Evaluación de dos nuevos algoritmos en el diseño de granjas eólicas

Fabricio Loor, Guillermo Leguizamón y Javier Apolloni

Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)
Departamento de Informática - FCFMN
Universidad Nacional de San Luis, Argentina
loorfabricio@gmail.com,{legui,javierma}@unsl.edu.ar

Resumen En los últimos años el crecimiento en el consumo de energía eléctrica ha sido exorbitante, lo cual ha generado la necesidad de utilizar un recurso prometedor como el viento para extraer dicha energía. La distribución de turbinas de viento dentro de una granja eólica, con el objeto de optimizar la energía capturada, es un problema complejo de resolver. En este artículo se intenta solucionar este problema abordándolo de dos formas distintas: una es la adaptación del algoritmo GWO para vectores booleanos y la otra, DonQuijote, es un método nuevo que incluye el uso de la Evolución Diferencial y surge del análisis del problema. Para mostrar la eficiencia de los métodos se comparan con un Algoritmo Genético, tan estudiado en el área. La mejor propuesta participó en la competencia WFLO de la GECCO 2015.

Palabras claves: energía eólica, diseño de granjas eólicas, optimización, evolución diferencial, metaheurísticas

1. Introducción

El diseño de granjas eólicas es un tema muy estudiado [1] y aún sigue en auge principalmente por las siguientes razones:

- Las energías renovables han sido un recurso ampliamente requerido y con un futuro prometedor.
- Son fuentes seguras y ecológicas para satisfacer la demanda energética actual.
- Encontrar la disposición óptima de las turbinas en una zona reservada para una granja eólica es un problema muy complejo. El número desmesuradamente grande de posibles lugares dentro de la granja para colocarlas lo convierte en un problema intratable. Claramente, el gran tamaño del espacio de búsqueda hace imposible utilizar un algoritmo de optimización mediante una búsqueda exhaustiva. Por este motivo, es necesaria otra clase de métodos de optimización. En este sentido, existe un gran repertorio de métodos y es importante analizar cuál es la mejor posibilidad.

De acuerdo a dónde y cuántas turbinas eólicas son ubicadas se podrá obtener grandes beneficios económicos. Una mala distribución puede disminuir significativamente la energía que las turbinas, en su conjunto, producen. Por esto resulta muy conveniente aplicar métodos estocásticos que permitan encontrar una solución provechosa en tiempos de ejecución aceptables, sacrificando la exactitud de la solución.
Como base para probar los métodos, se ha hecho uso de un Algoritmo Genético

(GA) simple, el cual fue dado como *baseline* en la competencia WFLO de GECCO 2015 [11]. El algoritmo dado en la competencia no tiene en cuenta la información del problema, por lo tanto, son admisibles muchas mejoras. En este artículo, se ha desarrollado un enfoque competitivo de un algoritmo relativamente complejo que considera las influencias entre las turbinas. Los resultados de la simulación demuestran la eficacia de la mejora.
Dentro de la literatura se pueden encontrar varios algoritmos considerados computacionalmente eficientes para resolver este problema, por ejemplo, GA[2], CMA-ES [3] y TDA [4] , DEVO-I [5], DEVO-II [6], PSO [7]. En el presente trabajo, se ha utilizado un algoritmo que incluye Evolución Diferencial (DE) para encontrar la mejor configuración de acuerdo a la zona de ubicación de las turbinas. Por lo tanto, se exhibe un nuevo enfoque a este problema. La sección 2 proporciona una visión general al problema de distribución de turbinas y los costos involucrados. En la sección 3 se describen los algoritmos propuestos. En la sección 4 se presentan los resultados experimentales y el análisis respectivo. La sección 5 incluye las conclusiones y posibles trabajos futuros.

2. Descripción del problema

Suponiendo un plano cartesiano para distribuir las turbinas y considerando que el plano admite coordenadas con valores continuos, la ubicación de cada turbina se indica mediante un par de variables que establecen una posición dentro de este plano. Los límites del plano son dados como parámetros. El plano también es conocido como escenario. Una solución al problema sería, por lo tanto, una secuencia de puntos (x, y) que satisfacen con estar dentro de los límites del escenario y, por condiciones de seguridad, cada punto se encuentra a una distancia fija de los restantes. Esta distancia fija es cuatro veces el tamaño del rotor de una turbina.
La competencia adopta como esquema para representar los escenarios el que se describe en [8]. El viento es simulado mediante un modelo homogéneo, es decir, todas las zonas del escenario reciben el mismo viento, excepto que exista una turbina que interfiera en la captación de la energía. Si esto último sucede, se utiliza una simulación del comportamiento del viento de manera estocástica mediante la distribución Weilbull (ver más detalles en [8]).
Además, las turbinas son ubicadas de forma tal que el rotor de las mismas esté orientado perpendicularmente a la dirección del viento [8].

2.1. Restricciones

Los escenarios cuentan con zonas prohibidas, llamadas obstáculos, para la ubicación de las turbinas. Los obstáculos son modelados como rectángulos, y una solución es considerada inválida si tiene una turbina dentro de algún obstáculo. En secciones posteriores se representa gráficamente un obstáculo.

2.2. Modelo del costo

Para determinar qué tan buena es una solución propuesta se sigue el estudio propuesto para la competencia WFLO 2015 [11], donde la función de fitness tiene dos variables confrontadas: la cantidad de turbinas y la potencia producida por la

configuración.

$$fitness = \frac{(c_t * n + c_s * floor(\frac{n}{m}))(\frac{2}{3} + \frac{1}{3} * e^{-0,00174 * n^2}) + c_{OM} * n}{\frac{(1-(1+r)^{-y})}{r}} * \frac{1}{8760 * P} + \frac{0,1}{n}$$

donde c_t, c_s, m , r, c_{OM} e y son constantes. Por otro lado, n y P son las variables que están en juego en las configuraciones de la granja. La constante c_t es el costo de una turbina, desde la creación de la misma hasta la instalación. En nuestro modelo de costo, el valor de c_t es de 750000 dólares. La constante c_s es el costo de la subestación, el cual es de 8000000 dólares. El significado de m es la cantidad de turbinas que entran por subestación. r ($rate$) es la tasa de interés anual; para la competencia, r tiene un valor de 0,03. La constante y ($year$) hace referencia al tiempo de vida de la granja eólica; para el problema se estima que el tiempo de vida promedio de una granja es de 20 años. Por último, la constante c_{OM} es el costo de operación anual por turbina, que, en nuestro caso, es de 20000 dólares. En cuanto a las variables, n es la cantidad de turbinas propuestas al diseñar la configuración. P es la función que devuelve la energía total de la granja a partir de la posición de cada una de las turbinas y se define mediante la siguiente ecuación:

$$P(M) = e_c/(w_f * length(M)) \tag{1}$$

En la ecuación 1, M es una matriz que describe las posiciones de cada una de las turbinas de la configuración, e_c es la energía capturada por la configuración propuesta, w_f es la energía que se pierde en la granja (*wake free energy*) y $length(M)$ corresponde con la cantidad propuesta de turbinas.

3. Un estudio comparativo

En esta sección se mostrará cómo las propuestas desarrolladas se cotejan con otro algoritmo de búsqueda estocástico. Después de la explicación de todos los métodos, damos paso para que en la siguiente sección se muestre una comparación de los resultados experimentales. En un principio, exponemos las estructuras de los algoritmos genéticos binarios, luego el GWO binario desarrollado y por último, nuestro algoritmo que se ha adaptado al problema. Cabe destacar que en todos los algoritmos descriptos las soluciones son vectores binarios que indican si en una posición (x, y) determinada se debe o no se debe colocar una turbina.

3.1. Variante binaria de un Algoritmo Genético (GA)

GA es un algoritmo metaheurístico inspirado en la evolución biológica y su base es genético-molecular. La analogía del algoritmo se encuentra planteada en que las soluciones al problema, consideradas individuos, son sometidas a acciones aleatorias que se identifican como recombinaciones (cruce), mutaciones y selecciones. El proceso esencial de un GA se muestra en el Algoritmo 1.

Algoritmo 1 Pseudocódigo del GA

```
1: generar P(0); //iniciar una población con soluciones candidatas aleatorias
2: evaluar P(0); //Evaluar cada candidato con la función de fitness
3: para e = 1 a EVALMAX hacer
4:    P'(e) = seleccionar( P(e) ); //Seleccionar padres
5:    P"(e) = aplicar operadores( P'(e) ); //Cruzar padres y generar una nueva población
6:    P(e+1) = reemplazar( P(e), P"(e) ); //Mutar nuevos individuos
7:    evaluar P(e+1);
8:    e = e + 1;
9: fin para
```

3.2. Variante binaria del algoritmo Grey Wolf Optimizer (GWO)

El algoritmo GWO simula el comportamiento natural de una manada de lobos para cazar a su presa. Dentro de los lobos se distinguen distintas categorías como $alfa, beta, delta$ y $omega$ para la simulación. El mecanismo que sigue es sencillo. El primer grupo de lobos $alfa$ son los que lideran la manada y, por lo tanto, influyen de una manera más contundente en el espacio de búsqueda. El pseudocódigo es presentado en el Algoritmo 2. Al final de la ejecución, el mejor individuo va a ser

Algoritmo 2 Pseudocódigo del GWO

```
1: generar (X) // Inicializar la población de lobos grises
2: inicializarParametros(a, A, C);
3: evaluar X(0);
4: seleccionarnuevos(Alpha, Betha, Delta, X(0));
5: para e = 1 a EVALMAX hacer
6:    para todo Lobo l en Omega hacer
7:       para i = 0 a DIM hacer
8:          actualizarPosicion(l, i); //Actualizar la posición actual
9:       fin para
10:      ajustarParametros(a, A, C); //Ajusta los parametros del algoritmo
11:      evaluar X(e+1);
12:      seleccionarnuevos(Alpha, Betha, Delta, X(e+1));
13:      e = e + 1;
14:   fin para
15: fin para
```

el que esté asociado al lobo $alfa$. El parámetro a es el factor de exploración que comienza con un valor de 2 y va decrementándose a lo largo de las evaluaciones hasta llegar a 0.
Los parámetro A y C son vectores que tienen tres números aleatorios distintos y que ayudan a las soluciones candidatas, moviéndolas en el espacio de búsqueda.
La clave del algoritmo se encuentra en la función actualizarPosición(..) (ver línea 8 del Algoritmo 2) que se define como:

$$\vec{X}_{l,i} = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{2}$$

Donde cada $\vec{X}_1$, $\vec{X}_2$ y $\vec{X}_3$ se obtiene de la siguiente manera:

$$\vec{X}_1 = |\vec{x}_\alpha - \vec{A}_1 * \vec{D}_\alpha|, \qquad \vec{X}_2 = |\vec{x}_\beta - \vec{A}_2 * \vec{D}_\beta|, \qquad \vec{X}_3 = |\vec{x}_\delta - \vec{A}_3 * \vec{D}_\delta| \quad (3)$$

Por último para determinar $\vec{D}_\alpha$, $\vec{D}_\beta$ y $\vec{D}_\delta$, usados en (3) se aplican las siguientes fórmulas:

$$\vec{D}_\alpha = |\vec{C}_1 * \vec{X}_\alpha - \vec{X}|, \qquad \vec{D}_\beta = |\vec{C}_2 * \vec{X}_\beta - \vec{X}|, \qquad \vec{D}_\delta = |\vec{C}_3 * \vec{X}_\delta - \vec{X}| \quad (4)$$

En este trabajo, se ha convertido el algoritmo inicialmente planteado para números reales en valores booleanos. Para cumplir este propósito, se ha discretizado el resultado de la fórmula (2) siguiendo el criterio que si $\vec{X}_{l,i}$ es mayor que 1, la función devuelve 1, en otro caso, devuelve 0.

3.3. Un nuevo modelo: DonQuijote

En este apartado, introducimos la idea de nuestro algoritmo de optimización predilecto. Para esto vamos a dividir el algoritmo en tres etapas: inicialización, preprocesamiento y un paso iterativo.

Inicialización:
Se generan varias poblaciones que contengan cuatro soluciones aleatorias con (i * 30) - 1 turbinas, donde i comienza desde el máximo valor posible y es decrementada a medida que se van generando nuevas poblaciones.
El hecho de que la cantidad de turbinas vaya saltando de a 30 unidades se debe a que la función de fitness tiene en su numerador la función piso ($floor$), entre una división del número de turbinas y una constante 30 (número de turbinas por subestación); esto ha resultado en una ventaja muy buena para discretizar el espacio de búsqueda.
Siguiendo con el proceso, se selecciona la población que tiene mejor promedio de fitness y/o la que tiene mejor individuo.

Pre-procesamiento:
En esta etapa se agregan varios indiviuos a la población seleccionada anteriormente para evitar centralizar las soluciones en un solo sector del espacio de búsqueda. En primer lugar, se agrega un individuo que representa una disposición de turbinas en posiciones formando líneas, intercalados en la parrilla, perpendicular al valor más alto de la matriz de salida y en la misma dirección de la más pobre de la misma matriz. Al mismo tiempo, se añaden otros nuevos individuos con un número fijo de turbinas ubicadas aleatoriamente.

Paso iterativo:
La estructura principal de esta fase se expresa en el algoritmo 3. Donde el método primeraOperacion() selecciona una turbina para que sea movida a una posición vacía indicada por la peor recepción de energía de entrada.
El método segundaOperacion() genera, para las primeras evaluaciones nuevas distribuciones aleatorias sin cambiar el número de turbinas. Cuando el número de evaluaciones es lo suficientemente grande, se selecciona la turbina que tiene el peor

Algoritmo 3 Estructura general del paso iterativo

```
1: para e= 1 a MAXEVAL  hacer
2:    para todo individuo en la población hacer
3:       switch (e%4)
4:       case 0:
5:          primeraOperacion();
6:       case 1:
7:          segundaOperacion();
8:       case 2:
9:          terceraOperacion();
10:      case 3:
11:         cuartaOperacion();
12:      end switch
13:   fin para
14: fin para
```

promedio de recepción de energía, para que desde ésta se elija una turbina que se encuentra en la mejor dirección (de las 24 posibles), y sobre esta última se realice una reubicación en la peor dirección de la primera.
El método terceraOperacion() aplica DE binaria con una probabilidad de recombinación de 0,783 y una probabilidad de mutación de 0,06.

Evolución Diferencial binaria: La DE, al igual que los algoritmos genéticos, forma parte de la categoría de la computación evolutiva; aplica pasos similares (cruzar, mutar y seleccionar) pero sus comportamientos son distintos.
La DE fue planteada inicialmente para espacios continuos. Pero en nuestro caso las soluciones son vectores booleanos, que están asociado al número de generaciones o evaluaciones (e). El vector puede ser representado como:

$$X(i,e) = (x_{ie}(1), x_{ie}(2), \ldots, x_{ie}(D)),$$

donde i representa el índice del vector en la población, o también conocido como el número del individuo, D significa la cantidad posible de lugares en donde pueden ubicarse turbinas [10].
La operación de mutación, imita sobre un vector base r_0 otros vectores dependiendo de un número aleatorio y la distancia de Hamming de otras dos soluciones distintas a la primera. El resultado de la distancia es multiplicado por un factor de mutación (0,06). Esta constante controla la velocidad y robustez de la búsqueda. Para la etapa de recombinación (crossover) se mezclan la solución actual sin mutar con la solución mutada. Cuan similar será el vector resultante al vector mutado va a estar determinado por la probabilidad de recombinación (0,783).
Por último el operador de selección elige, de forma determinística, la mejor solución entre el producto obtenido en la etapa del cruce y el miembro actual de la población.

El método cuartaOperacion() genera, para las primeras evaluaciones, distribuciones aleatorias decrementando el número de turbinas por uno. Cuando el número de evaluaciones es lo suficientemente grande, se aplica la primera operación.

4. Resultados experimentales

Para mostrar el desempeños de los distintos algoritmos se ha optado por tomar 5 escenarios, extraidos de la competencia WFLO 2015 [11]. La representación de los escenarios es detallada en la Fig. 1, donde los rectángulos azules son los obstáculos, es decir, lugares donde no es posible ubicar turbinas.

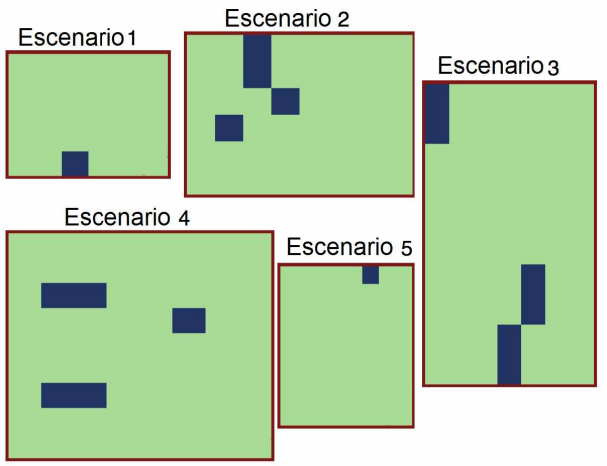


Figura 1. Tamaño de los escenarios y ubicación de los obstáculos

La Tabla 1 nos ayuda a comparar los escenarios entre sí dado que muestra el detalle de los parámetros utilizados en la prueba de cada uno de ellos.
La energía perdida a la que hace referencia la Tabla 1 es un factor que influye en la captura de energía de los aerogeneradores y que depende de las condiciones del terreno. Esta variable es definida en [8]. Mientras que, la cantidad máxima de turbinas es el número de turbinas que pueden ser ubicadas en el escenario formando hileras verticales y cumpliendo las restricciones de seguridad entre dos turbinas.

Característica	Escenario 1	Escenario 2	Escenario 3	Escenario 4	Escenario 5
Ancho	9240	6545	6930	10780	5390
Alto	6545	5005	12320	9240	6545
Energía perdida	6148.648	8674.542	12344.639	11314.82	7441.038
Cantidad máxima turbinas	607	362	843	967	390

Tabla 1. Parámetros de los escenarios usados en el estudio experimental

Para realizar las ejecuciones se hizo uso de una computadora con las siguientes características: Procesador Intel(R) Core(TM) i3-2330M CPU @2,20 GHz; Memoria física: 4044 MB; Sistema operativo Windows 7 Home Basic de 64 bits; las propuestas de este artículo fueron programadas utilizando el lenguaje JAVA.

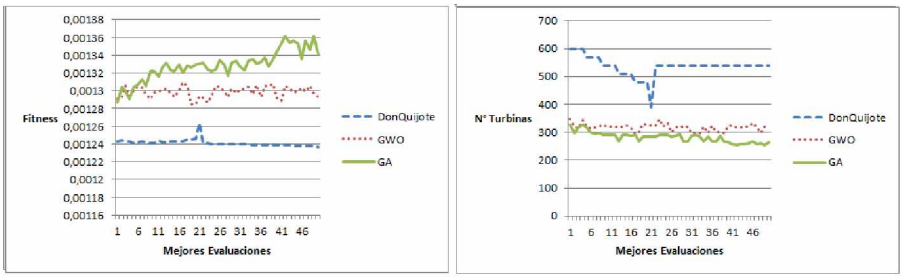
El resultado de las ejecuciones se observa en la Fig. 2. Por un lado, en las gráficas sobre la columna izquierda de la Fig. 2 se observan los resultados obtenidos por cada uno de los algoritmos en cada escenario considerando el costo de producir un kW. Por otro lado, para la misma Fig. 2, las gráficas sobre la columna de la derecha muestran la cantidad de turbinas utilizadas en cada configuración de los escenarios. De acuerdo con lo que se muestra en la Fig. 2, DonQuijote es capaz de proporcionar resultados muy competitivos y obtener, en todos los casos, el mejor valor final de fitness. Cabe destacar el comportamiento intuitivo de DonQuijote en la etapa de explotación en relación al número de turbinas (gráficos de la derecha en la Fig. 2); en esas mismas gráficas se puede observar la etapa de inicalización de DonQuijolte. El trazado inicial en forma escalonada de la gráfica se produce debido a que el algoritmo prueba, durante la inicialización, diferentes configuraciones con una cantidad fija y múltiplo de (i*30) -1 de turbinas, como se dijo previamente en la sección 3.3. Por lo tanto, estos resultados muestran que elegir el número de turbinas de esta forma provoca un rendimiento superior del algoritmo. La mejoría en el desempeño se debe a que el algoritmo explota más rápidamente un sector del espacio de búsqueda. Esta cualidad puede ser provechosa en el proceso de diseño de parques eólicos que se modelen con la función de fitness tratada aquí. En la siguiente sección vamos a discutir los beneficios y desventajas de este enfoque en el proceso de diseño de granjas eólicos.
Con respecto a los tiempos de ejecución para cada uno de los tres enfoques sobre los 5 escenarios son los siguientes:

- GA: 169 minutos con 45 segundos.
- GWO Binario: 173 minutos con 29 segundos.
- DonQuijote: 932 minutos 21 segundos.

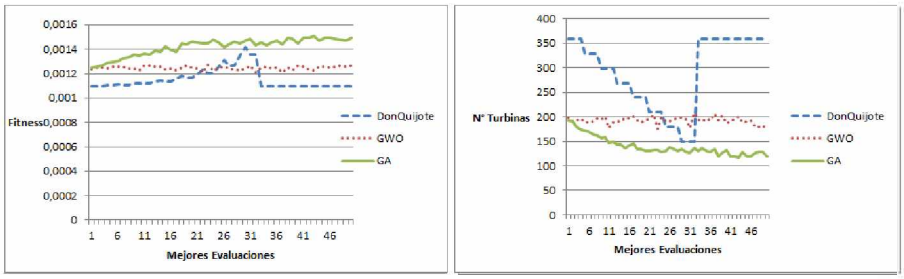
Se debe enfatizar que los tiempos no son un factor de gran énfasis en la competencia puesto que las ejecuciones son locales con respecto a cada competidor. Pero, a pesar de esto no deja de ser elemento a tener en cuenta para las posibles mejoras de DonQuijote.

5. Conclusiones y trabajos futuros

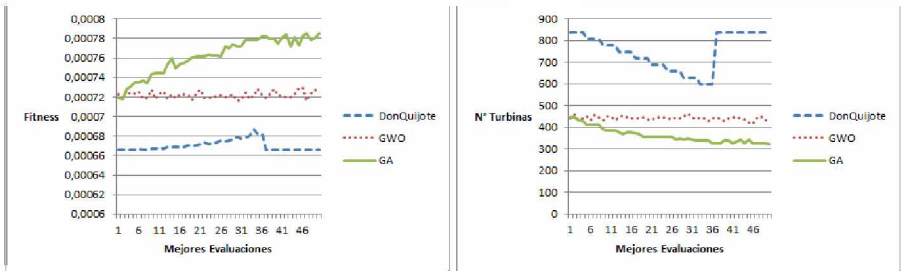
En este trabajo hemos presentado dos nuevos algoritmos para resolver el problema de configuraciones óptimas de granjas eólicas (WFLO). El mejor algoritmo combina una metaheurística con un fuerte análisis del problema, que cooperando juntos y en simultáneo son usados para optimizar el costo de producción de un kilowatt. La etapa de inicialización es utilizada para acotar en gran medida el espacio de búsqueda, la etapa de preprocesamiento sirve para explotar el sector del espacio de búsqueda seleccionado y el algoritmo evolutivo junto con otras operaciones propias del problema son utilizados para mejorar las soluciones actuales.
Los resultados preliminares muestran que el algoritmo propuesto (DonQuijote) supera al *baseline* de la competencia mediante la producción de la mejora de los resultados para el conjunto de casos probados, sin embargo posteriormente se podrían mejorar los tiempos de ejecución del mismo. Por otro lado, se ha corroborado experimentalmente los buenos resultados que arrojan los algoritmos basados en aprendizaje evolutivo con respecto a este problema.



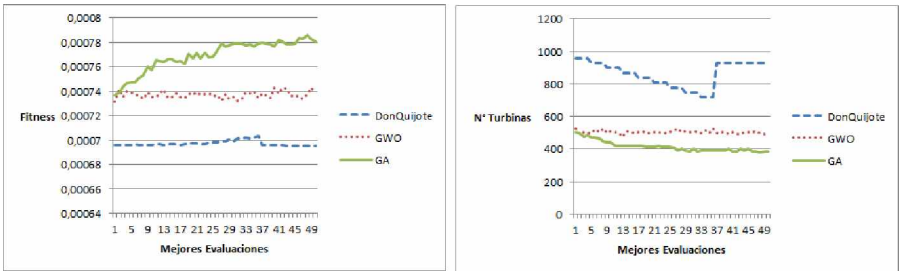
Escenario 1



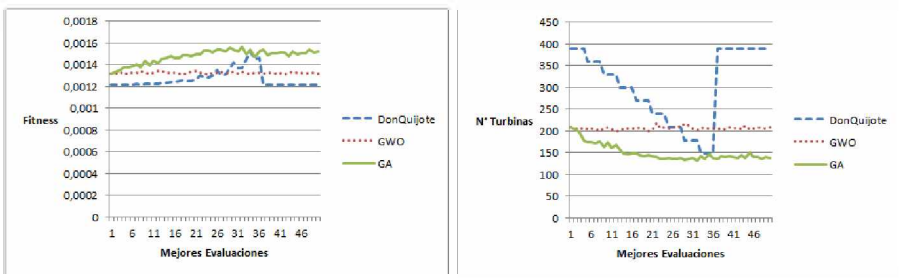
Escenario 2



Escenario 3



Escenario 4



Escenario 5

Figura 2. Comparación entre el GA y nuestros dos modelos desarrollados

A futuro se piensa probar de ubicar las turbinas en cualquier lugar, eliminando la grilla estática y guardando solo la condición de que no entren en el perímetro de seguridad de otra turbina. Otro trabajo es anticiparse a los cambios que se presenten en la competencia del año 2016 al considerar el terreno en un espacio tridimensional.

Reconocimientos

Los autores de este artículo agradecen al LIDIC (Laboratorio de Investigación y Desarrollo en Inteligencia Computacional) y al financiamiento de la Universidad Nacional de San Luis mediante el PROICO 330214.

Referencias

1. R. Wiser y M. Bolinger Annual report on U.S. wind power installation, cost, and performance trends: 2006. Available from. Golden, CO: NREL, US Department of Energy, `http://www.nrel.gov/wind/pdfs/41435.pdf`, (2007).
2. K.S. Tang, K.F. Man, S. Kwong, Q. He, Genetic algorithms and their applications, IEEE Signal Processing Magazine vol. 13 (6): 22—37, (1996).
3. M. Wagner, K. Veeramachaneni, K. Neumann, y U. O'Reilly. Optimizing the Layout of 1000 Wind Turbines.European Wind Energy Association Annual Event: 1–10, (2011).
4. M. Wagner , J. Day y F. Neumann. A Fast and Effective Local Search Algorithm for Optimizing the Placement of Wind Turbines, Renewable Energy vol. 51: 64–70, (2013)
5. D. Wilson, E. Awa, S. Cussat-Blanc, K. Veeramachaneni, y U.-M. O'Reilly. On learning to generate wind farm layouts. In Proceeding of the fteenth annual conference on Genetic and evolutionary computation conference, (2013).
6. D. Wilson , S. Cussat-Blanc, K. Veeramachaneni, , U. O'Reilly, y H. Luga. A Continuous Developmental Model for Wind Farm Layout Optimization. Proceedings of the GECCO. 745–752, (2014)
7. Veeramachaneni, K.; Wagner, M.; O'Reilly, U.-M.; Neumann, F., Optimizing energy output and layout costs for large wind farms using particle swarm optimization, Evolutionary Computation (CEC), IEEE Congress on, 1–7, (2012)
8. A. Kusiak y Z. Song. Design of wind farm layout for maximum wind energy capture. Renewable Energy, vol. 35(3):685–694, (2010).
9. S. Mirjalili, S. M. Mirjalili, A. Lewis. Grey Wolf Optimizer, accepted in Advances in Engineering Software , vol. 69: 46–61, (2013)
10. T. Gong y A. L. Tuson. Differential Evolution for Binary Encoding, Soft Computing in Industrial Applications, ASC 39, 251–262, (2009)
11. 2nd Edition of the Wind Farm Layout Optimization Competition. GECCO. `http://www.irit.fr/wind-competition/`, fuente de los escenarios: `https://github.com/d9w/WindFLO/tree/master/Wind%20Competition/2015/Scenarios`, (2015)