Vortex Search

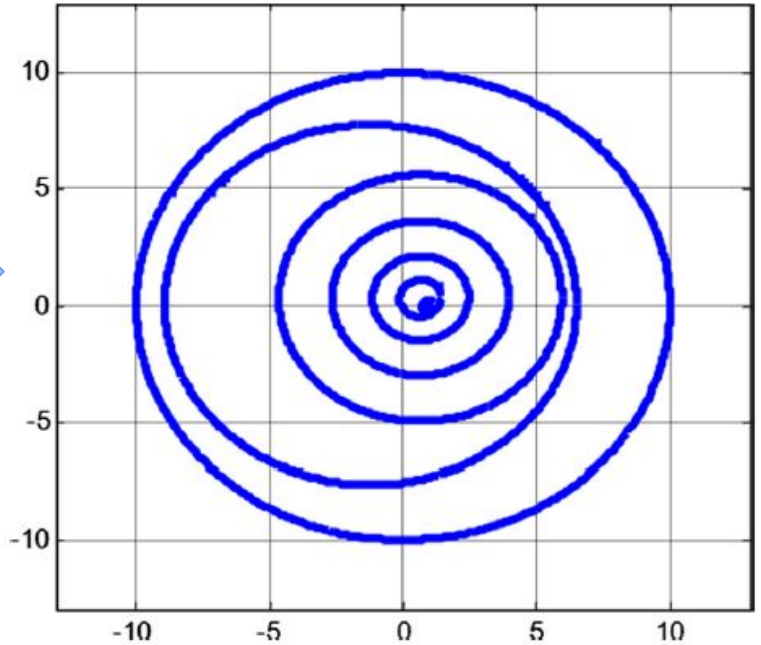
Germán González

Concepto

- Búsqueda por trayectorias simples que introduce diversidad mediante un mecanismo que imita los vórtices de agua.
- Autores: Berat Dogan y Tamer Olmez, profesores de la universidad técnica de Estambul.
- Basada en las ecuaciones de Navier-Stokes, que describen el comportamiento de los fluidos bajo entornos inestables (turbulentos).

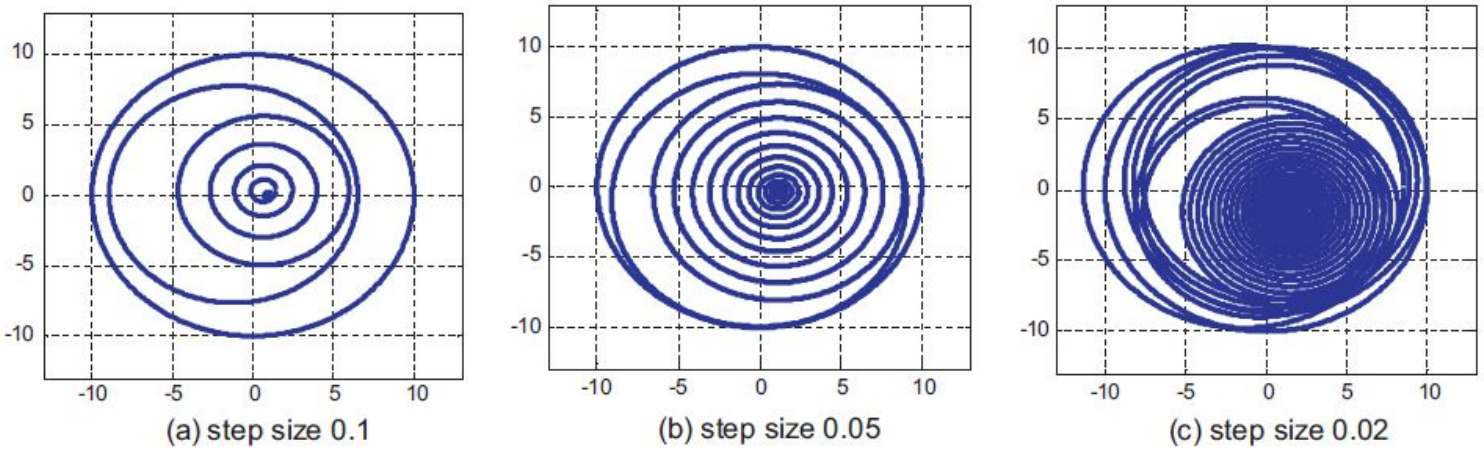
Modelo → Formalización

Navier-Stokes



Formalización

- La circunferencias mayores corresponden a los segmentos superficiales del vórtice, las menores corresponden a los segmentos más profundos.
- La variación del radio de la hiperesfera viene dada por un parámetro *step size*, y es calculado mediante la función gamma incompleta inversa.



(a) step size 0.1 (b) step size 0.05 (c) step size 0.02

Adaptación de la formalización

- En cada iteración se selecciona como nuevo centro de la búsqueda el mejor individuo hasta el momento

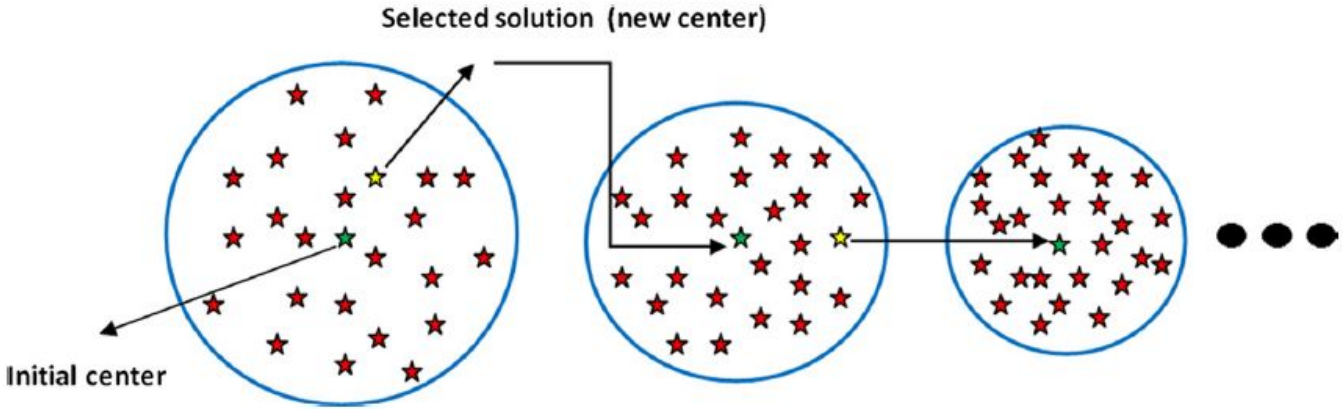
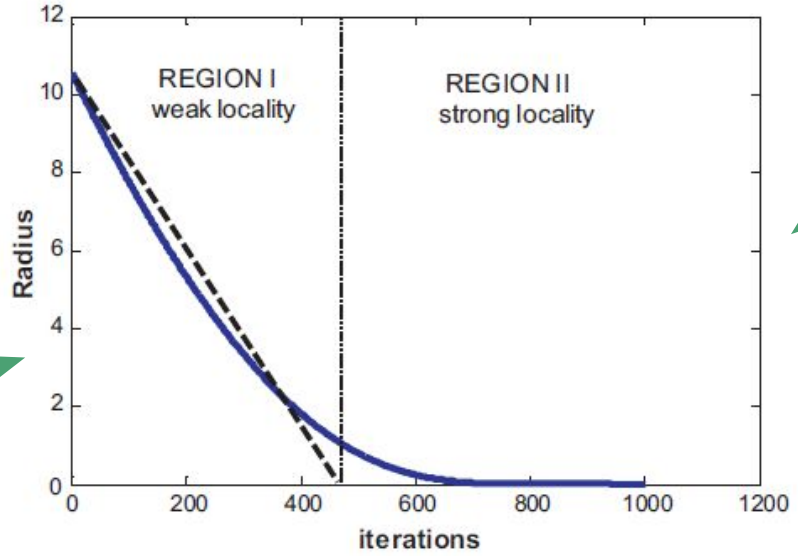


Fig. 2. An illustrative sketch of the search process.

Relevancia *step size*

- El parámetro *step size* controla la fase inicial de exploración.



Explotación

Exploración

Pseudocódigo

Ajuste de vecinos según radio actual

Actualización de la mejor solución

Asignación de nuevo centro

Decremento del radio

Inputs: Initial center μ_0 is calculated using Eq. 1
Initial radius r_0 (or the standard deviation, σ_0) is computed using Eq. 4
Fitness of the best solution found so far $f(s_{best}) = \inf$

$t = 0$;
Repeat
/* Generate candidate solutions by using Gaussian distribution around the center μ_t with a standard deviation (radius) r_t */
Generate($C_t(s)$) ;
If exceeded, then shift the $C_t(s)$ values into the boundaries as in Eq. 5
/* Select the best solution from $C_t(s)$ to replace the current center μ_t */
$s' =$ Select($C_t(s)$) ;
if $f(s') < f(s_{best})$
$s_{best} = s'$
$f(s_{best}) = f(s')$
else
keep the best solution so far s_{best}
end
/* Center is always shifted to the best solution found so far */
$\mu_{t+1} = s_{best}$
/* Decrease the standard deviation (radius) for the next iteration */
$r_{t+1} =$ Decrease(r_t)
$t = t + 1$;
Until the maximum number of iterations is reached
Output: Best solution found so far s_{best}

Experementación

- Para la experimentación se han considerado algunas de las funciones de la competición de 2005
- Se han obtenido resultados para las funciones mencionadas en dimensiones 10 y 30
- Las siguientes tablas recogen los resultados de las ejecuciones, contiene el valor del óptimo, el obtenido mediante VS y la desviación

Experimentación: 10 dimensiones

Vortex Search (Dimension 10)							
Función	**Valor**	**Optimo**	**Desv.**	**Función**	**Valor**	**Optimo**	**Desv.**
F1	-450.0	-450.0	0.0	F10	-308.11	-330.0	6.633
F2	-450.0	-450.0	0.0	F11	91.72	90.0	1.9162
F5	-285.62	-310.0	7.8655	F13	-127.47	-130.0	1.942
F6	395.507	390.0	1.4121	F14	-296.86	-300.0	1.046
F8	-119.99	-140.0	14.288	F17	166.44	120.0	38.7
F9	-314.08	-330.0	4.824	F24	460.0	260.0	76.9231

Experimentación: 30 dimensiones

Vortex Search (Dimension 30)							
Función	**Valor**	**Optimo**	**Desv.**	**Función**	**Valor**	**Optimo**	**Desv.**
F1	-449.98	-450.0	0.0042	F10	-121.97	-330.0	63.04
F2	-439.55	-450.0	2.32	F11	118.98	90.0	32.2
F5	11348.48	-310.0	3760.80	F13	-109.67	-130.0	15.63
F6	25200.59	390.0	6361.69	F14	-287.03	-300.0	4.32
F8	-119.51	-140.0	14.63	F17	113.646	120.0	5.29
F9	-198.58	-330.0	39.822	F24	460.03	260.0	76.93

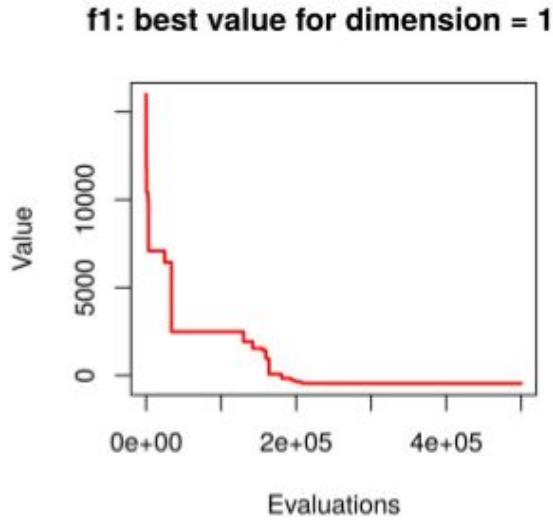
Análisis de Resultados

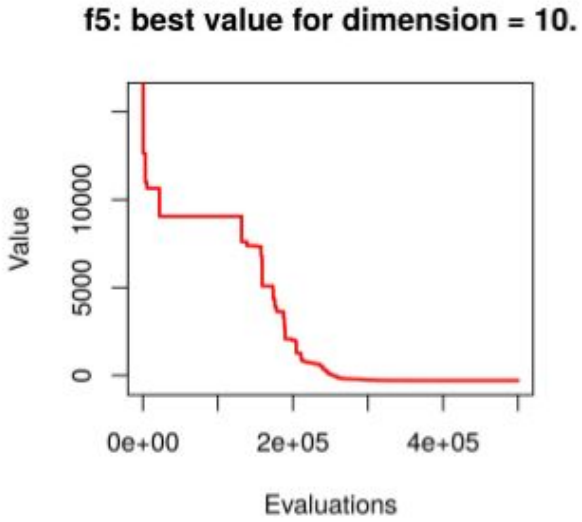
La búsqueda por vórtices obtiene el óptimo para el problema en los casos en los que este está bien diferenciado, ya sea porque no hay óptimos locales, o porque estos están lo suficientemente alejados del óptimo local como para interferir en el proceso de búsqueda. Por otra parte, en las funciones más complejas los resultados obtenidos son mejorables, debemos recordar que la búsqueda por vórtices es un algoritmo basado en trayectorias simples y por tanto tiene dificultades al encontrar óptimos globales cuando estos están rodeados por óptimos locales.

Cuando aumentamos el número de dimensiones, el algoritmo sigue dando buenos resultados para las funciones con un óptimo global bien diferenciado. Sin embargo, en algunos casos este aumento en la complejidad de la función a optimizar hace que el algoritmo no sea suficientemente potente como para optimizar funciones que antes si podía.

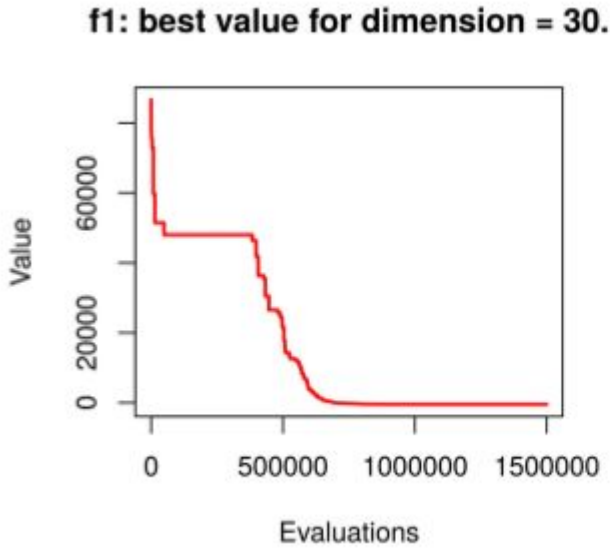
Comportamiento del algoritmo

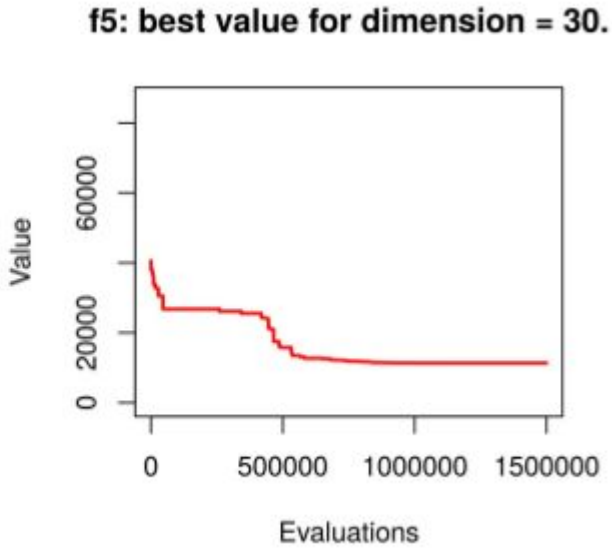
- Podemos representar en una gráfica el mejor valor encontrado frente al número de evaluaciones





El mejor valor encontrado por el algoritmo en cada momento disminuye con el paso de las iteraciones, aunque durante algunos periodos se mantiene estable; esto se debe a que en las fases de exploración el algoritmo puede no aceptar una nueva solución como solución actual si ninguna de ellas la mejora, por ello el valor es siempre decreciente y nunca ascendente. Observamos un comportamiento similar al aumentar las dimensiones:





Mejora: Combinación con Ideology Algorithm

Ante todo agradezco la colaboración de mi compañero Nestor Rodriguez Vico, que ha desarrollado el código de Ideology Algorithm y ha colaborado conmigo en la realización de esta sección.

La combinación de los algoritmos de Búsqueda por Vórtices e Ideology Algorithm consiste en emplear la búsqueda por vórtices como mejora local para los individuos de la población, es decir, los políticos, obteniendo como resultado un algoritmo memético. Puesto que, para el problema que nos atañe, los políticos no son más que puntos en el espacio, podemos dar el punto en el espacio asociado al político como centro inicial para la búsqueda por vórtices, de esta forma lanzamos una búsqueda por trayectorias simples en el área del espacio que rodea al político. Los resultados obtenidos con la combinación de estos algoritmos se muestra en las siguientes transparencias:

Mejora: Experimentación en 10 dimensiones

Algoritmo Combinado (Dimension 10)							
Función	Valor	Optimo	Desv.	Función	Valor	Optimo	Desv.
F1	-449.9	-450.0	0.0221	F10	-321.1	-330.0	2.72
F2	-449.1	-450.0	0.188	F11	91.95	90.0	2.172
F5	81.12	-310.0	126.169	F13	-129.3	-130.0	0.53
F6	399.798	390.0	2.5127	F14	-297.87	-300.0	0.71
F8	-119.97	-140.0	14.3	F17	163.946	120.0	36.62
F9	-324.86	-330.0	1.556	F24	460.04	260.0	76.93

Mejora: Experimentación en 30 dimensiones

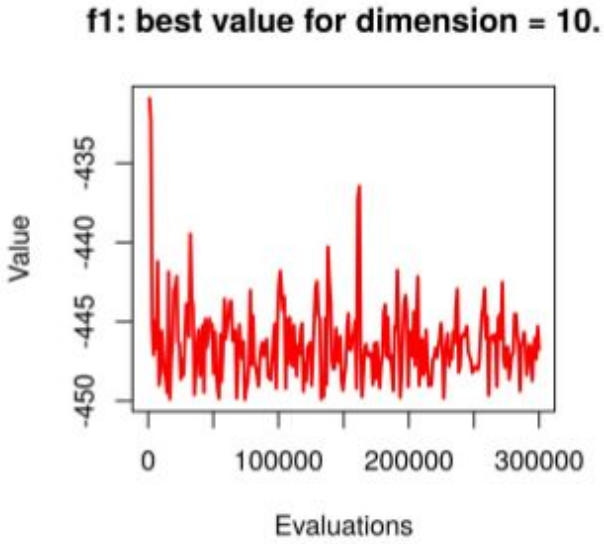
Algoritmo Combinado (Dimension 30)							
Función	Valor	Optimo	Desv.	Función	Valor	Optimo	Desv.
F1	82.87	-450.0	118.41	F10	-148.32	-330.0	55.1
F2	2256.158	-450.0	601.37	F11	114.37	90.0	27.07
F5	6536.26	-310.0	2208.5	F13	-111.19	-130.0	14.472
F6	1675756	390.0	429581	F14	-288.1	-300.0	3.9674
F8	-119.67	-140.0	14.52	F17	X	X	X
F9	-164	-330.0	50.3	F24	X	X	X

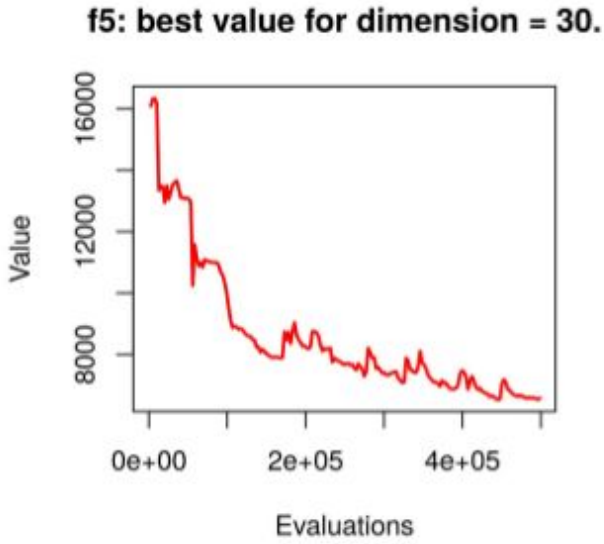
Análisis de Resultados del algoritmo memético

A la vista de los resultados podemos decir que, claramente la combinación de estos dos algoritmos no parece ser la ideal para resolver algunos de los problemas de optimización que la búsqueda por vórtices es capaz de resolver, sin embargo, y aunque estos casos sean la menoría, la combinación parece presentar mejora, por ejemplo, para las funciones F9 y F10.

Comportamiento del algoritmo memético

- Podemos representar en una gráfica el mejor valor encontrado frente al número de evaluaciones





Análisis del comportamiento

El mejor valor obtenido disminuye con el paso de las iteraciones, la diferencia más notable que encontramos en estas gráficas respecto a las de la búsqueda por vórtices es que, al contrario de lo que sucedía en esta última, Ideology Algorithm es capaz de explorar soluciones peores a la actual en cada iteración. Además cabe destacar que el comportamiento del algoritmo depende en gran medida de la función considerada para la optimización, es por ello que vemos grandes fluctuaciones el mejor valor en la función 1, al contrario de lo que sucede en la función 5.

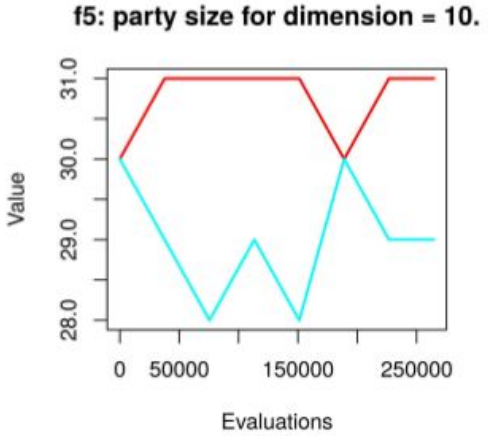
Mejoras consideradas para el algoritmo memético

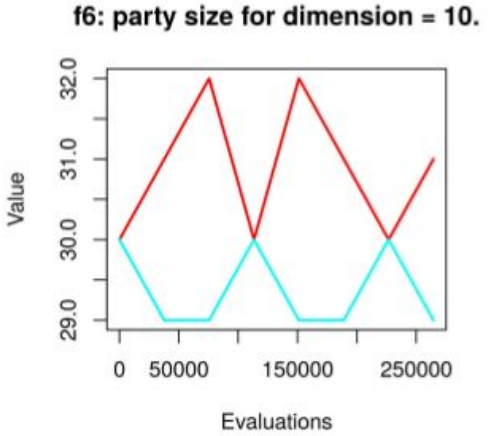
Una de las mejoras inmediatas que podemos considerar para un algoritmo memético es el reinicio de la población cuando detectamos que esta converge a un valor. Por tanto, podemos aplicar esta técnica al algoritmo desarrollado en la sección anterior. Sin embargo, tras la experimentación, la conclusión es que, dadas las características del algoritmo, así como de las funciones a optimizar, no es posible establecer un patrón para la identificación del umbral del reinicio de la población; de esta forma, los resultados obtenidos son, en general, peores que los obtenidos con el algoritmo memético básico.

Dadas las características del algoritmo consideramos como reinicio de la población la generación de nuevos individuos aleatorios, en concreto, 150 de ellos, y su posterior agrupación en partidos del mismo tamaño, a saber, 30. Teniendo esto en consideración, planteamos tres técnicas de reinicio de la población, detalladas en las siguientes transparencias.

Reinicio por diferencia de tamaño de partidos

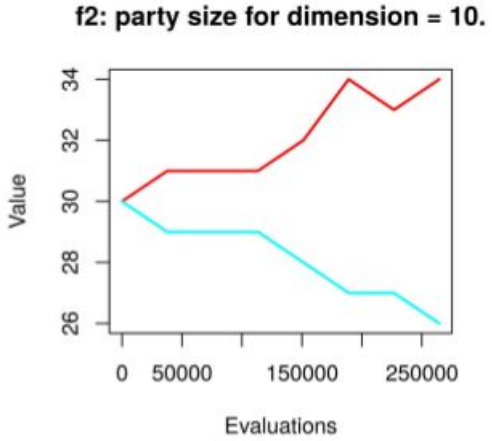
Esta técnica consiste en reiniciar la población cuando se detecta que el partido mayoritario es un tanto por ciento mayor que el minoritario. Los resultados obtenidos con esta técnica son, en el mejor de los casos iguales a los obtenidos sin ella. Podemos obtener una representación visual de lo que le sucede a la población durante el proceso de búsqueda representando los tamaños de los partidos mayoritarios y minoritarios frente al número de evaluaciones:

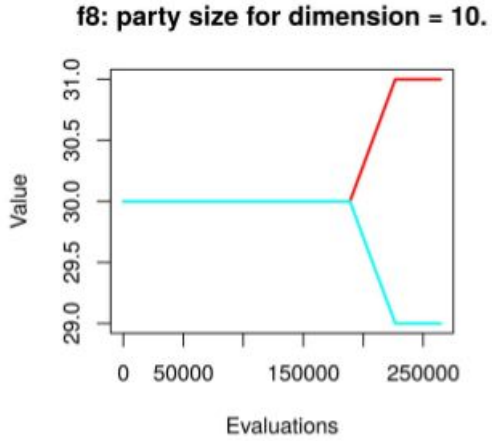




Reinicio por estabilización del mejor individuo

Esta técnica consiste en reiniciar la población cuando se detecta que el mejor individuo se estanca, es decir, no mejora con el paso de las iteraciones, lo que se traduce en que el proceso ha pasado de ser de exploración a ser de explotación. De igual forma que sucedía en el caso anterior, esta técnica no mejora el comportamiento del algoritmo, representaremos gráficamente lo que le sucede a la población durante el proceso de búsqueda:





Reinicio por convergencia de la población

Esta técnica consiste en reiniciar la población cuando se detecta que la diferencia entre el mejor individuo global y el peor individuo global es menor que un umbral, lo que se traduce en una convergencia de la población en general, es decir, consideramos la población como un único conjunto que nos proporciona los marcadores de reinicio, al contrario de los criterios empleados en las técnicas anteriores, que consideraban los partidos como marcadores. Una vez más los resultados obtenidos no mejora a los obtenidos con el algoritmo memético básico, por razones semejantes a las ya comentadas.

Conclusiones sobre las mejoras consideradas

A la vista de los resultados podemos concluir que, como era de esperar, la elección del umbral que supone la reinicialización de la población es determinante para el comportamiento del algoritmo, de esta forma, establecer un umbral demasiado bajo supone que la población no se reinicie y por tanto obtengamos los mismos resultados que con el algoritmo memético básico; por otra parte, establecer un umbral demasiado alto supondría, en el peor de los casos, un reinicio constante de la población, lo que, además de un sobrecoste computacional, supone cambiar radicalmente el comportamiento del algoritmo de forma que este no es capaz de explorar apropiadamente un espacio de soluciones en el que encontrar la mejor.