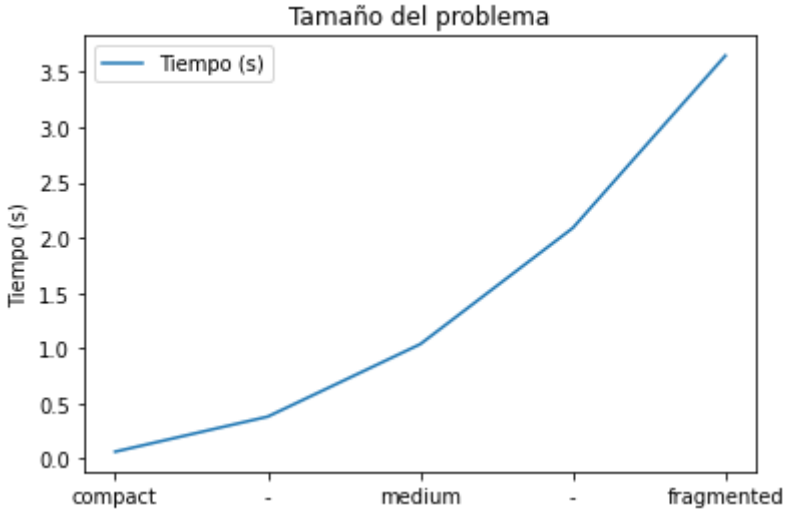
- Número de iteraciones: 500
- Máximo número de generaciones sin mejora: -1
- Temperatura inicial: 500
- Intensidad del enfriamiento: 10
- Ratio de mutación: 5%
- Número de retroalimentaciones: 0

Por defecto, no activaremos el criterio de terminación para comparar resultados con el mismo número de generaciones (n_convg = -1).

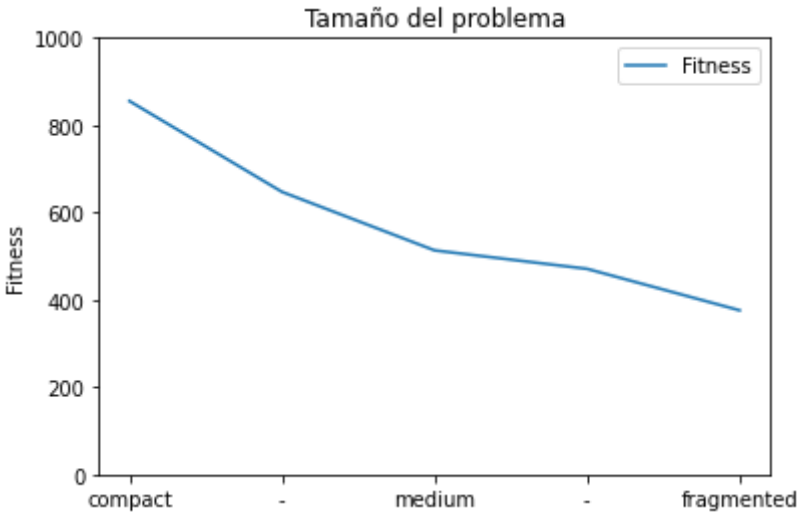
Tamaño del problema

Se ha evaluado cómo afecta el tamaño del problema al rendimiento del algoritmo de enfriamiento simulado. De nuevo, se ha ejecutado el algoritmo con los parámetros por defecto sobre los entornos 'Compacto', 'Medio' y 'Fragmentado'.

De nuevo, el tiempo de cómputo crece de forma exponencial con el tamaño del problema como se puede ver claramente en la siguiente gráfica. Sin embargo, la escala de tiempo es 2 órdenes de magnitud más pequeña que en el algoritmo genético. Para el tamaño de problema más grande probado la ejecución ha tenido una duración de 3,5 segundos.



De nuevo, el fitness máximo al que logra llegar el algoritmo decrece con el tamaño del problema como podemos ver en la siguiente gráfica.



Parámetros de evaluación

El algoritmo admite parámetros que permiten regular cómo o en qué intensidad el algoritmo explora el espacio de soluciones. Estos parámetros son:

- Temperatura inicial.
- Intensidad del enfriamiento.
- Ratio de mutación.
- Número de retroalimentaciones.

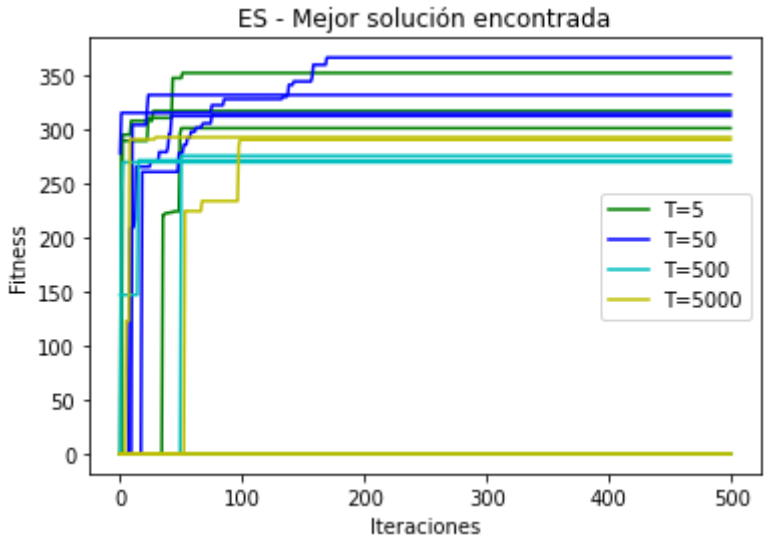
Por las mismas razones que la expuestas en la evaluación del algoritmo genético, no modificaremos los precios máximo y mínimo de mercado ni el criterio de terminación.

Temperatura inicial

Se han realizado 4 ejecuciones por cada una de las siguientes temperaturas iniciales sobre el entorno por defecto:

Temperatura inicial = [5, 50, 500, 5000]

La siguiente figura muestra cómo evoluciona el algoritmo para distintas temperaturas iniciales manteniendo el resto de los parámetros por defecto.



Como vemos en la gráfica, utilizar una temperatura inicial de 50º permite alcanzar una solución muy buena en menos de 100 iteraciones.

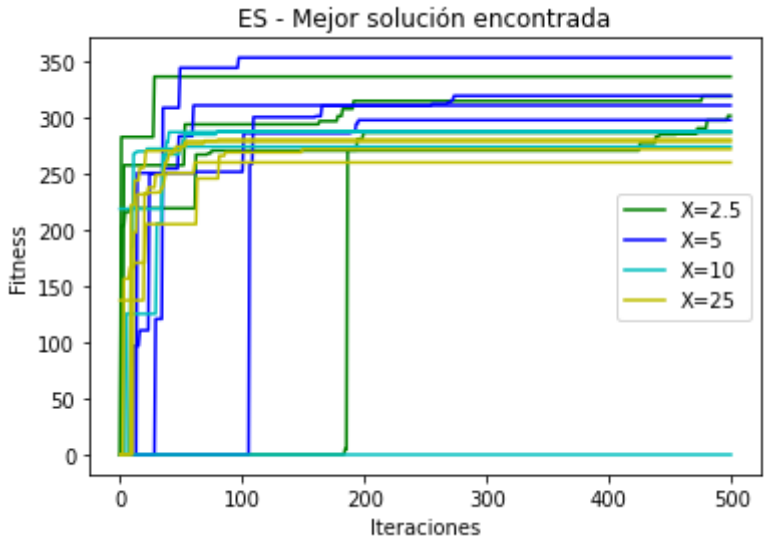
El mayor fitness se ha obtenido con una temperatura inicial de 50º (max_t=50).

Intensidad del enfriamiento

Se han realizado 4 ejecuciones por cada una de las siguientes intensidades de enfriamiento sobre el entorno por defecto:

Intensidad de enfriamiento = [2.5, 5, 10, 25]

La siguiente figura muestra cómo evoluciona el algoritmo para las distintas intensidades de enfriamiento manteniendo el resto de los parámetros por defecto.



Como vemos en la gráfica, utilizar intensidades bajas (2.5 y 5) permite alcanzar soluciones mejores y en menos iteraciones que utilizando el resto de los valores.

El mayor fitness se ha obtenido con una intensidad de enfriamiento de valor 5 (cooling=5).

Ratio de mutación

Se han realizado 4 ejecuciones por cada una de las siguientes ratios de mutación sobre el entorno por defecto:

Ratio de mutación = [1%, 5%, 10%, 25%]

Para el entorno utilizado, no hemos apreciado una gran diferencia en el rendimiento obtenido para cada ratio de mutación.

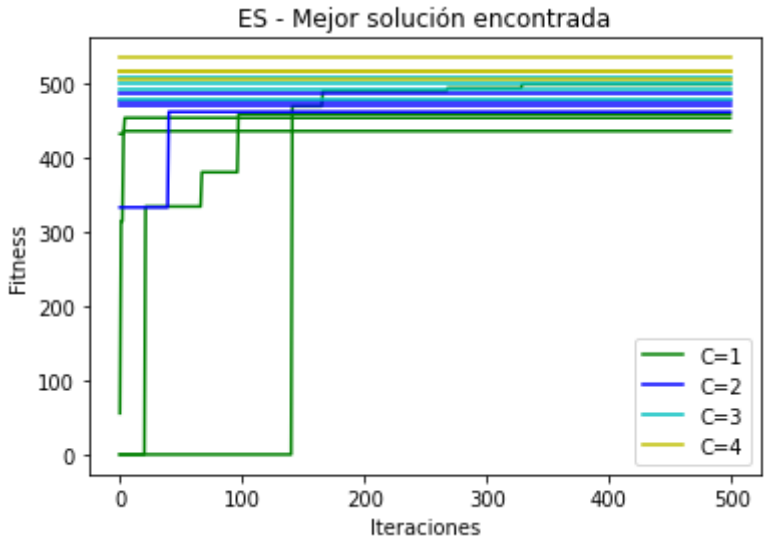
El mayor fitness se ha obtenido con una ratio de mutación del 5% (r_mut=0.05).

Número de retroalimentaciones

Se han realizado 4 ejecuciones por cada uno de los siguientes números de retroalimentaciones sobre el entorno por defecto:

Número de retroalimentaciones = [1, 2, 3, 4]

La siguiente figura muestra cómo evoluciona el algoritmo para las distintas intensidades de enfriamiento manteniendo el resto de los parámetros por defecto.



Este parámetro aumenta exponencialmente el tiempo de cómputo requerido para completar una ejecución y, como podemos ver en la gráfica, ya en el primer ciclo se puede alcanzar un fitness muy bueno que es mejorado ligeramente en ejecuciones con varios ciclos de retroalimentación.

El mayor fitness se ha obtenido con 4 ciclos (3 retroalimentaciones) pero la mejora es baja en relación con el tiempo extra que toma llegar hasta ella de modo que elegiremos no utilizar retroalimentación.

Conclusiones

La parametrización óptima que hemos encontrado para el algoritmo de enfriamiento simulado es:

- Temperatura inicial: 50º
- Intensidad del enfriamiento: 5
- Ratio de mutación: 5%
- Número de retroalimentaciones: 0

Comparativa

Se han realizado 4 ejecuciones sobre el entorno por defecto utilizando la configuración óptima para ambos algoritmos. La siguiente tabla muestra los resultados medios obtenidos de estas ejecuciones:

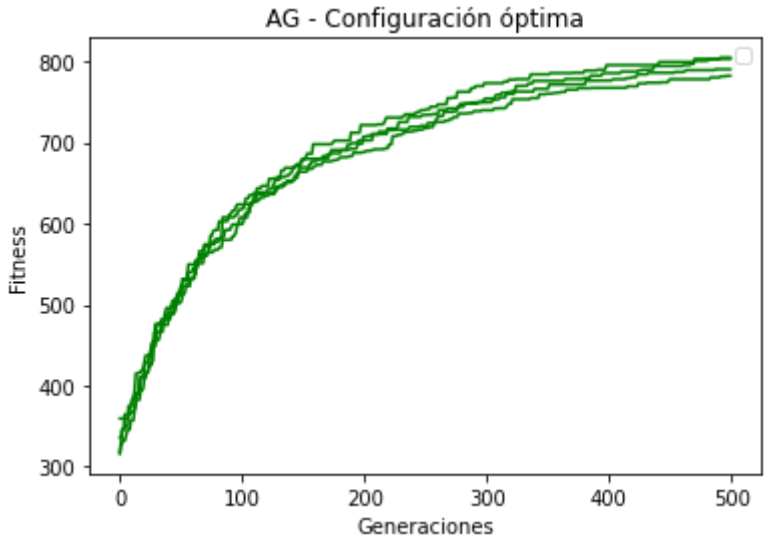
Algoritmo	**Iteraciones**	**Máximo fitness**	**Tiempo medio (segundos)**
Algoritmo Genético	500	804,541	244,075
Enfriamiento Simulado	500	228,415	0,554

Como podemos apreciar, la calidad de la solución y el tiempo empleado son muy dispares en ambos algoritmos. El poco tiempo de cómputo en enfriamiento simulado puede ser el causante de la baja calidad de las soluciones obtenidas.

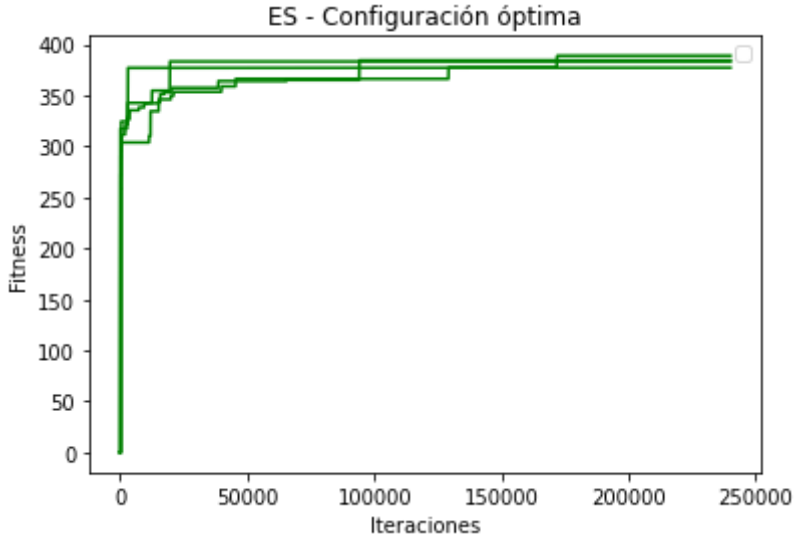
Hemos repetido el experimento, pero utilizando 240000 iteraciones para el algoritmo de enfriamiento simulado. La siguiente tabla muestra los resultados medios obtenidos tras este cambio:

Algoritmo	**Iteraciones**	**Máximo fitness**	**Tiempo medio (segundos)**
Algoritmo Genético	500	804,541	244,075
Enfriamiento Simulado	240000	383,384	243,910

La siguiente figura muestra cómo evolucionan el algoritmo genético (1) y el de enfriamiento simulado (2) en este segundo experimento.



(1)



(2)

En vista de estos resultados, resolver este problema a través de un algoritmo genético resulta la mejor opción. El algoritmo de enfriamiento simulado encuentra soluciones con un fitness aceptable en muy poco tiempo, pero converge demasiado pronto sin llegar a alcanzar soluciones mejores que sí son exploradas por el algoritmo genético.

Conclusiones

Hemos modelado el problema e implementado algoritmos genéticos y de enfriamiento simulado que permiten resolverlo. Después hemos comprobado cómo afectan los distintos parámetros de cada algoritmo al rendimiento en la resolución del problema. Finalmente, hemos comprobado cuál de estos algoritmos, en su mejor configuración, logra resolver mejor el problema propuesto.

En cuanto al algoritmo genético, hemos comprobado que un tamaño de población grande permite alcanzar buenas soluciones en las iteraciones iniciales y que resulta conveniente utilizar ratios altas (~95%) de selección y cruce, pero ratios bajas de mutación (~1%).

En cuanto al algoritmo de enfriamiento simulado, hemos comprobado que para los parámetros temperatura inicial e intensidad del enfriamiento conviene utilizar valores intermedios, en concreto 50 y 5 respectivamente. También hemos comprobado que utilizar retroalimentación permite encontrar mejores soluciones, pero a un coste en tiempo de cómputo alto.

Al comparar estos algoritmos para el problema propuesto hemos comprobado que el algoritmo genético alcanza soluciones mejores, mientras que enfriamiento simulado alcanza soluciones aceptables en muy poco tiempo. Sin embargo, extender la ejecución del algoritmo de enfriamiento simulado para que tome tanto tiempo como el genético no repercute en encontrar mejores soluciones.

Si el objetivo fuera encontrar la solución más cercana a la óptima (caso más habitual), diríamos que utilizar un algoritmo genético sería la opción más recomendada.