Análisis Estadístico de Algoritmos Evolutivos para el problema de Selección de Variables

(Statistical Analysis of Evolutionary Algorithms for the Variable Selection problem)

Victor Adrian Jimenez,[i] Diego Fernando Lizondo[ii] y Adrián Will[iii]

Resumen

Durante décadas de investigación en problemas de optimización, se han desarrollado innumerables algoritmos, tanto determinísticos como heurísticos. Aun así, debido a este amplio abanico de posibilidades resulta una tarea compleja determinar cuál de ellos es más adecuado para un problema específico. Se propone en este trabajo una comparativa entre diferentes algoritmos heurísticos de optimización haciendo uso de test estadísticos. Se consideró Simulated Annealing (SA), Algoritmo Genético Simple (sGA), Algoritmos Genético Compacto (cGA) y Deterministic Crowding (DC) aplicados al problema de Selección de Variables para Estimación utilizando Regresión Lineal. Se utilizaron tres casos de prueba, que consisten en la estimación de Radiación Solar, Consumo Eléctrico en la provincia de Tucumán (Argentina), y el tiempo de reaparición de células cancerígenas. Se comprobó que existe suficiente evidencia estadística para afirmar que los algoritmos arrojan resultados significativamente diferentes. Además se concluye que sGA y DC son los algoritmos más apropiados para estos problemas, permitiendo obtener valores de fitness similares siendo sGA ligeramente superior.

Palabras claves: computación evolutiva, selección de variables, algoritmo genético, regresión lineal, análisis estadístico

[i] Grupo de Investigación en Tecnologías Informáticas Avanzadas, Universidad Tecnológica Nacional - Facultad Regional Tucumán. adrian.jimenez@gitia.org

[ii] Grupo de Investigación en Tecnologías Informáticas Avanzadas, Universidad Tecnológica Nacional - Facultad Regional Tucumán. diego.lizondo@gitia.org

[iii] Grupo de Investigación en Tecnologías Informáticas Avanzadas, Universidad Tecnológica Nacional - Facultad Regional Tucumán. adrian.will@gitia.org

Abstract

Decades of research in optimization problems have generated a considerable number of algorithms, both deterministic and heuristic. However, due to this wide range of possibilities, determine which one is the most appropriate for a specific problem is a complex task. In this paper, a comparison among different heuristic optimization algorithms using statistical test is proposed. Simulated Annealing (SA), Simple Genetic Algorithm (sGA), Genetic Algorithms Compact (cGA) and Deterministic Crowding (DC) were used, applied to the Variable Selection for estimation problem using Linear Regression. Three test cases were used: solar radiation at the province of Tucuman (Argentina), power consumption estimation in the same area, and estimation of the reappearance of cancer cells. We concluded that there is sufficient statistical evidence to affirm that the algorithms yield significantly different results. Also, we concluded that sGA and DC were the most suitable algorithms, obtaining similar fitness values, being sGA slightly better.

Keywords: evolutionary computation, variable selection, genetic algorithm, linear regression, statistical analysis.

1. Introducción

Desde hace muchas décadas los problemas de optimización han sido causales de la necesidad de desarrollar métodos y técnicas cada vez más eficientes para la resolución de los mismos. Esto ha provocado el surgimiento y desarrollo de un gran número de algoritmos de optimización determinísticos y heurísticos. Algunos de ellos se encuentran basados en modelos matemáticos como "Constraint Programing", "Dynamic Programing" o "IDA" (Denardo, 2012; Peng, Bessho, Koshizuka, & Sakamura, 2014) y otros inspirados en la naturaleza y en la biología como Particle Swarm Optimization, Ant Colony Optimization, Genetic Algorithms, Artificial Immune Systems (Parsopoulos, 2010; Williams & Daneshyari, 2016; Yu & Gen, 2010). Asimismo, una amplia variedad de adaptaciones de los mismos se han desarrollado con el objetivo de conseguir cada vez mejores soluciones. Además, existen algoritmos híbridos desarrollados para explotar características particulares para obtener mejores resultados en problemas específicos (Talbi, 2009). Esto ha llevado a la aparición de una gran cantidad de métodos capaces de aplicarse a un mismo tipo de problema. Sin embargo, la abundancia de oferta de algoritmos que pueden resolver un mismo problema, llevó a la necesidad de responder a preguntas del tipo: "¿Cuál algoritmo debe ser aplicado en una situación particular?", "¿Cuál de ellos muestra un mejor desempeño?", "¿Cuáles no pueden ser considerados adecuados, aunque muestren aparentemente buenos resultados?". El procedimiento para determinar cuál heurística es la más adecuada, independientemente de los resultados y los tiempos de ejecución obtenidos en una corrida en particular ha resultado ser un tema de investigación de crucial importancia. En este sentido, los test-estadísticos fueron utilizados con éxito como método de validación en la comparación de algoritmos heurísticos para un conjunto de instancias de un mismo problema (Derrac, García, Molina, & Herrera, 2011).

En éste trabajo se propone aplicar cuatro heurísticas a tres instancias del problema de Selección de Variables para estimación de variables (Will, Bustos, Bocco, Gotay, & Lamelas, 2013), y evaluar su desempeño utilizando test estadísticos. Se ha optado por un algoritmo de trayectoria Simulated Annealing (SA), y tres poblacionales, Algoritmo Genético Compacto (cGA) (Ahn & Ramakrishna, 2003; Harik, Lobo, & Goldberg, 1999), Algoritmo Genético simple (sGA) (Yu & Gen, 2010) y Deterministic Crowding (DC) (Mahfoud, 1995). Los algoritmos implementados fueron puestos a prueba con tres conjuntos de datos, relacionados a la estimación de radiación solar en la provincia de Tucumán, estimación del consumo de energía a nivel provincia y la estimación del tiempo de reaparición de células cancerígenas. Para ello se cuenta con datos de clima provistos por la Estación Experimental Agroindustrial Obispo Colombres (E.E.A.O.C.), datos del consumo eléctrico a nivel provincia proporcionados por la Empresa de Distribución

Eléctrica de Tucumán S.A. (E.D.E.T.s.a.) y una base de datos de Cáncer extraída de UCI Machine Learning Repository.[iv]

2. Metodología y Datos Empleados

Para el desarrollo de este trabajo se consideran modelos lineales para las estimaciones de las variables de salida. Se utiliza Regresión Lineal debido a la fácil interpretación que puede hacerse del modelo que se desea conseguir. Sin embargo, debido a que las variables de entrada pueden presentar una correlación lineal no trivial entre ellas, es posible la aparición de sistemas mal condicionados (producen una fuerte variación en la salida ante pequeños cambios en la entrada). Por este motivo se decidió utilizar la pseudoinversa de Moore-Penrose (Eldén, 2007), ya que es capaz de obtener una buena solución incluso en presencia de sistemas mal condicionados.

2.1 Descripción de los datos

Para el desarrollo de este trabajo se utilizaron tres conjuntos de datos, cada uno de los cuales cuenta con variables de distinta naturaleza. Por un lado, los datos climáticos fueron recolectados de cinco diferentes estaciones meteorológicas ubicadas en distintos puntos geográficos de la provincia de Tucumán, Argentina (Figure 1). Los mismos están comprendidos en el periodo entre los años 2005 y 2012. Las variables disponibles en cada estación son:

- Temperatura Máxima diaria [ºC]
- Temperatura Mínima diaria [ºC]
- Presión atmosférica promedio diaria [hPa]
- Humedad Relativa Media promedio diaria [%]
- Radiación Solar Observada [W/m^2]

Los datos de la demanda eléctrica, por otro lado, corresponden a mediciones diarias de la demanda de consumo eléctrico en la provincia, en el periodo comprendido entre los años 2005 y 2012. Por último, el tercer conjunto de datos (William H. Wolberg, 1992) contiene 198 muestras descritas por 30 variables creadas a partir de una imagen digitalizada de una muestra de tejido. Dichas variables describen características de las células presentes en la imagen. Los datos incluyen dos variables adicionales: el diámetro del tumor y el número de ganglios linfáticos de axila positivos observados en el momento de tomar la muestra. Los datos fueron pre-procesados para normalizar sus valores entre 0 y 1.

iv UCI: https://archive.ics.uci.edu

A partir de estas bases de datos se generaron tres instancias del problema de selección de variables. En el primero se estima la radiación solar observada en una de las estaciones meteorológicas (ubicada en El Colmenar) a partir de las variables climáticas restantes de la propia estación y todas las variables disponibles en las otras cuatro estaciones. En el segundo caso se considera el mismo conjunto de datos climáticos del caso anterior pero esta vez para estimar el consumo eléctrico en la provincia de Tucumán. Además de datos climáticos se utiliza información adicional sobre días feriados, debido a la influencia que tiene esta información sobre la demanda de energía. Las variables adicionales utilizadas se listan a continuación:

- Variables climáticas (temperaturas mínima y máxima, presión promedio, humedad relativa media y radiación solar observada), de las cinco estaciones meteorológicas (25 variables).
- Temperaturas mínima y máxima al cuadrado para cada una de las estaciones meteorológicas (10 variables adicionales).
- Días feriados, codificados en una variable binaria indicando con un "1" los feriados y "0" para el resto de los días (1 variable adicional).
- Variables temporales: año, doy (day-of-year), mes, día del mes y día de la semana (5 variables adicionales).
- Demanda eléctrica en la provincia [MWh].

El tercer y último conjunto de datos es utilizado para predecir el tiempo que tarda un tumor cancerígeno en reaparecer con los datos obtenidos de la base de datos pública mencionada anteriormente.

En los dos primeros casos, al tratarse de series de tiempo resulta factible aplicar diferentes técnicas y metodologías de series de tiempo en pos de mejorar las estimaciones. Una de estas técnicas consiste en incluir información histórica como variables de entrada, llamadas *lag variables* (Jimenez, Lizondo, Will, & Rodriguez, 2017). Entonces, además de las variables convencionales agregamos otras correspondientes a datos de días previos, considerando sólo dos días hacia atrás, ya que según pruebas preliminares aumentar la cantidad de días no mejora significativamente los resultados. Por otro lado, en el caso de estimación de tiempo de reaparición de Cáncer al no tratarse de una serie temporal, no fue utilizada esta técnica.

Figura 1. Ubicación geográfica de las estaciones meteorológicas de la provincia de Tucumán, Argentina.

2.2 Selección de Variables

La presencia de una gran cantidad de variables disponibles para la generación de modelos para estimación incrementa la complejidad de un modelo lineal, lo que aumenta la probabilidad de que se produzca un sobre-entrenamiento (overfitting) (Varmuza & Filzmoser, 2008). Además, la presencia de variables con ruido o muestras erróneas incrementan el error cometido en la estimación de los valores de la variable de dicho realmente necesarias o de influencia significativa en el modelo. Por estas razones es necesario reducir el conjunto de variables de entrada, seleccionando aquellas que permitan obtener la mejor estimación de las variables deseadas. Este procedimiento es conocido como Selección de Variables o Feature Selection (Kotu & Deshpande, 2015).

En cuanto a la codificación para este problema se eligió la codificación binaria por subconjuntos entre otras codificaciones posibles como permutación y corte, por ser la más adecuada (Will et al., 2013). Ésta consiste en representar una solución del problema mediante un vector binario, donde cada componente indica si la variable es considerada en el modelo (con valor 1) o si es descartada (valor 0).

A continuación se detallan los algoritmos utilizados en el desarrollo de este trabajo.

2.2.1 Simulated Annealing

Simulated Annealing (Kirkpatrick, 1984) es una heurística de búsqueda global inspirada en el recocido en la metalurgia. Es una técnica que consiste en

el calentamiento y posterior enfriamiento controlado de un material (en general aplicado en metales) para obtener estructuras cristalinas de baja energía. En cada iteración del algoritmo se efectúa una perturbación en la solución. En nuestro caso, al tratarse de un problema binario, esta perturbación se realiza cambiando los valores de coordenadas del vector binario seleccionadas al azar con una probabilidad p. Luego, basado en un valor de temperatura, se decide si la solución resultante se conserva o no para la próxima iteración. La aceptación de malas soluciones con una cierta probabilidad es una propiedad fundamental de este tipo de heurísticas, que proporciona un mecanismo para escapar de las zonas de óptimos locales. En la Figura 2 se muestra el pseudo-código del algoritmo Simulated Annealing.

```
Parámetros: T0: temperatura inicial, μ: parámetro de enfriamiento, p:
  probabilidad para perturbación de soluciones.
t = 0
mejorX = generateRandomSolution()
mejorF = evaluar(mejorX)
while comprobarCondicionTerminacion() == false do
    /* obtener nueva solución                                    */
    x = perturbar(mejorX, p)

    /* determino si conservo la nueva solución                   */
    if f(x) > mejorF then
        mejorX = x
        mejorF = f(x)
    else
        if random() ≤ Temperatura(t, T0, μ) then
            mejorX = x
            mejorF = f(x)

    /* incremento contador de iteraciones                        */
    t = t + 1
```

Figura 2. Pseudocódigo para Simulated Annealing.

El esquema de enfriamiento utilizado sigue un decremento exponencial en el tiempo, según la Ecuación 1, donde T_0 es la temperatura inicial y μ es un parámetro que permite controlar su descenso.

$$T(t)=T_0 \exp(-\mu t) \quad (1)$$

2.2.2 Algoritmo Genético Simple

Los AG basan su funcionamiento en la teoría evolutiva, imitando en mayor o menor medida los principios biológicos de la misma. Se plantea que los individuos son diferentes soluciones a un problema determinado, y la capacidad de adaptación de los mismos se ponderará mediante una función de evaluación definida por el problema (Yu & Gen, 2010). El Algoritmo Genético Simple (sGA) tiene una

arquitetura tradicional, en la cual inicia su ejecución creando una población de individuos o conjunto de soluciones generados al azar. Luego, a la población se le aplican ciertos operadores genéticos (basados en la biología en la que están inspirados los AG), con la finalidad de combinar ciertas características de las soluciones para obtener otras mejores. Este procedimiento permite evolucionar la población de individuos a través de las generaciones hasta encontrar mejores soluciones que cada vez estarán más cerca del óptimo esperado.

El pseudo-código del sGA se muestra en la Figura 3. El método de selección utilizado es Stochastic Universal Sampling (SUS) (Yu & Gen, 2010) porque posee alta eficiencia y simplicidad. La recombinación, por otro lado, se realiza mediante el operador Uniform Crossover que intercambia cada bit de las cadenas de los padres con una probabilidad de p_{UC}= 0.5. Debido a que este operador puede intercambiar cualquier coordenada de las soluciones padres, permite una mejor exploración del espacio de búsqueda. En cuanto a la mutación, se utiliza una versión probabilística del operador Binary Mutation, donde las coordenadas son cambiadas de valor según un umbral de probabilidad fijada previamente.

```
Parámetros: popSize: cantidad de individuos en la población, selSize:
  cantidad de individuos obtenidos de selección, mutSize: cantidad de
  individuos mutados
/* generar población inicial con individuos generados al
   azar                                                    */
pop = generateRandomPopulation(popSize)

while checkTerminationCondition() = false do
    /* obtener y guardar mejor individuo de la población  */
    elite = getBestIndividual(pop)

    /* seleccionar individuos para cruzamiento            */
    selPop = selection(pop, selSize)

    /* completar población seleccionada con individuos
       elegidos al azar                                   */
    pop = shufflePopulation(pop)
    for k = 1 to popSize - selSize do
        selPop = selPop ∪ pop[k]

    /* aplicar cruzamiento                                */
    newPop = {}
    selPop = shufflePopulation(selPop)
    for k = 1 to selSize-1 step 2 do
        newPop = newPop ∪ crossover(selPop[k], selPop[k+1])

    /* aplicar mutación                                   */
    newPop = shufflePopulation(newPop)
    for k = 1 to mutSize do
        newPop[k] = mutation(newPop[k], prob_mut)

    /* reinsertar elite en un lugar al azar dentro de la
       población                                          */
    k = randomInteger(1, cant_sel)
    newPop[k] = elite

    /* evaluar fitness de individuos                      */
    pop = newPop; for k = 1 to popSize do
        evaluate(pop[k])

bestSolution = getBestIndividual(pop)
```

Figura 3. Pseudocódigo para Algoritmo Genético Simple.

2.2.3 Algoritmo Genético Compacto

Los Algoritmos de Estimación de Distribución (Estimation of Distribution Algorithm o EDA) son un tipo de AG que utilizan modelos estadísticos para generar nuevos individuos en lugar de los operadores de mutación y cruzamiento de los AGs convencionales (Hauschild & Pelikan, 2011). El Algoritmo Genético Compacto (compact Genetic Algorithm o cGA) (Harik et al., 1999) es un tipo de algoritmo EDA que representa la población como un vector de probabilidad *(PV)* sobre el conjunto de soluciones. De este modo, operacionalmente imita el comportamiento de un sGA con cruzamiento uniforme.

Ahn et al. (Ahn & Ramakrishna, 2003) propusieron una mejora al cGA, presentando un esquema elitista persistente: persistent elitist compact GA (pe-cGA). Este algoritmo proporciona un medio para reducir la deriva genética, mediante el mantenimiento de una copia del mejor individuo para convertirse en parte de la nueva población en la siguiente generación. De esta manera se asegura que el mejor individuo pueda pasar sus rasgos a la siguiente generación. Este esquema también introduce una simulación de una selección por torneos para aumentar la presión de selección. Esto consiste en generar *s* (tamaño del torneo) individuos a partir del vector de probabilidad y luego dejar que el mejor de ellos compita con los otros *s*-1 individuos.

```
Parámetros: n: tamaño de la población, l: tamaño del individuo, s: tamaño del
  torneo
/* Inicializar el vector de probabilidad y el individuo
   elite:                                                    */
for i = 1 to l do
    PV[i] = 0.5;
E = generate(PV)

while checkTerminationCondition() == false do

    /* Generar individuos a partir del vector de
       probabilidad                                          */
    S[1] = E
    for i = 2 to s do
        S[i] = generate(PV)

    /* Ordenar S de manera que S[1] sea el individuo con
       mayor fitness                                         */
    S = sortPopulation(S)

    /* Reemplazar el individuo elite                         */
    E = S[1]

    /* Hacerlos competir y actualizar el vector de
       probabilidad                                          */
    for j = 2 to s do
        winner, loser = compete(E, S[j])
        for i = 1 to l do
            if winner[i] ≠ loser[i] then
                if winner[i] = 1 then
                    PV[i] = PV[i] + 1/n
                else
                    PV[i] = PV[i] - 1/n

bestSolution = generate(PV)
```

Figura 4. Pseudocódigo para persistent elitist compact GA.

2.2.4 Deterministic Crowding

Dentro de las variantes de Algoritmos Genéticos tipo Niching, Deterministic Crowding (DC) es una de las más difundidas. Estos algoritmos permiten encontrar y conservar todos los posibles "Nichos Ecológicos", es decir todos los posibles puntos o zonas del espacio de búsqueda alrededor de los óptimos locales. DC fue introducido por De Jong (De Jong, 1975) como una técnica para preservar la diversidad de la población y prevenir la convergencia prematura. El mismo consta de dos fases principales. En la primera fase, de *cruzamiento*, los individuos de la población se combinan de a pares y generan individuos hijos. En segunda fase, de *sustitución*, se emparejan los padres con los hijos de acuerdo a una métrica de similitud, para luego tomar una decisión sobre cuál de ellos sobrevive en la población utilizando la función de fitness *f* (descrita en la siguiente Sección). Esta decisión se toma de acuerdo al algoritmo detallado en la Figura 5. Adicionalmente, en la implementación utilizada se muta un porcentaje de los hijos generados en la fase de cruzamiento, utilizando el mismo operador que en el caso de sGA. Se elige DC entre las variantes Niching más populares dado que presenta buenos resultados en la búsqueda de óptimos locales, es estable respecto a repeticiones y fue utilizado con éxito en otros trabajos relacionados con selección de variables (Will et al., 2013).

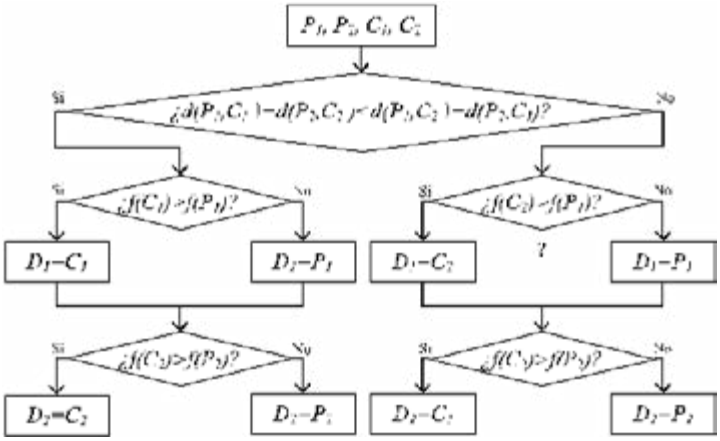


Figura 5. Diagrama de flujo para decidir cuales individuos entre los padres (P_1 y P_2) e hijos (C_1 y C_2) pasarán a la siguiente generación (D_1 y D_2) en Deterministic Crowding.

2.3 Definición de función objetivo

El uso de algoritmos heurísticos requiere la definición de una función objetivo. Para el caso de selección de variables, ésta función objetivo debe dar una medida del

rendimiento del modelo creado a partir de las variables que indique cada solución. Una forma de medir el rendimiento del modelo es utilizar el error de estimación calculado sobre el conjunto de datos experimentales. En nuestro caso, las métricas de error utilizadas son: Root Mean Square Error (RMSE) y el Coeficiente de Correlación Lineal (R). Las buenas soluciones serán aquellas que produzcan un RMSE bajo y un valor de R cercano a 1. Sin embargo, si sólo se consideran estos errores el algoritmo genético encontrará soluciones con el menor error posible, ignorando el hecho de que entre las variables seleccionadas pueda haber información redundante. Se adopta entonces la función objetivo descripta en la Ecuación 2.

$$(Sol) = -\frac{RMSE}{R}[1 + \alpha\, Pen(Sol)] \quad (2)$$

La función así definida provoca que el algoritmo tienda a elegir soluciones con pocas variables. Esto sucede debido a que se introdujo el factor de penalización Pen(Sol) definido por la Ecuación 3. Éste se calcula como la cantidad de variables utilizadas sobre la cantidad total de variables disponibles *n*. De este modo, mientras más variables elija la solución, peor será el fitness que se le asigne.

$$Pen(Sol) = \frac{\sum_{i=1}^{n} Sol(i)}{n} \quad (3)$$

El efecto de la penalización propuesta se controla mediante el parámetro α. Para valores bajos de este parámetro la penalización es suave, admitiendo soluciones con muchas variables que posiblemente producirán un error muy bajo. Por el contrario, para valores grandes el algoritmo encontrará soluciones con pocas variables, pero con un error de estimación más grande. Un valor adecuado para el parámetro α es aquel que permite un compromiso equilibrado entre cantidad de variables y error de estimación obtenido por la solución.

2.4 Test Estadísticos

Como se ha mencionado anteriormente, resulta necesario emplear un método que sea capaz de juzgar o realizar una comparativa clara entre el desempeño de dos o más algoritmos, teniendo en cuenta no únicamente los resultados particulares obtenidos en una única ejecución. Uno de los métodos disponibles más utilizados para hacer esto son los test estadísticos, que ayuda a tomar la decisión sobre cuál de los métodos utilizados es mejor. El test de Kruskal-Wallis (no-paramétrico) es uno de ellos, y fue desarrollado para comparar muestras mutuamente independientes de dos o más grupos de observaciones (Coffin & Saltzman, 2000). Esto lo realiza

comparando las medianas de las muestras, arrojando un valor p para la hipótesis nula *H0,* la cual establece que todas las muestras provienen de distribución con igual mediana. Si el valor *p* es cercano a cero, se pone en duda la hipótesis nula y sugiere que al menos una de las medianas es significativamente diferente a las demás. En nuestro caso consideramos que las medianas son significativamente diferentes si el valor *p* es menor a *0.01* (valor utilizado usualmente en este tipo de test).

2.5 Ajuste de parámetros

Para ajustar los parámetros de los algoritmos se probaron diferentes combinaciones de valores para determinar una configuración adecuada. La combinación de valores para cada algoritmo que tuvieron mejor performance en las estimaciones se muestra en la Tabla 1.

Parámetro	Radiación Solar	Consumo Eléctrico	Reap. De Cáncer
Simulated Annealing			
Temperatura Inicial *(T_0)*	2.00	2.00	2.00
Cte. de enfriamiento *(μ)*	0.03	0.05	0.05
Prob. de generación de Vecindad *(p)*	0.10	0.20	0.10
Alg. Genético Simple			
Población (popSize)	30	20	20
Seleccionados (selSize)[%]	80	80	80
Mutaciones (mutSize) [%]	10	10	10
Alg. Genético Compacto			
Tamaño de Población *(n)*	120	80	120
Tamaño del torneo *(s)*	5	3	6
Deterministic Crowding			
Tamaño de Población *(n)*	40	50	40
Mutaciones (mutSize) [%]	15	20	10

Tabla 1. Configuración de parámetros empleada para cada algoritmo en todos los casos.

Debido a que el objetivo de este trabajo es realizar una comparativa de métodos para selección de variables, la función fitness utilizada en cada caso debe estar definida de igual forma para todos los algoritmos. Por este motivo se adopta un valor de factor de penalización α=0.3 en todos los casos (ver Ecuación 3). Además, para poder realizar la comparativa de algoritmos de manera correcta es necesario que cada uno de ellos considere la misma condición de terminación para el mismo

problema. Por este motivo, en todos los casos se optó por terminar la ejecución del algoritmo si luego de un número de iteraciones/generaciones no se encontraba una mejor solución que la mejor actual. Este criterio es un buen indicativo de que el algoritmo ya encontró una buena solución (un óptimo local o global) e independiza el criterio de parada del problema en sí. En nuestro caso consideramos que para que el algoritmo finalice es necesario que ocurran 100 iteraciones sin cambios en la mejor solución encontrada.

Por último, para que los resultados sean estadísticamente confiables, cada algoritmo fue ejecutado 30 veces utilizando diferentes semillas del generador aleatorio. Luego, para cada problema se aplica el test de Kruskal-Wallis para determinar si existe diferencia significativa en los valores de fitness obtenidos.

Conjunto de datos	Algoritmo	Fitness Máx.	Fitness Mín.	Fitness - Media	Fitness - Mediana	Fitness - Desv. Est.	Valor-p
Radiación Solar	SA	-21.25	-23.28	-21.85	-21.76	0.43	2.2-16
	sGA	-20.81	-20.94	-20.83	-20.82	0.03	
	pe-cGA	-20.90	-21.46	-21.21	-21.22	0.13	
	DC	-20.81	-20.97	-20.87	-20.88	0.04	
Consumo Eléctrico	SA	-348.69	-529.23	-368.43	-356.54	43.23	2.2-16
	sGA	-335.93	-341.64	-338.26	-337.88	1.48	
	pe-cGA	-346.00	-356.09	-350.25	-349.80	2.64	
	DC	-335.85	-339.55	-337.91	-337.81	1.02	
Reaparición de Cáncer	SA	-13.27	-16.88	-14.04	-13.85	0.67	2.2-16
	sGA	-13.12	-13.42	-13.13	-13.12	0.06	
	pe-cGA	-13.12	-14.45	-13.59	-13.54	0.31	
	DC	-13.12	-13.45	-13.25	-13.23	0.09	

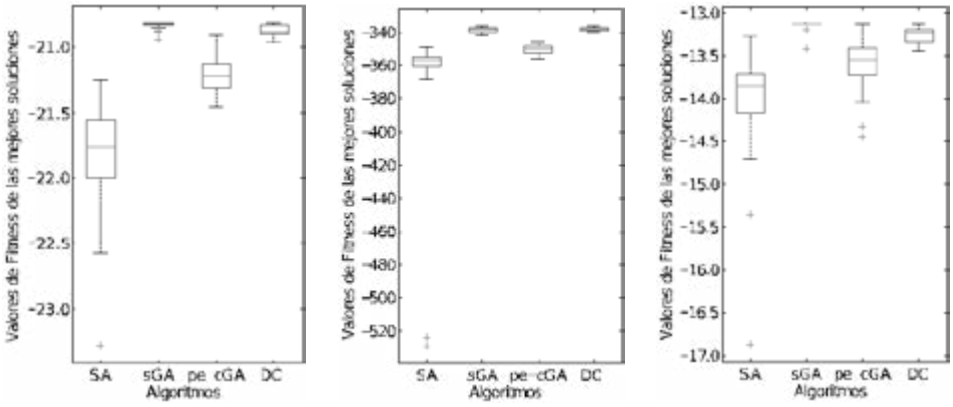
Tabla 2. Resumen estadísticos de pruebas Kruskal-Wallis para los distintos casos.

3. Resultados

En primer lugar, en la Tabla 2 se presenta un resumen estadístico de las diferentes corridas de los algoritmos. En ella se reporta para cada problema el valor mínimo, valor máximo, media, mediana y desviación estándar de los valores de fitness obtenidos al final de la corrida de cada problema. Por otro lado, en la Figura 6 se presenta un gráfico de cajas que permite observar fácilmente cuál algoritmo tuvo mejor desempeño comparando las medianas de los valores de fitness obtenidos en las repeticiones de cada corrida. Además, este grafico permite ver que tan estable

a repeticiones es cada algoritmo viendo la desviación obtenida en cada caso de prueba. Por último, en la Tabla 3 se muestra un resumen los resultados de una prueba de comparación múltiple, en el cual se comparan los algoritmos de a pares para determinar si sus resultados son similares entre sí.

Podemos ver que en todos los casos la media del valor de fitness es mucho peor (valor más pequeño) para Simulated Annealing. Tanto para el Algoritmo Genético Simple, como para Deterministic Crowding el resultado es muy similar, siendo ligeramente superior para el primero en casi todos los casos. La gran diferencia existente entre Simulated Annealing y el resto de los algoritmos implementados se ve reflejada en el *valor-p* resultante del test de Kruskal-Wallis. Al ser el valor-p muy pequeño se rechaza la hipótesis nula *(H0)* y por lo tanto se puede afirmar estadísticamente que existen diferencias significativas entre los algoritmos. Esto se evidencia en los gráficos de cajas de la Figura 6. En la prueba de comparación múltiple, detallada en la Tabla 3, se puede ver que los únicos algoritmos que producen resultados estadísticamente similares son el Algoritmo Genético Simple

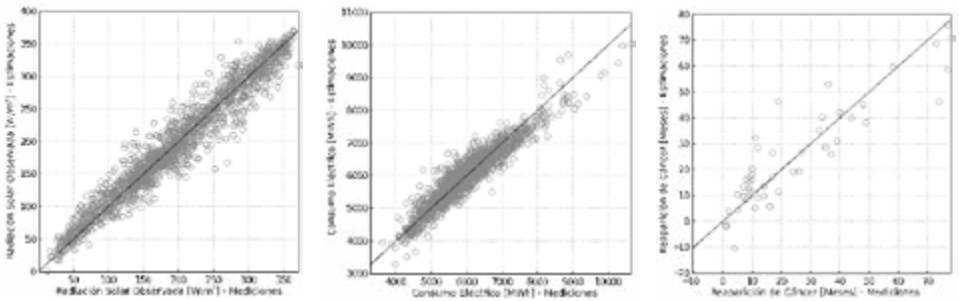


y Deterministic Crowding.

a) b) c)

Figura 6. Gráfico de cajas generados a partir de los valores de fitness obtenidos en 30 corridas de cada algoritmo en el caso de a) Radiación Solar Observada, b) Consumo Eléctrico y c) Tiempo de Reaparición de Cáncer.

La presencia de muchos valores alejados en el caso de Simulated Annealing se debe principalmente a que el algoritmo es muy dependiente del punto inicial utilizado. Es decir, para diferentes soluciones iniciales el algoritmo puede llegar



a converger en diferentes óptimos locales, por lo que el resultado varía entre diferentes ejecuciones.

a) b) c)

Figura 7. Comparativas de estimaciones obtenidas con regresión lineal y datos experimentales reales para a) Radiación Solar Observada diaria en El Colmenar, Tucumán, b) Consumo Eléctrico diario en la provincia de Tucumán y c) Tiempo de Reaparición de Cáncer.

Radiación Solar				
Alg.	**SA**	**sGA**	**pe-cGA**	**w**
SA	-	no	no	no
sGA	no	-	no	si
pe-cGA	no	no	-	no
DC	no	si	no	-
Consumo Eléctrico				
Alg.	**SA**	**sGA**	**pe-cGA**	**DC**
SA	-	no	no	no
sGA	no	-	no	si
pe-cGA	no	no	-	no
DC	no	si	no	-
Reaparición de Cáncer				
Alg.	**SA**	**sGA**	**pe-cGA**	**DC**
SA	-	no	si	no
sGA	no	-	no	no
pe-cGA	si	no	-	no
DC	no	no	no	-

Tabla 3. Resultados de prueba de comparación múltiple que indica cuáles algoritmos arrojan resultados similares analizados de a pares.

La calidad de la predicción realizada con el método de regresión lineal se puede ver en los gráficos de dispersión de la Figura 7, donde se puede apreciar el error

cometido por las estimaciones comparándolas con los valores reales, lo que permite a su vez determinar si el sistema logra captar la linealidad del problema. Para radiación solar se obtuvo un valor RMSE de *19.35* [W/m^2], para consumo eléctrico *289.38* [MWh] y para el problema de células cancerígenas *9.83* [Meses]. En la Tabla 4 se dan detalles de ésta y otras métricas de error logrado por los algoritmos para cada caso, como así también los tiempos de cálculo correspondientes. Se puede ver fácilmente a partir de los valores de Fitness logrados, que el mejor algoritmo en todos los casos fue Deterministic Crowding seguido del Algoritmo Genético simple, y que el primero de ellos requiere menor tiempo de cálculo que el segundo.

Conjunto de datos	Algoritmo	Vars.	Iterac.	Fitness	RMSE	R	Tiempo [seg]
Radiación Solar	-	75	0	-24.89	18.69	0.9763	-
	SA	19	598	-21.25	19.25	0.9748	22.1
	sGA	12	160	-20.81	19.35	0.9746	210.0
	pe-cGA	13	154	-20.90	19.36	0.9745	31.4
	DC	12	45	-20.81	19.35	0.9746	141.5
Consumo Eléctrico	-	126	0	-382.87	279.13	0.9477	-
	SA	60	599	-348.69	288.09	0.9442	24.2
	sGA	40	454	-335.93	289.44	0.9437	365.7
	pe-cGA	53	305	-346.00	289.87	0.9435	54.4
	DC	40	79	-335.85	289.39	0.9437	318.3
Reaparición de Cáncer	-	33	0	-13.82	9.61	0.9040	-
	SA	24	274	-13.27	9.80	0.8999	0.2
	sGA	22	41	-13.12	9.83	0.8993	1.4
	pe-cGA	22	122	-13.12	9.83	0.8993	0.7
	DC	22	35	-13.12	9.83	0.8993	2.8

Tabla 4. Métricas obtenidas de la ejecución de cada algoritmo.

4. Conclusiones

En éste trabajo se presentó una comparativa entre los algoritmos de Simulated Annealing, Algoritmo Genético Simple, Algoritmos Genético Compacto y Algoritmo Genético Niching para Selección de Variables para estimación. Para evaluar el desempeño de cada uno, fueron aplicados a tres instancias del problema: 1) Estimación de Radiación Solar, 2) Consumo de Energía y 3) Tiempo de reaparición de Células Cancerígenas.

Se hizo la comparativa del valor de fitness obtenido en cada caso mediante la

aplicación de un test estadístico de Kruscal-Wallis y test de comparación múltiple. En base a los resultados obtenidos se sacaron las siguientes conclusiones:

- Para el problema de selección de variables para estimación en los casos propuestos, la regresión lineal demostró ser adecuada para lograr buen nivel de error en las estimaciones.
- La performance del Algoritmo Genético Simple y Deterministic Crowding es muy similar, logrando valores muy próximos entre sí en todos los casos. Tanto para Radiación Solar como para Tiempo de Reaparición de Cáncer, el Algoritmo Genético Simple tuvo un mejor desempeño, pero no hay suficiente evidencia estadística que permita establecer cual es mejor entre ellos.
- Los test aplicados demostraron que existe una diferencia significativa entre los resultados de cada algoritmo, arrojando un valor-p pequeño, lo que permite rechazar la hipótesis nula.
- Simulated Annealing fue el algoritmo que tuvo la peor performance en todos los casos de prueba. Los valores de fitness finales logrados con este algoritmo están muy por encima de los obtenidos con algoritmos genéticos.
- En cuanto a tiempos de ejecución de los 3 mejores algoritmos, el pe-cGA requirió menos tiempo, del orden de un 70% más rápido que los demás. Por este motivo, y debido a que cGA tiene una menor cantidad de parámetro para ajustar se concluye que, entre los algoritmos implementados, cGA es el más adecuado para resolver problemas donde el tiempo de ejecución es más importante que la precisión de la solución obtenida.

5. Agradecimientos

Este trabajo fue soportado parcialmente por los subsidios UTI4015 y UTN3870. También deseamos extender los agradecimientos a la Empresa de Distribución Eléctrica de Tucumán S. A. (E.D.E.T.s.a.) y a la Estación Experimental Obispo Colombres (E.E.A.O.C.) por proporcionar los datos necesarios para la realización del trabajo.

6. Referencias

Ahn, C. W., & Ramakrishna, R. S. (2003). Elitism-based compact genetic algorithms. *Evolutionary Computation, IEEE Transactions on*, *7*(4), 367-385. https://doi.org/10.1109/TEVC.2003.814633

Coffin, M., & Saltzman, M. J. (2000). Statistical analysis of computational tests

of algorithms and heuristics. *INFORMS Journal on Computing*, *12*(1), 24–44. https://doi.org/10.1287/ijoc.12.1.24.11899

De Jong, K. A. (1975). *Analysis of the behavior of a class of genetic adaptive systems* (PhD Thesis). University of Michigan.

Denardo, E. V. (2012). *Dynamic programming: models and applications*. Courier Corporation.

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, *1*(1), 3-18. https://doi.org/10.1016/j.swevo.2011.02.002

Eldén, L. (2007). *Matrix Methods in Data Mining and Pattern Recognition*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Harik, G. R., Lobo, F. G., & Goldberg, D. E. (1999). The compact genetic algorithm. *Evolutionary Computation, IEEE Transactions on*, *3*(4), 287–297. https://doi.org/10.1109/4235.797971

Hauschild, M., & Pelikan, M. (2011). An introduction and survey of estimation of distribution algorithms. *Swarm and evolutionary computation*, *1*(3), 111–128. https://doi.org/10.1016/j.swevo.2011.08.003

Jimenez, V. A., Lizondo, D., Will, A., & Rodriguez, S. (2017). Short-Term Load Forecasting for Low Voltage Distribution Lines in Tucumán, Argentina. *5to Congreso Nacional de Ingeniería Informática / Sistemas de Información (CoNaIISI 2017)*, 940-949. Santa Fe, Argentina.

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, *34*(5-6), 975–986. https://doi.org/10.1007/BF01009452

Kotu, V., & Deshpande, B. (2015). Chapter 12 - Feature Selection. En V. Kotu & B. Deshpande (Eds.), *Predictive Analytics and Data Mining* (pp. 347-370). https://doi.org/10.1016/B978-0-12-801460-8.00012-4

Mahfoud, S. W. (1995). Niching methods for genetic algorithms. *Urbana*, *51*(95001), 62–94.

Parsopoulos, K. E. (2010). *Particle swarm optimization and intelligence: advances and applications: advances and applications*. IGI global.

Peng, X., Bessho, M., Koshizuka, N., & Sakamura, K. (2014). A framework for peak electricity demand control utilizing constraint programing method in

smart building. *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*, 744–748. https://doi.org/10.1109/GCCE.2014.7031211

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation* (Vol. 74). John Wiley & Sons.

Varmuza, K., & Filzmoser, P. (2008). *Introduction to multivariate statistical analysis in chemometrics*. CRC press.

Will, A., Bustos, J., Bocco, M., Gotay, J., & Lamelas, C. (2013). On the use of niching genetic algorithms for variable selection in solar radiation estimation. *Renewable Energy*, *50*, 168–176. https://doi.org/10.1016/j.renene.2012.06.039

William H. Wolberg, O. L. M. (1992). *Wisconsin Breast Cancer Database [https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+ (Prognostic)]*.

Williams, M., & Daneshyari, M. D. (2016). *Clonal vs. Negative Selection in Artificial Immune Systems (AIS)*.

Yu, X., & Gen, M. (2010). *Introduction to Evolutionary Algorithms*. En *Decision Engineering*. Springer.