

Estudio del Comportamiento En-Línea de Algoritmos Bio-Inspirados Usando Medidas de Desempeño en Optimización con Restricciones

Blanca Cecilia López-Ramírez y Efrén Mezura-Montes
Laboratorio Nacional de Informática Avanzada (LANIA A.C.)
Rébsamen 80, Centro, Xalapa, Veracruz, 91000, MÉXICO.
blopez@lania.edu.mx, emezura@lania.mx

Resumen

En este artículo se presenta un estudio empírico de algoritmos bio-inspirados para estudiar su comportamiento en-línea (durante el proceso de búsqueda) al resolver un conjunto de problemas de optimización con restricciones usados en la literatura especializada. Se utilizaron: La evolución diferencial, un algoritmo genético, una estrategia evolutiva y la optimización mediante cúmulos de partículas. Se usaron dos medidas de desempeño también encontradas en la literatura especializada, que miden la rapidez de llegada a la zona factible y la capacidad de mejora de la primera solución factible generada. Además, se analizó la convergencia de cada uno de los algoritmos comparados. El objetivo del estudio es identificar la afinidad que tiene cada algoritmo con las características del problema que resuelve y así establecer qué tipo de algoritmo es más conveniente de utilizar de acuerdo a las necesidades del usuario y del problema atacado.

1. Introducción

En el presente trabajo se considera a los algoritmos bio-inspirados (AB's) como el conjunto que engloba a las heurísticas que simulan procesos naturales, como la evolución de las especies y comportamientos sociales de animales o insectos para resolver problemas complejos de búsqueda y optimización. Actualmente, estas técnicas se aplican ampliamente en diversas áreas, tales como la ingeniería, biología, medicina, electrónica, robótica, etc. [3,6].

Los algoritmos evolutivos (AE's) pueden considerarse como un subconjunto de los AB's, basados en la teoría Neo-Darwiniana (simulan en su estructura el proceso de la evolución natural de las especies y la supervivencia del más apto), fueron concebidos inicialmente para solucionar problemas complejos de búsqueda [3]. Tres paradigmas se distinguen como los

pilares del área [3]: Los algoritmos genéticos (AG), las estrategias evolutivas (EE) y la programación evolutiva (PE). Los algoritmos evolutivos más recientes son la Programación Genética [5] (PG) y la Evolución Diferencial (ED) [9]. Por otro lado, ha surgido un área conocida como Inteligencia en Cúmulos (Swarm Intelligence) [4], la cual agrupa heurísticas basadas en el comportamiento social y cooperativo de animales e insectos. De esta área se desprende la Optimización mediante Cúmulos de Partículas (Particle Swarm Optimization, PSO) [4] y la Colonia de Hormigas (Ant Colony Optimization, ACO) [4]. La primera fue propuesta para resolver problemas de optimización numérica y la segunda se aplica principalmente en problemas combinatorios.

Los AB's, en sus propuestas originales, no cuentan con un mecanismo para manejar las restricciones de un problema de optimización. La función de penalización es el enfoque más socorrido [2], sin embargo, la calibración de los factores que determinan la severidad del "castigo" a la aptitud de las soluciones no factibles es normalmente dependiente del problema. Por ello, se han propuesto métodos alternativos [7]. Por otro lado, de acuerdo a la literatura especializada, el trabajo de investigación en esta área (optimización con restricciones) se ha centrado en la generación de mecanismos eficientes que pueden agregar parámetros extra a un AB, y que pueden involucrar la modificación de la heurística en sí. Ejemplos de ello son los siguientes: Mezura et al. [7], utilizaron ED en su variante ED/rand/1/bin, modificándola para que cada vector padre genere más de un vector hijo. Liang y Suganthan[6] modificaron el PSO en su versión Dynamic Multi-Swarm Optimizer para satisfacer mediante sub-cúmulos de partículas las restricciones del problema; además, usan un método clásico de optimización (Sequential Quadratic Programming) como buscador local. Brest [1] modifica a la ED auto ajustando los parámetros durante la evolución. La

técnica mantiene los parámetros originales en un vector inicial y por cada generación produce un nuevo vector con parámetros diferentes de acuerdo a las características del problema.

Por otro lado, escasos son los trabajos relacionados para analizar el comportamiento de los distintos algoritmos en sus versiones originales, para determinar su desempeño y comportamiento propios al resolver problemas de optimización con restricciones. Por ello, se propone utilizar un mecanismo común de manejo de restricciones que no modifique a los algoritmos y así poder determinar las capacidades de cada heurística y la conveniencia de utilizarla de acuerdo al comportamiento deseado.

Se escogió una técnica simple y efectiva para el manejo de restricciones, la cual no agrega parámetros extra. Fue propuesta por Deb [2] y consiste en un conjunto de reglas basadas en factibilidad:

1. Entre 2 soluciones factibles, la solución con un mejor valor de la función objetivo gana.
2. Si una solución es factible y la otra es no factible, la solución factible gana.
3. Si ambas soluciones son no factibles, aquella con el valor más bajo en la suma de la violación de restricción gana.

Este trabajo parte de la hipótesis de que, mediante un estudio empírico general, se pueden establecer comportamientos de cada heurística, en particular al resolver un problema con características determinadas. Por lo tanto, la contribución principal del trabajo es el ofrecer a los interesados, guías para escoger una heurística en particular dependiendo de las características del problema a resolver.

Sabemos de antemano que el estudio empírico cuenta con diferentes factores variables (operadores utilizados, mecanismos de selección, valores de los parámetros, etc.). Sin embargo, la intención del presente trabajo es ser punta de lanza en una serie de estudios que cada vez se irán particularizando más.

Este documento estudia el desempeño de 4 algoritmos representativos: ED, AG, EE y PSO, utilizando un mismo mecanismo para manejar las restricciones del problema, con la intención de establecer los comportamientos de cada algoritmo con respecto al tipo de problema que resuelve. Para ello se analizan tres aspectos: (1) rapidez de llegada a la zona factible, (2) capacidad de mejorar soluciones factibles y (3) convergencia.

El documento está organizado de la siguiente forma: La Sección 2 presenta el diseño del estudio empírico, la Sección 3 incluye los resultados y discusiones. Finalmente, en la Sección 4 se exponen las conclusiones y sugerencias de investigación a futuro.

2. Nuestro estudio empírico

En la sección anterior resaltamos la tendencia en las investigaciones actuales en el diseño de mecanismos para manejar las restricciones y la modificación de la heurística utilizada como motor de búsqueda.

En nuestro estudio investigamos el comportamiento en línea de 3 algoritmos evolutivos y un algoritmo de inteligencia en cúmulos, todos en sus versiones originales y más populares encontradas en la literatura especializada, para conocer su desempeño durante el proceso de búsqueda y así poder establecer la afinidad del comportamiento de la heurística con la función a resolver.

Se utilizaron dos medidas de desempeño encontradas en la literatura especializada: (1) EVALS propuesta por Lampinen [8], cuenta el número de evaluaciones necesarias para localizar la primera solución factible y (2) la medida de desempeño Progress Ratio, propuesta por Mezura y Coello para espacios restringidos [8] determina la capacidad de la técnica para mejorar la primera solución factible encontrada. La fórmula es la siguiente:

$$PR = \left| \ln \sqrt{\frac{f_{\min}(G_{ff})}{f_{\min}(T)}} \right|$$

Donde $f_{\min}(G_{ff})$ es el valor de la función objetivo de la primera solución factible encontrada y $f_{\min}(T)$ es el valor de la función objetivo de la mejor solución factible encontrada en la última generación.

Las características de las heurísticas de estudio son las siguientes: (1) El algoritmo de Evolución Diferencial (ED) versión ED/rand/1/bin con tamaño de población=100, generaciones=3000, CR= aleatoriamente generado entre (0.8, 1.0), F= generado aleatoriamente entre (0.3, 0.9), (2) el Algoritmo Genético (AG) con representación real, tamaño de población=100, generaciones=3000, porcentaje de cruza=0.3, porcentaje de mutación=0.3, cruza aritmética simple, mutación uniforme, selección por torneo binario y reemplazo generacional, (3) una Estrategia Evolutiva $(\mu+\lambda)$ -EE con generaciones=750, recombinación panmítica intermedia aplicados a los parámetros de la técnica y a las variables de decisión, mutación Gaussiana no-correlacional, y (4) un algoritmo de Optimización mediante Cúmulos de Partículas (PSO) versión gbest con tamaño del cúmulo=40, generaciones=7500, fórmula del vuelo con peso de inercia, C1=3.9, C2=0.1, W=1.5.

Sabiendo que cada algoritmo requiere de parámetros diferentes (codificación, operadores evolutivos, mecanismos de selección de supervivencia y reproducción), se buscaron las condiciones más equitativas posibles para su comparación. A continuación se mencionan las condiciones:

- La calibración de sus parámetros fue la que obtuvo los mejores resultados posibles.
- Utilizan el mismo mecanismo para manejar las restricciones del problema (Mecanismo de selección por reglas de factibilidad).
- Codificación real para todos los algoritmos
- 13 funciones de prueba.
- 30 ejecuciones por heurística por función.
- Cada ejecución con 300, 000 evaluaciones para todos los algoritmos.

Los pseudo-códigos de las heurísticas se presentan en la Figura 1 para ED, en la Figura 2 para AG, en la Figura 3 para EE y en la Figura 4 para PSO.

3. Experimentos y resultados

Se utilizaron 13 problemas de prueba encontrados en la literatura especializada [6]. Las características de ellos se presentan en la Tabla 1, donde ρ es una métrica sugerida para estimar el tamaño de la región factible con respecto al espacio de búsqueda y fue estimada de la siguiente manera: $\rho = |F|/|S|$, donde $|F|$ es el número de soluciones factibles y $|S|$ es el total de número de soluciones generadas aleatoriamente, (S=1000000 en este trabajo).

El experimento consiste en analizar la rapidez de llegada a la zona factible de cada algoritmo (EVALS), su desempeño en mejorar esa solución factible (PR) y su gráfica de convergencia.

Los resultados resumidos se encuentran en la Tabla 2 para EVALS y en la Tabla 3 para PR, donde el símbolo "1*" señala al algoritmo que fue el que obtuvo el resultado más competitivo (mejor, media y Desv. Estándar), la técnica con el símbolo "1" es la segunda más rápida en alcanzar la región factible. Cuando se encuentren más de un "1*" en alguna función de prueba significa que esos algoritmos fueron igualmente competitivos. Con la finalidad de obtener mayor detalle en el análisis de los resultados, realizamos una clasificación de las funciones de prueba dividiéndolas en categorías (véase Figura 5).

Con base en dichas categorías realizamos los concentrados de la Tabla 4 y Tabla 5 para EVALS y PR, respectivamente. En dichas Tablas se resume el total de problemas con resultados exitosos para la

correspondiente medida de desempeño de acuerdo a la clasificación de la Figura 5. El número entre paréntesis indica el número de problemas donde se alcanzaron soluciones factibles en cada una de las 30 ejecuciones. Al observar los resultados en la Tabla 4, con respecto a los 13 problemas, a los PFOL, PAD y PRA, PSO fue el más rápido para llegar a la zona factible. PSO fue similarmente efectivo que ED en PFONL, PRIN y PTP, AG muestra rapidez en llegar a la zona factible en PFONL. El algoritmo de EE es el más tardado para localizar la región factible en todas las clasificaciones. Cabe mencionar que ED fue el único algoritmo que consistentemente encontró soluciones factibles para todos los problemas de prueba en cada ejecución independiente.

Tabla 1. Resumen de las características de los 13 problemas aplicadas en los experimentos. n es la dimensionalidad del problema, ρ es el radio estimado de la región factible con respecto al espacio de búsqueda. Las restricciones son representadas como: DL desigualdades lineales, DN desigualdades no lineales, IL igualdad lineal, IN igualdad no lineal, "a" restricciones activas.

P	n	F	ρ	DL	DN	IL	IN	a
g01	13	Cuadrática	0.0111%	9	0	0	0	6
g02	20	No lineal	99.9971%	0	2	0	0	1
g03	10	Polinomial	0.0000%	0	0	0	1	1
g04	5	Cuadrática	52.1230%	0	0	0	0	2
g05	4	Cúbica	0.0000%	2	0	0	3	3
g06	2	Cúbica	0.0066%	0	2	0	0	2
g07	10	Cuadrática	0.0003%	3	5	0	0	6
g08	2	No lineal	0.8560%	0	2	0	0	0
g09	7	Polinomial	0.5121%	0	4	0	0	2
g10	8	Lineal	0.0010%	3	3	0	0	6
g11	2	Cuadrática	0.0000%	0	0	0	1	1
g12	3	Cuadrática	4.7713%	0	1	0	0	0
g13	5	No lineal	0.0000%	0	0	0	3	3

En cuanto al desempeño en PR (Tabla 5), EE es la técnica que demuestra mayor capacidad de mejora de la primera solución factible encontrada, tanto para el caso de los 13 problemas de prueba, como para PFONL, PRIN y PTP. Para el caso de PAD, ED es superior y en PRA, tanto ED, EE y PSO obtuvieron desempeños similares. PSO fue el mejor en PFOL. AG es la técnica menos competente en mejorar la solución para todas las categorías, únicamente mejora las soluciones en el problema g02.

La convergencia de cada algoritmo se analizó con la gráfica de la ejecución en la mediana de las 30 corridas en los resultados finales. Por cuestiones de espacio

mostramos sólo dos funciones que sin embargo, son representativas del comportamiento en las funciones restantes.

Inicio
 Generar una población aleatoria inicial con un tamaño = popsize
 Evaluar cada individuo de la población
Para i = 1 hasta MaxGenerations
 j=0
Mientras(j <= popsize)
 El vector padre “j” genera un vector hijo “j” variante *ED/rand/1/bin*
 Evaluar el vector hijo “j”
 Comparar los vectores padre “j” e hijo “j” con **reglas de factibilidad**
 El mejor permanecerá para la siguiente generación
 j = j + 1
Fin de Mientras
Fin de Para
Fin

Figura 1. Seudo-código del algoritmo ED

Inicio
 Generar una población aleatoria inicial con un tamaño = popsize
 Evaluar cada individuo de la población
Para i=1 hasta MaxGenerations
 j=0
Mientras(j <= popsize)
 Seleccionar 2 individuos de *popsize* utilizando **reglas de factibilidad**
 Aplicar cruce aritmética simple para crear dos hijos “j” y “j + 1”
 Aplicar mutación a “j” y “j + 1”
 Evaluar hijos “j” y “j + 1”
 j = j + 2
Fin de Mientras
 Reemplazo generacional. (Los hijos Permanecen y los padres son eliminados).
Fin de Para
Fin

Figura 2. Seudo-código del algoritmo AG

La gráfica inicia cuando se encuentra la primera solución factible. La Figura 6, muestra la convergencia para el problema g01 donde ED tiene una convergencia más rápida y sin quedar estancada en óptimos locales. La segunda es EE, que converge con mayor lentitud, pero es capaz de aproximar el óptimo global. AG parece converger más rápido que EE, pero se queda estancada en un óptimo local. Finalmente,

PSO es la técnica que entra a la región factible más rápidamente que las otras heurísticas, pero de igual manera queda estancada en un óptimo local.

Inicio
 Generar una población inicial con tamaño = μ
 Evaluar cada individuo en la población
Para i = 1 hasta MaxGenerations
 j=0
Mientras (j <= λ)
 Selecciona 3 individuos aleatoriamente en μ
 Aplicar recombinación para crear un hijo “j”
 Aplicar mutación Gaussiana a “j”
 Evaluar el hijo “j”
 j = j + 1
Fin de Mientras
 Escoger de entre los $\mu + \lambda$ individuos a los μ mejores usando **reglas de factibilidad**
Fin de Para
Fin

Figura 3. Seudo-código del algoritmo EE

Inicio
 Generar una población aleatoria inicial de tamaño = *parsize*
 Evaluar cada individuo en el cúmulo
Para i = 1 hasta MaxGenerations
 Seleccionar el líder del cúmulo utilizando **reglas de factibilidad**
 j=0
Mientras (j <= *parsize*)
 Aplicar fórmula de vuelo a la partícula “j” usando fórmula con factor de inercia.
 Evaluar la nueva posición de la partícula
 Actualizar el valor pbest (memoria) de la partícula “j” usando las **reglas de factibilidad**
 j = j + 1
Fin de Mientras
Fin de Para
Fin

Figura 4. Seudo-código del algoritmo PSO

En la Figura 7, se muestra la gráfica para el problema g03, donde DE y EE muestran un comportamiento similar durante el proceso de búsqueda y alcanzan resultados muy cercanos al mejor reportado. No así

AG y PSO que se quedan estancados con valores muy lejanos al óptimo global.

Tabla 2. Resultados de la medida de desempeño EVALS. El símbolo "1*" representa la técnica más rápida en localizar la región factible, el símbolo "1" es la que segunda más rápida en entrar a la región factible.

EVALS				
.P	ED	AG	EE	PSO
g0 1			1	1*
g0 2	1*	1*	1*	1*
g0 3		1		1*
g0 4	1*	1*	1*	1*
g0 5	1*	-	1	-
g0 6	1*		1	1
g0 7	1	1*		
g0 8	1*	1*		1*
g0 9	1*	1*		1*
g1 0		1		1*
α1	1			1*

Tabla 3. Resultados de la medida de desempeño PR. El símbolo "1*" representa la técnica con el mejor progreso en la zona factible, el símbolo "1" indica el algoritmo con el segundo mejor desempeño.

PR				
.P	ED	AG	EE	PSO
g0 1	1*	1		
g0 2	1	1*		
g0 3			1	1*
g0 4			1*	1
g0 5	1	-	1*	-
g0 6	1		1*	
g0 7	1		1*	
g0 8			1*	1
g0 9	1*			1
g1 0	1			1*
α1			1	1*

Problemas con Función Objetivo Lineal (PFOL)
g10
Problemas con Función Objetivo No lineal (PFONL)
g01, g02, g03, g04, g05, g06, g07, g08, g09, g11, g12, g13
Problemas con Restricciones de Igualdad No Lineal (PRIN)
g03, g05, g11, g13
Problemas con Alta Dimensionalidad ($n \geq 5$) (PAD)
g01, g02, g03, g07, g09, g10
Problemas con un Número Mayor (a 6) de Restricciones Activas (PRA)
g01, g07, g10
Problemas con un Tamaño de ρ de 0.00000 % (PTP)
g03, g05, g11, g13

Figura 5. Categorías de las 13 funciones de prueba

Tabla 4. Número de problemas (por categoría) donde el algoritmo fue competitivo para la medida EVALS.

	EVALS			
	ED	AG	EE	PSO
13 Prob.	8	6 ⁽¹¹⁾	2	9 ⁽¹¹⁾
PFOL (1)	0	0	0	1
PFONL (12)	8	6 ⁽¹⁰⁾	2	8 ⁽¹⁰⁾
PRIN (4)	2	0 ⁽²⁾	0	2 ⁽²⁾
PAD (6)	2	3	1	5
PRA (3)	0	1	0	2
PTP (4)	2	0 ⁽²⁾	0	2 ⁽²⁾

Tabla 5. Número de problemas (por categoría) donde el algoritmo fue competitivo para la medida PR.

	PR			
	ED	AG	EE	PSO
13 Prob.	2	1 ⁽¹¹⁾	7	4 ⁽¹¹⁾
PFOL (1)	0	0	0	1
PFONL (12)	2	1 ⁽¹⁰⁾	7	3 ⁽¹⁰⁾
PRIN (4)	0	0 ⁽²⁾	2	2 ⁽²⁾
PAD (6)	2	1	1	2
PRA (3)	1	0	1	1
PTP (4)	0	0 ⁽²⁾	2	2 ⁽²⁾

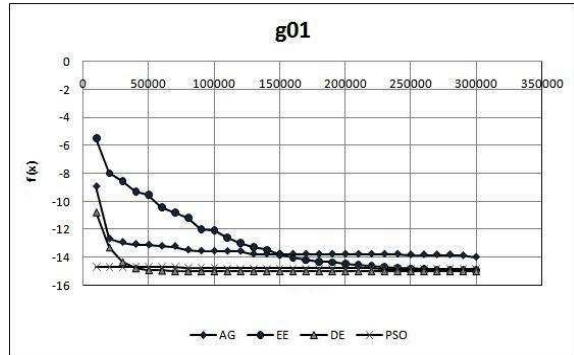


Figura 6. Gráfica de convergencia de la función g01

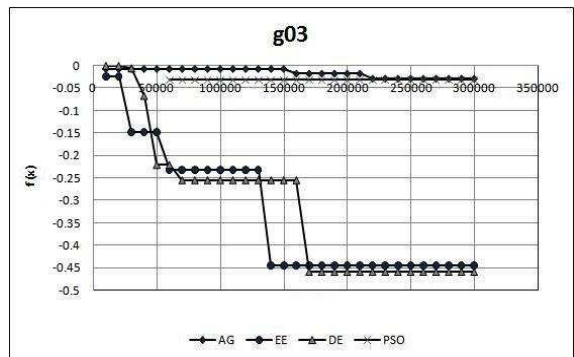


Figura 7. Gráfica de convergencia de la función g03

Como observaciones generales derivadas de los experimentos tenemos las siguientes:

- PSO es el algoritmo que mostró mayor capacidad en la mayoría de funciones de prueba para generar rápidamente soluciones factibles. Sin embargo, para algunos problemas (g05, g13) no pudo generar soluciones factibles y en otros fácilmente queda estancado en óptimos locales, lo cual se confirma con su baja competitividad para mejorar soluciones factibles.

- ED fue rápida para generar soluciones factibles en presencia de una función objetivo no lineal, restricciones de igualdad no lineales o zonas factibles muy pequeñas. También fue la más consistente en generar, para toda problema, soluciones factibles en cada ejecución independiente. ED mostró capacidad para mejorar soluciones factibles en problemas con alta dimensionalidad o con restricciones activas. Finalmente, su comportamiento convergente muestra, en general, que es capaz de evadir óptimos locales.

- EE fue el más lento para alcanzar la zona factible, pero mostró mayor capacidad para, en un corto tiempo, mejorar las soluciones factibles encontradas, sobre todo en problemas con función objetivo no lineal, restricciones de igualdad no lineal y zonas factibles pequeñas. EE mostró una convergencia más lenta que ED, pero con la capacidad de evadir óptimos locales.

- AG alcanzó la zona factible en algunos problemas con función objetivo no lineal, pero mostró problemas para mejorar soluciones factibles previamente encontradas. Además, mostró convergencia prematura en varios problemas.

4. Conclusiones y trabajo futuro

Se ha presentado un estudio empírico del comportamiento en línea de 4 algoritmos bio-inspirados (ED, AG, EE y PSO) al resolver un conjunto de funciones de prueba utilizadas en la literatura especializada. Se utilizaron medidas de desempeño para evaluar la rapidez de llegada a la zona factible y la capacidad para mejorar soluciones factibles. Además, se analizó la convergencia de cada uno de ellos.

Los resultados sugieren que PSO es útil si se quieren encontrar soluciones factibles rápidamente, EE es útil si se quieren obtener mejores resultados pero requiere mayor tiempo para hacerlo. ED mostró un desempeño balanceado en lo que respecta a encontrar soluciones factibles con rapidez y evadir óptimos locales para mejorar significativamente soluciones factibles. Finalmente, AG sólo mostró ser competitivo en algunos problemas con función objetivo no lineal.

Como trabajo futuro se estudiará con detalle la capacidad de los operadores de EE para mejorar soluciones factibles, además, se probarán otros operadores (no tradicionales) para mejorar el desempeño del AG.

Agradecimientos

El primer autor agradece el apoyo del Consejo Nacional de Ciencia y Tecnología (CONACyT) mediante una beca para realizar estudios de Maestría en LANIA. El segundo autor agradece el apoyo de CONACyT mediante el proyecto 52048-Y.

Referencias

- [1] J. Brest, V. Zumer, and M.S. Mau. Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In Proceedings of the Congress on Evolutionary Computation 2006, pages 919–926, IEEE Service Center, July 2006.
- [2] K. Deb. An Efficient Constraint Handling Method for Genetic Algorithms, Computer Methods in Applied Mechanics and Engineering, 186(2/4), pages 311–338, 2000.
- [3] A.E. Eiben. Introduction to Evolutionary Computing, Springer, Germany, 2003.
- [4] A.P. Engelbrecht, Fundamentals of Computational Swarm Intelligence, Wiley, England, 2005.
- [5] J.R. Koza. Genetic Programming, MIT Press, Cambridge, MA, 1992.
- [6] J. J. Liang and P.N. Suganthan. Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism. In Proceedings of the Congress on Evolutionary Computation 2006, pages 316–323, IEEE Service Center, July 2006.
- [7] E. Mezura-Montes, J. Velázquez-Reyes, and C.A. Coello Coello. Modified differential evolution for constrained optimization. In Proceedings of the Congress on Evolutionary Computation 2006, pages 333–339, IEEE Service Center, July 2006.
- [8] E. Mezura-Montes and C.A. Coello Coello. Identifying On-line Behavior and Sources of Difficulty in Constrained Optimization using Evolutionary Algorithms. In Proceedings of the Congress on Evolutionary Computation 2005, pages 1477–1484, IEEE Service Center, September, 2005.
- [9] K.V. Price, R. Storn and J.A. Lampinen. Differential Evolution, Springer, Germany, 2005.