Centro de Investigación en Inteligencia Artificial, Universidad de Veracruz

Evolución Diferencial para Resolver Problemas de Optimización Dinámica con Restricciones

Autor

María Yaneli Ameca Alducin

Tesis sometida para obtener el grado de

Doctor en Inteligencia Artificial

Sometida: 28 de Noviembre de 2016 Matricula: S12015494

Director de Tesis: Dr. Nicandro Cruz Ramírez y co-director
Dr. Efrén Mezura Montes

[página dejada en blanco intencionalmente]

Abstract

The dynamism in the objective function and/or the constraints is an intrinsic feature in the called Dynamic Constrained Optimization Problems (DCOPs). In this dissertation, an algorithm based on combined variants of Differential Evolution that incorporates five mechanisms (change detection, exploration promotion,random inmmigrants, memory population and repair method) was designed in order to cope with a changing environment (called DDECV + Repair). Further, in this work, it was carried out a deep analysis of DDECV + Repair by considering (1) the role of each one of its elements, (2) how affected is with different change frequencies and severities, (3) its ability to detect a change and recover after it, and its diversity handling, and (4) its performance with dynamism in different parts of the problem. Seven performance measures, eighteen recently proposed test problems and eight algorithms found in the specialized literature are considered in four experiments. The statistically validated results indicate that DDECV + Repair is robust to change frequency and severity variations, and that it is particularly fast to recover after a change in the environment, but highly depends on its repair method and its memory population to provide competitive results. Moreover, DDECV + Repair shows evidence on the convenience of keeping a proportion of infeasible solutions in the population when solving dynamic constrained optimization problems. Finally, DDECV + Repair is highly competitive particularly when dynamism is present in both, objective function and constraints.

Resumen

El dinamismo en la función objetivo y/o restricciones es una característica intrínseca de los llamados problemas de optimización dinámica restringida (DCOPs por sus siglas en inglés). En esta Tesis fue diseñado un algoritmo basado en evolución diferencial con combinación de variantes que incorpora 5 mecanismos (detección del cambio, promoción de exploración, inmigrantes aleatorios, población de memoria y método de reparación) con la finalidad de lidiar con entornos dinámicos. Además, en este trabajo se realizó un análisis profundo de DDECV + Repair considerando: (1) El rol de cada uno de sus elementos, (2) como se ve afectado con diferentes frecuencia y severidades del cambio, (3) su habilidad para detectar cambios y recuperarse después de ellos, y su diversidad, y (4) su desempeño ante el dinamismo en diferentes partes del problema. Fueron consideradas siete medidas de desempeño, dieciocho problemas de prueba propuestos recientemente y ocho algoritmos encontrados en la literatura especializada para los cuatro experimentos. La validación estadística de los resultados indica que DDECV + Repair es robusto ante variaciones en la frecuencia y severidad del cambio, y que éste tiene una recuperación rápida después de un cambio en el entorno, pero es altamente dependiente del método de reparación y la población de memoria para proporcionar resultados competitivos. Además, DDECV + Repair muestra evidencia sobre lo conveniente de mantener soluciones no factibles en la población al resolver problemas de optimización dinámica restringida. Finalmente, DDECV + Repair es altamente competitivo, especialmente cuando el dinamismo esta presente tanto en la función objetivo como en las restricciones.

Lista de Publicaciones

Publicaciones derivadas del trabajo de tesis doctoral

- **Publicación en revista**

1. M.-Y. Ameca-Alducin, E. Mezura-Montes, and N. Cruz-Ramírez. Differential evolution with combined variants plus a repair method to solve dynamic constrained optimization problems: A comparative study. *Soft Computing*, pages 1–30, 2016.

- **Publicaciones en conferencias internacionales**

1. M.-Y. Ameca-Alducin, E. Mezura-Montes, and N. Cruz-Ramírez. A repair method for differential evolution with combined variants to solve dynamic constrained optimization problems. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, GECCO '15, pages 241–248, New York, NY, USA, 2015. ACM.
2. M.-Y. Ameca-Alducin, E. Mezura-Montes, and N. Cruz-Ramírez. Differential evolution with a repair method to solve dynamic constrained optimization problems. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference*, GECCO Companion '15, pages 1169–1172, New York, NY, USA, 2015. ACM.
3. M.-Y. Ameca-Alducin, E. Mezura-Montes, and N. Cruz-Ramirez. Differential evolution with combined variants for dynamic constrained optimization. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 975–982, July 2014.

Publicaciones colaterales al trabajo de tesis doctoral

- **Publicaciones en revistas**

1. N. Cruz-Ramírez, H. G. Acosta-Mesa, E. Mezura-Montes, A. Guerra-Hernández, G. Hoyos-Rivera, R.-E. Barrientos-Martínez, K. Gutiérrez-Fragoso, L. A. Nava-Fernández, P. González-Gaspar, Novoa-Del-Toro, V.-J. Aguilera-Rueda, and M.-Y. Ameca-Alducin. How good is crude mdl for solving the bias-variance dilemma? an empirical investigation based on bayesian networks. *PLoS ONE*, 9(3), 2014.

2. N. Cruz-Ramírez, E. Mezura-Montes, M.-Y. Ameca-Alducin, E. Martín-Del-Campo-Mena, H. G. Acosta-Mesa, N. Pérez-Castro, A. Guerra-Hernández, G. Hoyos-Rivera, and R.-E. Barrientos-Martínez. Evaluation of the diagnostic power of thermography in breast cancer using bayesian network classifiers. *Computational and Mathematical Methods in Medicine*, 2013:1–10, 2013.

- **Publicaciones en conferencias internacionales**

1. V.-J. Aguilera-Rueda, M.-Y. Ameca-Alducin, E. Mezura-Montes, and N. Cruz-Ramírez. Particle swarm optimization with feasibility rules in constrained numerical optimization. a brief review. In *Power, Electronics and Computing (ROPEC), 2016 IEEE International Autumn Meeting on*, pages 1–6, Noviembre 2016.
2. M.-Y. Ameca-Alducin, N. Cruz-Ramírez, E. Mezura-Montes, E. Martín-Del-Campo-Mena, N. Pérez-Castro, and H. G. Acosta-Mesa. *Assessment of Bayesian Network Classifiers as Tools for Discriminating Breast Cancer Pre-diagnosis Based on Three Diagnostic Methods*, pages 419–431. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

Índice

Índice de figuras

Índice de tablas

Parte I

Introducción

Capítulo 1

Introducción

En este capítulo se presenta la introducción al documento de tesis, también se describen los antecedentes de la investigación propuesta. Además son establecidos los objetivos de la presente tesis doctoral, así como la hipótesis de investigación a resolver y los aportes que en la misma se realizan. Finalmente, se muestra la estructura de la tesis, detallando el contenido de cada capítulo.

1.1 Antecedentes

En la literatura especializada en problemas de optimización numérica restringida con meta-heurísticas, los algoritmos evolutivos (AEs) han sido ampliamente utilizados en la resolución de dichos problemas [18, 52, 56]. Sin embargo, en los últimos años, la presencia de dinamismo en la función objetivo y/o en las restricciones han atraído el interés de investigadores [53, 64, 62, 66]. Esta clase de problema es conocido como: problema de optimización dinámica restringida (DCOP, por sus siglas en inglés; de aquí en adelante este término será utilizado) [89, 68, 64, 62, 66]. Un DCOP puede ser considerado como un problema de búsqueda, en el cual la función objetivo y/o restricciones cambian a través del tiempo. Los AEs en sus versiones originales no fueron diseñados para hacer frente a entornos dinámicos. Dadas estas condiciones, los AEs tradicionales pueden ser adaptados para identificar cambios en el paisaje de aptitud y/o en la región factible, y de esta manera ser capaces de encontrar los nuevos óptimos [62, 64, 66].

En los últimos veinticinco años, la literatura especializada en AEs ha mostrado un aumento significativo en la investigación sobre optimización dinámica no restringida, p. ej., funciones multimodales [85, 26, 44, 58, 43, 92, 106, 105, 71] y optimización multi-objetivo [47, 8, 49, 37]. Sin embargo, en presencia de restricciones dinámicas, la investigación es escasa [102]. En una revisión reciente sobre AEs para resolver DCOPs, el algoritmo genético (AG) aparece como el algoritmo inspirado en la naturaleza más utilizado [64, 3]. Sin embargo, existen nuevas propuestas basadas en otros algoritmos bio-inspirados, p. ej., evolución diferencial (DE por sus siglas en inglés) en su versión tradicional [3], algoritmo de búsqueda gravitatoria (gravitational search algorithm GSA)[70], algoritmo evolutivo (EA) [88], sistema inmune

artificial T-Cell [7] y algoritmo de multi-población [12]. Para poder hacer frente a espacios dinámicos restringidos, han sido agregados dos clases de mecanismos a los algoritmos previamente mencionados: (i) introducción y/o mantenimiento de la diversidad , p. ej., en [30], donde es presentado un AG con elitismo e inmigrantes aleatorios (RIGAElit), un AG con elitismo e hypermutación (HyperMElit) [16]; y (ii) mecanismos de reparación de soluciones, p. ej., en [64] se utiliza un AG tradicional con un mecanismos de reparación (GA + Repair) [64], DE [69] en su versión original y GSA [70] con mecanismo de reparación similar (DE + Repair y GSA + Repair, respectivamente).

Los algoritmos que tienen integrado un método de reparación suelen superar al resto de los algoritmos sin método de reparación con los que son comparados [69, 70]. La principal característica de los métodos de reparación previamente mencionados consiste en que necesitan de soluciones factibles (que cumplan con todas las restricciones) como referencia para poder convertir las soluciones no factibles en factibles.

En este trabajo doctoral se propone un algoritmo basado en un algoritmo que combina variantes de evolución diferencial que incorporé mecanismos para hacer frente a los cambios en ambientes restringidos (p. ej., método de reparación novedoso que no requiere soluciones factibles), además de un profundo análisis del enfoque propuesto en la resolución de DCOPs.

1.2 Motivación

Como ya se mencionó, en la literatura especializada sobre AEs en problemas de optimización dinámica no restringida existe una cantidad significativa de trabajos de investigación [62, 85, 26, 44, 58, 43, 92, 106, 105, 71, 47, 8, 49, 37]. Sin embargo, en problemas de optimización dinámica *restringida* existe escasa investigación, debido a que es un campo relativamente nuevo, por esta razón genera un nicho de oportunidades a la investigación [64]. Por otra parte, DE es un algoritmo popular, el cual presenta un desempeño altamente competitivo en la resolución de problemas no restringidos [72] y restringidos [54]. Además, DE ha sido poco utilizada para la resolución de DCOPs [69], por lo tanto, unas de las motivaciones de este trabajo es probar la eficiencia de DE con la adaptación de algunos mecanismos utilizados en la resolución de DCOPs.

1.3 Objetivos de la investigación

Objetivo General

Generar un algoritmo basado en una propuesta que combina variantes de evolución diferencial que incorpore mecanismos para resolver problemas de optimización dinámica restringida, el cual obtenga resultados competitivos con respecto a los algoritmos del estado del arte.

Objetivos Específicos

1. Definir los mecanismos a incorporarse para el diseño del algoritmo propuesto.

2. Diseñar e implementar el algoritmo propuesto.

3. Hacer pruebas extensivas al algoritmo con medidas de desempeño encontradas en la literatura especializada.

1.4 Hipótesis de investigación

La hipótesis del presente trabajo de investigación es la siguiente.

Un nuevo Algoritmo evolutivo basado en evolución diferencial con combinación de variantes puede ser diseñado e implementado para resolver problemas de optimización dinámica restringida y su desempeño será mejor que los algoritmos del estado del arte, en al menos una o más medidas de desempeño.

1.5 Contribuciones

Las contribuciones del presente trabajo doctoral son las siguientes.

- La primera adaptación de una propuesta reciente basada en dos variantes de evolución diferencial, la cual originalmente fue diseñada para resolver problemas de optimización estática restringida, dicha adaptación ahora resolverá problemas de optimización dinámica restringida.

- Un análisis profundo del enfoque propuesto, donde los siguientes temas que han sido poco explorados serán investigados.

    1. El papel de cada uno de los elementos del algoritmo en el comportamiento de la búsqueda y su repectivo rendimiento.
    2. Los efectos de (a) la frecuencia de cambio y (b) la severidad en el cambio en ambos, la función objetivo y las restricciones.
    3. Su habilidad para detectar y hacer frente a los cambios, así como su recuperación después de ellos y la manera en la que se maneja la diversidad (es decir, el mantenimiento adecuado de las soluciones factibles durante la búsqueda).
    4. Su desempeño en la resolución de problemas en los que el dinamismo esta presente sólo en la función objetivo, sólo en las restricciones o en ambas.

1.6 Organización del documento de tesis

El documento de Tesis esta compuesto por seis capítulos (incluido el presente), el resto del documento se organiza de la siguiente manera:

Capítulo 2. Entorno dinámico. En este capítulo, se presentan los conceptos básicos de los temas relacionados con esta investigación. Además, se discuten brevemente aspectos relacionados a los entornos dinámicos. También se describen los diferentes mecanismos agregados a EAs para la resolución de DCOPs.

Capítulo 3. Enfoque propuesto. En este capítulo se describe de manera general eñ algoritmo de evolución diferencial, así como la variante basada en DE (llamada DECV) donde se combinan dos variantes. Además, el algoritmo propuesto se introduce y detalla.

Capítulo 4. Evaluación del enfoque. En este capítulo se muestra el diseño experimental empleado en este trabajo de investigación, así como los resultados obtenidos por el algoritmo propuesto.

Capítulo 5. Conclusiones. En este capítulo, se exponen las conclusiones obtenidas en esta tesis doctoral, y de acuerdo a ellas, se presentan los trabajos futuros.

Parte II

Marco Teórico

Capítulo 2

Entornos Dinámicos

En este capítulo se describen las características principales de los entornos dinámicos. Además se presentan algunos conceptos básicos relacionadas a un problema de optimización dinámica. Después se definen los problemas de optimización dinámica restringida, así como el conjunto de funciones de prueba a resolver en esta tesis doctoral. También las principales medidas de desempeño utilizadas en DCOPs. Finalmente, se enlistan los mecanismos que se agregan a los algoritmos inspirados en la naturaleza para hacer frente al dinamismo (función objetivo y/o restricciones).

2.1 Introducción

La oprimización se puede definir como un problema de búsqueda, en el cual se intenta encontrar la mejor solución a dicho problema (maximización o minimización). Este problema puede tener ciertas condiciones y/o restricciones [1]. De manera tradicional, en cualquier proceso de optimización numérica, éste se puede definir en términos de sus variables de decisión [2]., incluidas en un vector de n-dimensional ($\vec{x}$) definido de la siguiente manera:

$$\vec{x} = [x_1, x_2, \ldots, x_n]^T \tag{2.1}$$

Entonces, la meta o el objetivo se define en términos de este vector de decisión. Por lo tanto, de manera formal *optimización* puede ser definida como el proceso de búsqueda para tratar de encontrar las condiciones ideales en un problema, p. ej., el conjunto de valores en el vector de decisión. En esta tesis, el problema a resolver es de minimización de la función objetivo [3]. En caso de requerir maximización, una

[1] Las restricciones son condiciones de un problema particular, las cuales se deben cumplir y éstas definen algunas características del entorno.

[2] Las variables de decisión representan el conjunto de n parámetros del problema, estos valores son modificados durante el proceso de optimización con la finalidad de resolver el problema

[3] La función objetivo es el criterio de evaluación usado para estimar que tan buena es una solución.

opción es transformar el problema en minimización, mediante la multiplicación de la función objetivo por -1 [74].

Otro concepto importante es el de la*optimización global*, la cual consiste en encontrar el óptimo global dentro de algún espacio de búsqueda [23]. De manera formal el óptimo global se puede definir de la siguiente forma.

Definición 2.1.1. Mínimo global

Dada una función $f : X \subseteq \mathbb{R}^n \to \mathbb{R}, X \neq \emptyset$, para $\vec{x} \in X$ el valor $f^* = f(\vec{x}^*)$ es llamado mínimo global si sólo si:

$$\forall \vec{x} \in X : f(\vec{x}^*) \leq f(\vec{x}) \tag{2.2}$$

$\vec{x}^*$ es la solución mínima global, $f(\vec{x}^*)$ es la correspondiente función objetivo y el conjunto X es el espacio de búsqueda.

2.1.1 Entorno dinámico

En el mundo real algunos problemas de optimización pueden ser dinámicos y cambiar sus condiciones con respecto al tiempo. A continuación se presentan algunos ejemplos de este tipo de problema en el mundo real: optimización dinámica en procesos de fermentación de alimentación discontinúa [83], problemas de programación hidrotermal [50], problemas de movimientos de carga [35], problemas de control óptimo [86], entre otros.

A los entornos dinámicos también se les conoce como problema dependiente del tiempo o problema dinámico. En el contexto de problemas de optimización, el problema dinámico es aquel en donde el paisaje de aptitud cambia con el tiempo. [4] Formalmente, $\exists\ t, t' \in \tau$, donde τ es el conjunto de todos los pasos del tiempo, y $\exists\ \vec{x} \in S \subseteq \mathbb{R}$ dado que:

$$f(\vec{x}, t) \neq f(\vec{x}, t') \tag{2.3}$$

donde S es el espacio de búsqueda y f es una función dinámica. Por lo tanto, el valor optimo en el paisaje de aptitud puede cambiar su localización, así como el valor de la función objetivo conforme pasa el tiempo[25]. Uno de los elemento importantes que intervienen en los ambientes dinámicos es el *tiempo*, el cual será definido a continuación.

Definición 2.1.2. Unidad de tiempo: cuando en un problema se involucra el tiempo para ser resuelto, es necesario integrar una unidad de tiempo la cual mide los periodos de tiempo involucrados en el problema. Esta unidad representa las duraciones de tiempo necesarias para completar una evaluación de la función objetivo de ese problema. En [14] y [84], el tiempo es discreto y la unidad de tiempo más pequeña es una

[4] El *paisaje de aptitud dinámico* difiere del estático, ya que sus características topológicas cambian con el tiempo[80].

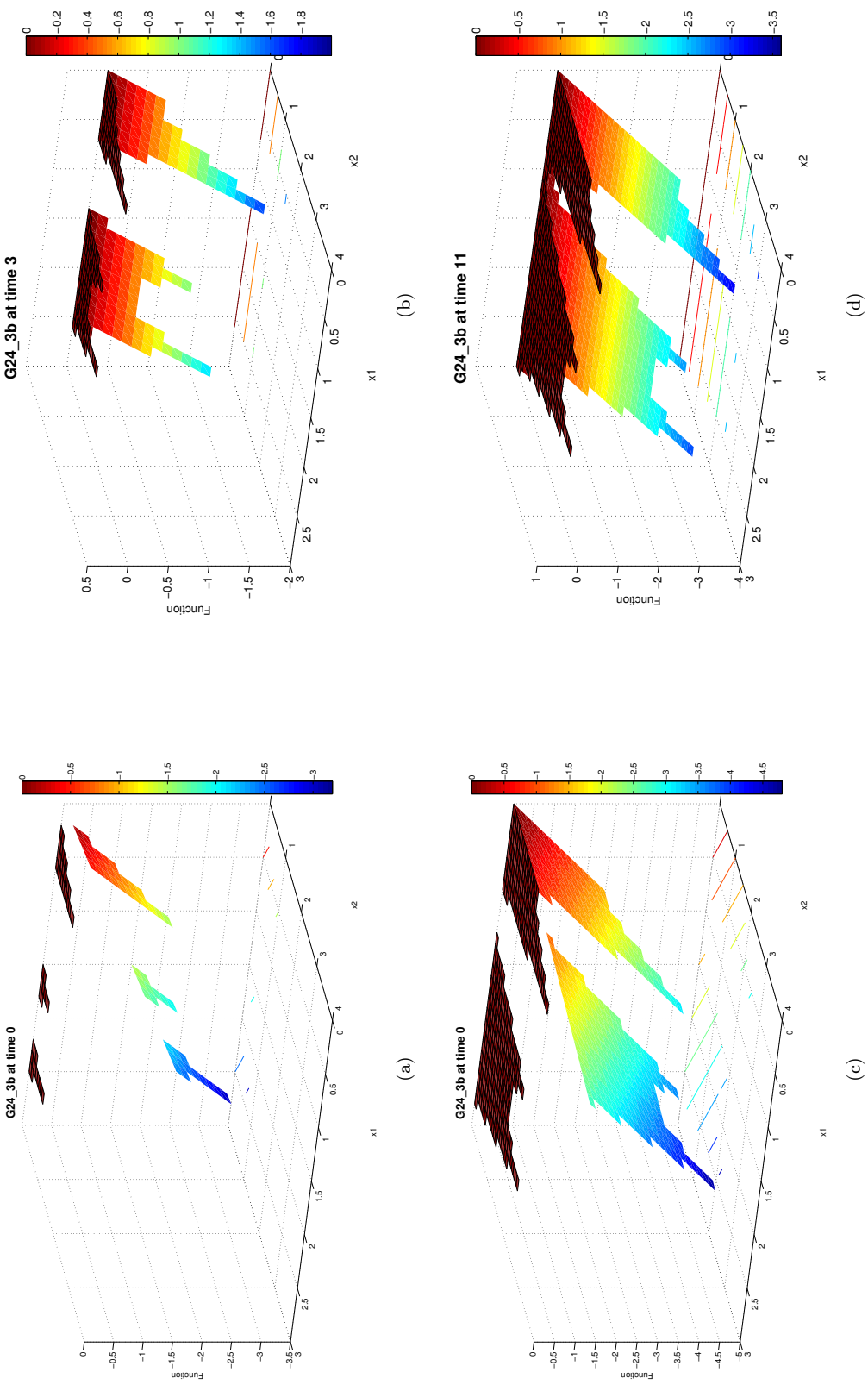
evaluación de la función objetivo. El número de unidades de tiempo que han sido evaluadas desde que se comienza a resolver el problema hasta el tiempo total esta representado por la variable $\tau \in N^+$ [65].

Otro concepto importante inmerso en DCOPs es el cambio del tiempo, el cual se define a continuación.

Definición 2.1.3. Cambio del tiempo: cuando un problema involucra el tiempo en su resolución, *un cambio en el tiempo* representa el instante preciso donde el problema cambia. El número de cambios que se han producido desde que se inicia la resolución del problema hasta el tiempo actual en un problema que involucra el tiempo es representado por la variable $t \in N^+$. Por lo tanto la variable t es el número de cambios realizados desde que se inició la resolución del problema hasta el momento [5].

Otro importante concepto es el *paisaje de aptitud* brevemente mencionado en el ámbito de optimización dinámica. El concepto fue introducido inicialmente en biología teórica para explicar los efectos de las mutaciones genéticas en la supervivencia de los individuos [98]. Este concepto posteriormente fue adoptado en el área de ciencias de la computación para conceptualizar un marco de trabajo, el cual propuso las siguientes preguntas: i)¿ qué tan difícil es resolver un problema específico de optimización para un algoritmo de optimización?, ii) ¿cuál es el comportamiento del algoritmo en ciertas propiedades del problema de optimización?, iii) ¿cuál es el desempeño ideal que se puede esperar de un algoritmo?. Las soluciones potenciales de un algoritmo de optimización pueden moverse dentro del paisaje de aptitud con la finalidad de encontrar regiones con altos valores de aptitud (buenos valores de función objetivo), p. ej., resolviendo el problema de optimización propuesto [39, 40, 33, 51, 90, 80]. Por lo tanto, el paisaje de aptitud en optimización numérica tiene una topología la cual define la forma del espacio de búsqueda de un determinado problema (p. ej., mesetas, valles, número de regiones factibles desconectadas en problemas restringidos, entre otros) [80, 78]. En el caso de entornos estáticos la topología es la misma durante todo el proceso de búsqueda. Sin embargo, en lo que respecta a los entornos dinámicos la topología puede cambiar su forma, porcentaje o estructura de región factible/infactible (en problemas restringidos) [80, 64]. En la Figura 2.1 se muestra un ejemplo de una función dinámica (G24_3b) de un benchmark dinámico basado en la función estática G24 [60], se generó el gráfico de la función en diferentes tiempos, con la finalidad de observar el cambio en el paisaje de aptitud. En esta figura podemos observar cómo cambia el número de regiones desconectadas factibles en cada tiempo, así como el valor del óptimo global que se va modificando.

[5] El cambio del tiempo en esta tesis doctoral esta dado por un número determinado de evaluaciones de la función objetivo.



(a) (b) (c) (d)

Figura 2.1: Ilustra el paisaje de búsqueda factible de la función dinámica G24_3b [60] en cuatro diferentes tiempos: (a) en el tiempo 0, (b)tiempo 3, (c) tiempo 8 y (d) tiempo 11. La función tiene 2 variables de decisión representadas en el gráfico por $x1$ y $x2$. Además, en la barra de la derecha podemos observar los valores de la función objetivo, el último valor indica el mínimo global de la función objetivo para cada tiempo. La figura coloreada representa la región factible de la función. Se observa en las 4 figuras como cambia la región factible así como el valor del óptimo global, el cual está indicado en la barra derecha de cada figura.

2.2 Problemas de optimización dinámica

Los problemas de optimización dinámica (DOPs por sus siglas en inglés), se resuelven en línea mediante un algoritmo de optimización conforme pasa el tiempo [6] .

En [62] Nguyen propone una definición para DOPs, la cual es presentada a continuación.

Definición 2.2.1. Problemas de Optimización dinámica

Dado un problema dinámico $f(\vec{x}, t)$, un algoritmo de optimización A para resolver $f(\vec{x}, t)$, y un tiempo de optimización dado $[t^{begin}, t^{end}]$, $f(\vec{x}, t)$ es llamado *problema de optimización dinámica* en el tiempo $[t^{begin}, t^{end}]$, si durante $[t^{begin}, t^{end}]$ el paisaje de aptitud subyacente de A utiliza para representar los cambios de $f(\vec{x}, t)$ y A tiene que reaccionar ante este cambio para proporcionar nuevas soluciones óptimas.

A continuación se presenta una lista de conceptos relativos a los DOPs, los cuales se basan en algunas características que pueden presentar los DOPs [102].

tiempo de ligamiento (Time-linkage). Cuando el futuro comportamiento del problema depende de la solución actual encontrada por un algoritmo o no.

Predictibilidad. En caso de que los cambios presentados en el problema son predecibles o no.

Visibilidad del cambio. Cuando el cambio es visible para el algoritmo de optimización usado en la resolución del problema mediante la utilización de algunos detectores del cambio (descritos en secciones posteriores).

Problemas restringidos. En caso de tener restricciones o no en el problema (Esta característica es el el principal objetivo a resolver en la tesis).

Número de objetivos. Cuando el problema a resolver tiene sólo un objetivo o es multi-objetivo (tiene más de un objetivo).

Tipo de cambios. Es utilizado para explicar cómo ocurre el cambio en el espacio de búsqueda (si se presenta severo o no el cambio).

Cíclico. Cuando los cambios son cíclicos o recurrentes en el espacio de búsqueda o no.

Periodicidad. Si los cambios son periódicos o no.

Factores que cambian. Los cambios se pueden presentar en: i)la función objetivo, ii)las restricciones, iii) en el dominio de las variables, y iv) en otros parámetros.

[6] En ocasiones entorno dinámico y problemas de optimización dinámica no se distinguen o son utilizados de manera intercambiable [102].

Dadas las características previamente mencionadas de los problemas de optimización dinámica, para poder resolverlos, es necesario utilizar meta-heurísticas adecuadas que puedan lidiar con el dinamismo. Estas meta-heurístas serán descritas en secciones posteriores.

2.3 Problemas de 0ptimización dinámica restringida (DCOPs)

En la literatura especializada en algoritmos evolutivos (AEs), la optimización dinámica no restringida ha sido ampliamente investigada [85, 62]. Sin embargo en presencia de restricciones dinámicas existe una escasa investigación. Los DCOPs son un tipo de problemas de optimización restringida que tienen 2 propiedades: 1) la función objetivo, las restricciones, o ambas pueden cambiar con el tiempo, ii) Los cambios que se presentan son tomados en cuenta en el proceso de optimización [61]. Por lo tanto, un DCOP puede ser visto como un problema de búsqueda donde su paisaje de aptitud y la región factible cambia con el tiempo, el cual sin perder generalidad puede definirse de la siguiente manera:

Encontrar $\vec{x}$, en el tiempo t, tal que:

$$\min_{\vec{x} \in F_t \subseteq [L,U]} f(\vec{x}, t)$$

donde $t \in N^+$ es el tiempo actual,

$$[L,U] = \{\vec{x} = (x_1, x_2, ..., x_n) | L_i \leq x_i \leq U_i, i = 1 \dots n\}$$

es el espacio de búsqueda,
sujeto a:

$$F_t = \{\vec{x} | \vec{x} \in [L,U], g_i(\vec{x}, t) \leq 0, i = 1 \dots m,$$
$$h_j(\vec{x}, t) = 0, j = 1 \dots p\}$$

donde F_t es llamada la región factible del tiempo t.

$\forall \vec{x} \in F_t$ si existe una solución $\vec{x}^* \in F_t$ tal que $f(\vec{x}^*, t) \leq f(\vec{x}, t)$, entonces $\vec{x}^*$ es llamada solución optima factible y $f(\vec{x}^*, t)$ es llamada valor óptimo factible en el tiempo t.

Se definen cuatro tipo de DCOPs [64, 3]: i) función objetivo estática y restricciones estáticas (es decir, problema de optimización restringida estático), ii) función objetiva dinámica y restricciones estáticas, iii) función objetivo estática y restricciones dinámicas y iv) función objetivo dinámica y restricciones dinámicas.

2.3.1 Problemas de prueba en DCOPs

En DCOPs la función objetivo y las restricciones pueden cambiar en el tiempo, así como combinarse en 4 diferentes tipo mencionados en la sección anterior. En los cuatro tipos de combinaciones, la presencia

de áreas no factibles puede afectar la forma en que aparecen los movimientos del óptimo global factible después que se presenta un cambio. Ello conlleva a algunas características que no se encuentran presentes en los problemas no restringidos [60]. Las características especiales son: (i) el dinamismo en las restricciones puede dar lugar a cambios en la forma o el porcentaje o la estructura de la región factible/no factible, (ii) el dinamismo en la función objetivo puede causar que el óptimo global cambie, p. ej., de una región factible desconectada a otra, (iii) en casos donde la función objetivo es estática y las restricciones son dinámicas, se pueden exponer nuevas áreas no factibles cuando un cambio se presenta y con ello un mejor óptimo global cuya característica sería la no factible, por lo tanto el valor de la óptima factible no cambia, y (iv) las características comunes de los problemas con restricciones es que puede estar presente el óptimo global en los límites del espacio de búsqueda; así como tener una o múltiples regiones factibles desconectadas.

En [60] Nguyen propuso un conjunto de funciones de prueba, el cual captura las características comunes de los DCOPs previamente mencionados. Dicho conjunto contiene dieciocho problemas, cada uno con una función objetivo unimodal (a pesar de que la función objetivo es unimodal, algunos problemas tienen múltiples óptimos, debido a la presencia de múltiples regiones factibles desconectadas). Todo conjunto está basado en la función G24, que fue presentada en la competencia de optimización numérica restringida del Congreso de Cómputo Evolutivo del 2006 [46].

Las características principales de los problemas de prueba son presentadas en la Tabla 2.1. En el Apéndice A muestra su descripción.

Tabla 2.1: Características principales de los problemas de prueba [64].

Problema	Función Objetivo	Restricciones	DFR	SwO	bNAO	OICB	OISB	Path
G24_u	Dinámica	Sin restricciones	1	No	No	No	Sí	N/D
G24_1	Dinámica	Estática	2	Sí	No	Sí	No	N/D
G24_f	Estática	Estática	2	No	No	Sí	No	N/D
G24_uf	Estática	Sin restricciones	1	No	No	No	Sí	N/D
G24_2*	Dinámica	Estática	2	Sí	No	Sí y No	Sí y No	N/D
G24_2u	Dinámica	Sin restricciones	1	No	No	No	Sí	N/D
G24_3	Estática	Dinámica	2-3	No	Sí	Sí	No	N/D
G24_3b	Dinámica	Dinámica	2-3	Sí	No	Sí	No	N/D
G24_3f	Estática	Estática	1	No	No	Sí	No	N/D
G24_4	Dinámica	Dinámica	2-3	Sí	No	Sí	No	N/D
G24_5*	Dinámica	Dinámica	2-3	Sí	No	Sí y No	Sí y No	N/D
G24_6a	Dinámica	Estática	2	Sí	No	No	Sí	Difícil
G24_6b	Dinámica	Estática	1	No	No	No	Sí	N/D
G24_6c	Dinámica	Estática	2	Sí	No	No	Sí	Fácil
G24_6d	Dinámica	Estática	2	Sí	No	No	Sí	Difícil
G24_7	Estática	Dinámica	2	No	No	Sí	No	N/D
G24_8a	Dinámica	Sin restricciones	1	No	No	No	No	N/D
G24_8b	Dinámica	Estática	2	Sí	No	Sí	No	N/D

DFR	Número de regiones factibles desconectadas
SwO	El óptimo global cambia entre las regiones desconectadas
bNAO	Un mejor óptimo aparece sin cambiar el existente
OICB	El óptimo global esta en los límites de las restricciones
OISB	El óptimo global en los límites del espacio de búsqueda
Path	Indica si es fácil o difícil para usar mutación para moverse entre las regiones factibles
Dinámica	La función es dinámica
Estática	No hay cambio
*	En algunos períodos, el paisaje de aptitud o es una meseta o contiene infinidad de óptimos y todas las óptimas (incluyendo el óptimo vigente) se encuentran en una línea paralela a uno de los ejes
N/D	No disponible

2.3.2 Medidas de desempeño utilizadas en DCOPs

Con la finalidad de realizar una comparación cuantitativa del desempeño entre los diferentes algoritmos de la literatura especializada que resuelven DCOPs, en este trabajo de tesis doctoral se adopta un conjunto de medidas de desempeño propuesta por Nguyen en [62]. Tales medidas se pueden clasificar en dos grupos: basadas en la optimalidad y basado en el comportamiento.

Medidas de desempeño basadas en la optimalidad

Este tipo de medidas evalúan la habilidad de los algoritmos para encontrar las soluciones con el mejor valor de función objetivo/aptitud (medidas basadas en la aptitud) o encontrar las soluciones más cercanas del óptimo global (medidas basadas en la distancia). Estas medidas se pueden clasificar de la siguiente manera:

Error fuera de línea (Offline error) [11]: Es la medida más popular en la literatura especializada en DCOPs [64, 70]. Se define como el promedio de los errores calculados en cada una de las generaciones que cubren el número total de tiempos. Offline error es más grande o igual a cero. Cuando es igual a cero indica que el algoritmo tiene un desempeño perfecto, lo que significa que el algoritmo es capaz de encontrar el óptimo global desde la primera generación después de que sucede un cambio en el entorno. Esta medida se define en la Ecuación 2.4:

$$offline\_error = \frac{1}{G_{max}} \sum_{G=1}^{G_{max}} e(G) \tag{2.4}$$

donde G_{max} es el número máximo de generaciones calculadas por el algoritmo y $e(G)$ representa el error de la generación actual G (ver Ecuación 2.5):

$$e(G) = |f(\vec{x}^*, t) - f(\vec{x}_{best,G}, t)| \tag{2.5}$$

donde $f(\vec{x}^*, t)$ representa el óptimo global factible en el tiempo actual t, y $f(\vec{x}_{best,G}, t)$ es la mejor solución (factible o no factible) encontrada hasta ahora en la generación G en el tiempo actual t. Utilizando el valor absoluto del error se mitigan los problemas relacionados con la presencia de soluciones no factibles con mejor valor de función objetivo con respecto a la solución óptima factible del tiempo correspondiente.

Error factible fuera de línea (offline error factible) [7]: Esta medida se calcula de manera similar a offline error (ver Ecuación 2.4), con la diferencia de que sólo las soluciones factibles son tomadas en cuenta para realizar el calculo de esta medida. Si una solución no factible es encontrada, entonces esta solución es descartada en el cálculo.

Error fuera de línea modificado (offline error modificado) [64]: Esta medida se calcula de forma similar a offline error (ver Ecuación 2.4) [7], la diferencia radica cundo se agrega $f(\vec{x}_{best,G}, t)$ la mejor solución (factible o no factible) encontrada hasta ahora en la generación G, en caso de que $f(\vec{x}_{best,G}, t)$ sea no factible entonces esta solución es sustituida por la peor solución encontrada en la generación G.

Mejor error antes de un cambio: Esta medida, propuesta en [91], es el promedio de los errores más pequeños calculados al final de cada período de tiempo, justo antes de que se produzca un cambio en el entorno (Ver Ecuación 2.6):

$$E_{best} = \frac{1}{t_{max}} \sum_{t=1}^{t_{max}} e_{best}(G_{max}(t)) \tag{2.6}$$

donde t_{max} es el número total de tiempos, en los que se produce un cambio en el ambiente y cada cambio se produce en el tiempo t, $G_{max}(t)$ es el número total de generaciones al tiempo t y $e_{best}(G_{max}(t))$ es la diferencia entre la solución óptima global factible en el tiempo actual t y la mejor solución encontrada hasta el momento en la generación $G_{max}(t)$ (Véase Ecuación 2.5).

[7] Offline error. Se modifica porque esta medida fue diseñada para entornos no restringidos, por lo tanto tenía que hacer frente a los retos de los entornos restringidos, p. ej., que la mejor solución encontrada en un generación $f(\vec{x}_{best,G}, t)$ sea no factible.

Aunque estas medidas indican qué tan cerca estuvo un algoritmo de encontrar la mejor solución antes de que ocurra un cambio con respecto al óptimo global factible, no proporcionan más información acerca del cómo se alcanzaron dichas soluciones, por esta razón se incluyen las siguientes medidas de desempeño.

Medidas de desempeño basadas en el comportamiento

Las medidas basadas en el comportamiento son aquellas que evalúan si un algoritmo exhibe ciertos comportamientos que se consideran útiles en entornos dinámicos. Un ejemplo de este tipo de comportamiento es cuando un algoritmo es capaz de mantener una alta diversidad durante el proceso de búsqueda; otro comportamiento deseado sería la recuperación rápida de una caída en el desempeño del algoritmo cuando ocurre un cambio en el entorno, pudiendo limitar la caída en la calidad de la función objetivo ante este cambio. Estas medidas son usualmente complementarias a la medidas basadas en la optimalidad, con la finalidad de estudiar el comportamiento de los algoritmos. Su clasificación es la siguiente:

Conteo de cambios_detectados: Cada cambio detectado es contado para verificar si el algoritmo tiene la habilidad para detectar todos los cambios que son presentados durante el proceso de búsqueda.

Porcentaje de individuos no factibles seleccionados: Esta medida calcula el promedio del número de soluciones no factibles en cada generación del número total de generaciones G_{max}. Un valor superior a cero es deseable en la mayoría del proceso de búsqueda, para que el algoritmo sea capaz de mantener soluciones no factibles y así evitar el quedar atrapado en algún óptimo local factible.

Tasa de recuperación (RR) [64]: Esta medida de desempeño fue diseñada para analizar qué tan rápido se recupera un algoritmo después de que ocurre un cambio en el entorno y comienza a converger hacia la nueva mejor solución antes de que el siguiente cambio ocurra. Dado que la nueva solución no necesariamente es la óptima global. El valor de RR puede ser 1 en el mejor caso cuando el algoritmo es capaz de recuperarse y converger inmediatamente hacia la mejor solución después de un cambio. Un valor cercano a 0 indicaría que el algoritmo es incapaz de recuperarse (Ver Ecuación 2.7):

$$RR = \frac{1}{t_{max}} \sum_{t=1}^{t_{max}} \frac{\sum_{G=1}^{G_{max}(t)} [f(\vec{x}_{best,G}, t) - f(\vec{x}_{best,1}, t)]}{G_{max}(t)[f(\vec{x}_{best,G}, t) - f(\vec{x}_{best,1}, t)]} \tag{2.7}$$

donde t_{max} es el número total de tiempos (cambios) en el entorno y cada cambio ocurre en el tiempo t, $G_{max}(t)$ es el número total de generaciones al tiempo t, $f(\vec{x}_{best,G}, t)$ es el mejor valor de la función objetivo de la mejor solución *factible* en la generación actual G del tiempo t, y $f(\vec{x}_{best,1}, t)$ es el mejor valor *factible* de la función objetivo en la primera generación del nuevo tiempo (p. ej., la primera mejor solución obtenida por el algoritmo después de un cambio).

Tasa de recuperación absoluta (ARR) [64]: Es similar a RR, pero es utilizada para analizar qué tan rápido un algoritmo empieza a converger hacia el *óptimo global* antes de que ocurra el próximo

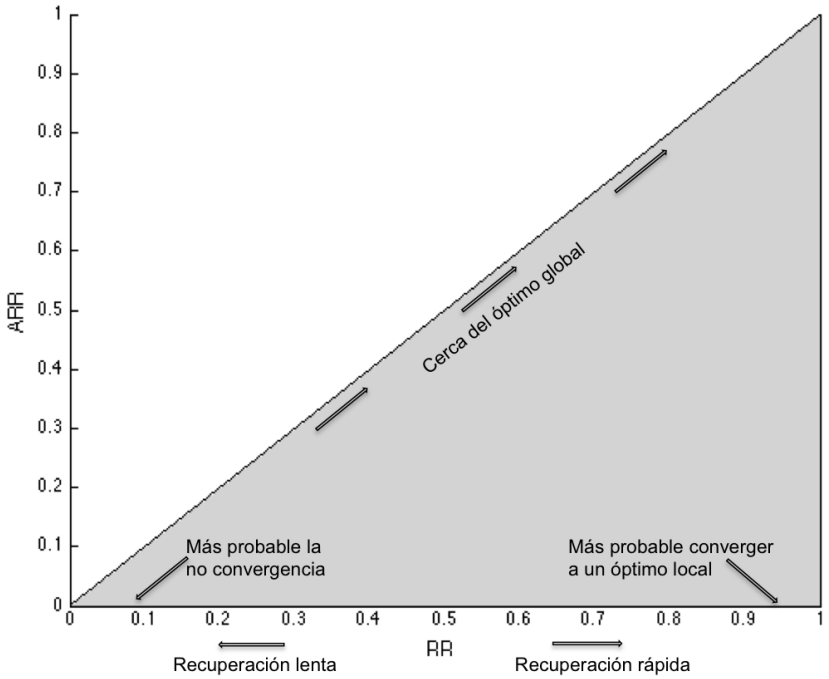


Figura 2.2: Guía para el análisis del comportamiento de convergencia y recuperación de un cambio de un algoritmo dadas los valores de RR/ARR. Los valores de estas medidas para un algoritmo, representan un punto dado en coordenadas (eje x RR y eje y ARR) sobre la línea diagonal o dentro del área sombreada. Cuando el punto se encuentre sesgado a la derecha, indica que el algoritmo en cuestión tiene una convergencia/recuperación más rápida. De lo contrario, el sesgo a la izquierda indica una convergencia/recuperación más lenta. Si el punto se encuentra sobre la diagonal, el algoritmo en cuestión ha sido capaz de recuperarse de un cambio y además, converge hacia el nuevo óptimo global. De lo contrario, si el punto se encuentra en el área sombreada, el algoritmo converge hacia un óptimo local.

cambio. El valor de ARR podría ser 1 en el mejor caso cuando el algoritmo es capaz de recuperarse y converger hacia el nuevo *óptimo global* inmediatamente después de un cambio, y podría ser 0 en caso de que el algoritmo sea incapaz de recuperarse (Ver Ecuación 2.8):

$$ARR = \frac{1}{t_{max}} \sum_{t=1}^{t_{max}} \frac{\sum_{G=1}^{G_{max}(t)} [f(\vec{x}_{best,G}, t) - f(\vec{x}_{best,1}, t)]}{G_{max}(t)[f(\vec{x}^*, t) - f(\vec{x}_{best,1}, t)]} \tag{2.8}$$

donde $f(\vec{x}^*, t)$ es el valor de la óptima global factible en el tiempo actual t. t_{max}, t, $G_{max}(t)$, G, $f(\vec{x}_{best,G}, t)$, y $f(\vec{x}_{best,1}, t)$ fueron definidas en la Ecuación 2.7 para RR.

En [64] Nguyen utiliza por primera vez las medidas de desempeño RR y ARR en conjunto, proponiendo un gráfico para poder visualizar el comportamiento de los algoritmo, en la Figura 2.2, se observa la representación de RR y ARR en conjunto, ya que más adelante en los experimentos realizados en esta tesis doctoral se emplearon estas gráficas.

2.3.3 Mecanismos usados para hacer frente a entornos dinámicos restringidos

En optimización estática restringida, el objetivo principal del algoritmo de optimización es encontrar el óptimo global factible lo más rápido posible. Sin embargo, en optimización dinámica restringida [8] donde el paisaje de aptitud cambia a través del tiempo, los objetivos del algoritmo de optimización dinámica son: (i) detectar los cambios, y (ii) rastrear el nuevo óptimo global después de un cambio. Los algoritmos han agregado algunos mecanismos para hacer frente a los retos previamente mencionados, como los siguientes:

Detección del cambio

Una importante tarea para cualquier algoritmo es el tratar de detectar los cambios inmediatamente cuando éstos ocurren en el entorno; ello debido a que el algoritmo podría tener información obsoleta, o porque en ocasiones el algoritmo reacciona ante un cambio en el entorno realizando tareas específicas. Los mecanismos más comunes para detectar los cambios en el entorno son los siguientes:

Detección de cambios por re-evalución. El mecanismo más popular para la detección de cambios es la re-evaluación de soluciones existentes. Los algoritmos normalmente re-evalúan algunas soluciones específicas para detectar cambios. Este tipo de detección es también conocida como *detección basada en sensor* [76]. Los detectores son parte de la población actual, por ejemplo, una solución aleatoria o la mejor solución o algún miembro de una sub-población [67, 99]. En algunos casos, los detectores también pueden mantenerse separados de la población actual. En estos casos, los detectores pueden se un punto fijo, o ser un conjunto de soluciones aleatorias [15, 77, 89].

Por otra parte, debido a que los detectores implican evaluaciones adicionales de la función objetivo y las restricciones [9], la mayoría de los algoritmos existentes que resuelven problemas de optimización dinámica usa sólo uno o un reducido número de detectores. Sin embargo, en problemas donde sólo algunas partes del espacio de búsqueda cambian y en problemas del mundo real, usar un número reducido de detectores podría no garantizar que los cambios sean detectados [67].

Por otra parte, la principal ventaja de este enfoque es que permite una detección robusta, si un número suficiente de detectores es utilizado. Además Richter in [77] demostró que una de las tareas más complicadas es la detección del cambio y la mejor práctica para realizar esta tarea es la re-evaluación de soluciones.

Detección de cambios basada en el comportamiento del algoritmo. Otro enfoque que es usado para la detección de cambios es la identificación de cuando un algoritmo presenta irregularidades en su comportamiento. Una manera de detectar estos cambios es basado en el monitoreo del promedio

[8] Un DCOP puede ser visto como un solo problema de búsqueda en el que un conjunto de problemas de optimización con restricciones deberán ser resueltas durante el proceso de búsqueda [3].

[9] La principal desventaja de estos detectores es el costo adicional considerando que los detectores tienen que ser re-evaluados en cada generación.

de la mejor solución encontrada en un número determinado de generaciones (caída de este promedio) [16]. Otra manera de detectar un cambio en el entorno es basado en la diversidad, y la relación entre la diversidad de los valores de la función objetivo y la tasa de éxito de la detección de cambio, lo que fue estudiado por Morrison en [57].

Otra técnica utilizada para detectar los cambios es la basada en pruebas de hipótesis estadísticas para poder encontrar la diferencia entre la distribución de las poblaciones de dos generaciones consecutivas [77]. La principal ventaja de este enfoque es que no requiere ninguna evaluación adicional de la función objetivo/restricciones. Sin embargo, una desventaja de esta técnica es que puede causar falsos positivos y entonces provocar que el algoritmo reaccione innecesariamente cuando no ocurre un cambio [67].

Introducción de la diversidad

El objetivo principal de un algoritmo que resuelve problemas de optimización estática restringida es tratar de encontrar el óptimo global lo más rápido posible. Sin embargo en DCOPs la convergencia puede tener efectos negativos, ésto es porque el paisaje de aptitud puede cambiar en algunas zonas y como todas las soluciones están localizadas en un área específica del espacio de búsqueda entonces el algoritmo puede fallar para tratar de moverse hacia el nuevo óptimo global factible. Una solución a esta desventaja es incrementar la diversidad de un algoritmo inmediatamente después de que un cambio es detectado. Una posible solución es descrita en el Algoritmo 1.

Algoritmo 1 Mecanismo_introducción_diversidad

1: G=0 {Generación}
2: Crear una población inicial aleatoria $\vec{x}_{i,G}\,\forall i, i = 1, \ldots, NP$
3: **while** $G \leq G_{max}$ **do**
4: Evaluar cada $\vec{x}_{i,G}\,\forall i, i = 1, \ldots, NP$
5: **if** Si se detecta un cambio (p.ej. una reducción en los mejores valores de la función objetivo o re-evaluación de soluciones anteriores) **then**
6: Incrementar diversidad como: cambio en la mutación (tamaños o tasas) y/o recolocando individuos
7: Evolucionar $\vec{x}_{i,G}\,\forall i, i = 1, \ldots, NP$ utilizando la tasa ajustada de mutación/aprendizaje/adaptación
8: **else**
9: Evolucionar $\vec{x}_{i,G}\,\forall i, i = 1, \ldots, NP$
10: **end if**
11: $G = G + 1$
12: **end while**

La manera más común de introducir la diversidad es la siguiente.

Incrementar la tasa de mutación. Cobb en [16], propuso un operador de mutación adaptativo llamado factor de hiper-mutación cuya tasa de mutación es una multiplicación de la tasa normal de mutación y un factor de hiper-mutación. Esta hiper-mutación es usada sólo si se detecta un cambio.

En algunos algoritmos dinámicos, el porcentaje de mutación se incrementa con la finalidad de moverse hacia la región factible. Cuando la región factible ha sido localizada satisfactoriamente, el porcentaje de mutación se disminuye y retorna hacia su valor previo.

Adición de individuos generados aleatoriamente. En este mecanismo, los individuos (soluciones) son generados aleatoriamente usualmete con distribución uniforme e insertados en la población de búsqueda [99, 29].

Aumentar el tamaño de paso de la mutación. En [94], el tamaño de paso de la mutación [10] es controlado por un rango de búsqueda local variable. Este rango es determinado por una fórmula, cuyo valor es ajustado durante el proceso de búsqueda.

Mover individuos de una población a otra. Un ejemplo de este mecanismo fue propuesto para optimización multi-objetivo dinámica en [29], cuando un cambio es detectado, individuos generados aleatoriamente y algunos individuos competitivos de otras sub-poblaciones son introducidos en cada una de las sub-poblaciones con la finalidad de incrementar la diversidad.

Mantener sub-poblaciones/individuos lejos el uno del otro. Este mecanismo es comunmente utilizado en en optimización por cúmulo de partículas (PSO). Por ejemplo, Hu and Eberhart [99] propusieron un mecanismo simple, en el cual una parte del cúmulo (población) o el cúmulo entero es re-diversificado usando asignación aleatoria inmediatamente después de que un cambio es detectado. Por lo tanto, este método evita que el cúmulo converja a la antigua posición del óptimo global con rapidez.

Los enfoques previamente mencionados para introducir la diversidad presentan un buen desempeño sólo sí los cambios son pequeños o medianos. Esto se debe a que al invocar mutaciones o perturbaciones a los individuos cerca del óptimo, este mecanismo es similar a un tipo de búsqueda local, lo cual sería útil si el óptimo no se mueve muy lejos después de que un cambio ocurre, así nuevo óptimo entonces podría ser rastreado [93, 95].

Los mecanismos para introducir la diversidad tienen las siguientes desventajas: (i)dependen de la detección del cambio, (ii)puede ser complicado determinar la cantidad correcta de diversidad requerida, y (iii) los enfoques pueden no ser efectivos para resolver problemas con cambios aleatorios o con una severidad grande, ya que estos mecanismos se limitan a un rango de búsqueda específico [67].

[10] El tamaño de paso de mutación es usado en el operador de mutación Gaussian [41].

Mantenimiento de la diversidad

Otro mecanismo agregado a los algoritmos dinámicos es el mantener la diversidad dentro de la población durante el proceso de búsqueda, este mecanismo es detallado en el algoritmo 2. Este mecanismo no detecta cambios de manera explicita, por otra parte, en función de la diversidad de forma adaptativa, puede hacer frente a los cambios.

Algoritmo 2 Mecanismo_mantenimiento_diversidad

1: G=0
2: Crear una población inicial aleatoria $\vec{x}_{i,G} \forall i, i = 1, \ldots, NP$
3: **while** $G \leq G_{max}$ **do**
4: Evaluar cada $\vec{x}_{i,G} \forall i, i = 1, \ldots, NP$
5: Mantener la diversidad: agregar un número determinado de nuevos individuos o seleccionar individuos con mayor diversidad o reubicar individuos para mantenerlos alejados unos de otros.
6: Evolucionar $\vec{x}_{i,G} \forall i, i = 1, \ldots, NP$
7: $G = G + 1$
8: **end while**

Los principales mecanismos para mantener la diversidad son los siguientes:

Adición de individuos aleatorios (llamados inmigrantes aleatorios) en la población [30]. En este método, en cada generación un número de individuos generados aleatoriamente son insertados dentro de la población para mantener la diversidad. Además, en [61] fue reportado que un alto nivel de diversidad proveída por los inmigrantes ayuda a la satisfacción de las restricciones.

Distribución específica de algunos individuos (centinelas). En [57], el método de colocación de centinela inicializa un número de soluciones, las cuales son específicamente distribuidas a través del espacio de búsqueda. En este estudio se demostró que el método de colocación de centinelas puede tener mejor desempeño que los inmigrantes aleatorios y la hiper-mutación en problemas con una severidad grande de cambio [57].

Recompensar a los individuos genéticamente diferente de sus padres. En esta propuesta [96], tres poblaciones son usadas, la primera es la población regular, la segunda población mantiene los individuos con una mayor distancia de Hamming (asumiendo representación binaria) con respecto a sus padres (para promover la diversidad), y la tercera contiene los individuos con los mejores valores de función objetivo que sus padres (para promover la explotación).

Dedicando uno de los objetivos para los propósitos de diversidad. En [13], Los objetivos múltiples fueron usados para mantener la diversidad. El problema dinámico es representado como un problema con dos objetivos: (i)objetivo original, y (ii) objetivos especiales creados para mantener la diversidad.

El mantenimiento de la diversidad puede ser bueno en la resolución de problemas con gran severidad en el cambio. Sin embargo, este enfoque tiene algunas desventajas: (i) enfocarse constantemente en la diversidad puede hacer una lenta convergencia o incluso perder el proceso de optimización [38], y (ii) Este enfoque puede ser menos eficaz en la resolución de problemas con pequeña severidad en el cambio [17].

Enfoques de Memoria

Debido a las características de los DOPs, existen cambios en el ambiente, los cuales pueden ser periódicos o recurrentes, por lo tanto, cuando se presentan estas condiciones en un problema dinámico, el óptimo global podría retornar cerca o incluso a la misma región de su localización previa. Ésto puede ser útil, ya que si se mantienen las soluciones anteriores para cambios posteriores en el entorno se podrían tener soluciones prometedoras. La manera más natural de re-utilización de estas soluciones previas, es el incluir una memoria como parte del algoritmo. la memoria puede estar integrada de manera *implícita* como una representación redundante en el algoritmo, o mantenerse *explícita* como un componente separado de memoria.

Estos dos tipos de memoria se describen a continuación.

Memoria implícita. Este tipo de memoria es la más usada en algoritmos evolutivos dinámicos. En esta memoria es redundante usando genomas diploides[11]. Un diploide en algoritmos evolutivos es generalmente un algoritmo cuyos cromosomas contienen dos alelos en cada locus ("lugar") [12]. En [42], se demostró que algoritmos genéticos que usan representaciones diploides son más adecuados en la resolución de problemas donde el entorno cambia con el tiempo, debido a la información adicional guardada en los cromosomas dobles, lo cual puede asegurar la diversidad que a su vez permite responder al algoritmo ante los cambios que se presentan en el paisaje de aptitud. Para el funcionamiento de este tipo de memoria es necesario agregar los siguientes componentes: (i) representación redundante del código, (ii) re-ajuste de la dominancia de los alelos [13], y (iii) detección de cambios. Este mecanismo es detallado en el Algoritmo 3.

Memoria explicita. Este tipo de memoria se puede mantener de las siguientes maneras: (i) directamente (buenas soluciones previas), e indirectamente (información asociativa). Diferentes tipos de memoria asociativa pueden ser integrados como: la probabilidad de que ocurra una buena solución en cada una de las áreas del paisaje de aptitud [82, 81] o regiones factibles probables [79], la probabilidad de que un individuo genere los mejores individuos [104], el individuo exitoso para ciertos

[11]Diploides son organismos que posen dos conjuntos de cromosomas.

[12]Locus es una posición fija en un cromosoma en biología, en computación evolutiva se utiliza para indicar una posición en los alelos

[13]La dominancia de los alelos es usualmente representada por una tabla [59] o mascara [19] de mapeo entre genotipos y fenotipos, entonces la dominancia puede cambiar de manera adaptativa entre alelos dependiendo de la detección de cambios en el paisaje de aptitud.

tipos de cambios [101], la información de la distribución estadística de la población en un tiempo determinado [100]. La idea general del mecanismo de memoria es mostrado en el Algoritmo 4.

Generalmente la mejor solución encontrada en la generación actual sin importar qué tipo de memoria se utilice (directa o asociativa) es usada para actualizar la memoria [97]. Las nuevas soluciones pueden remplazar algunas soluciones existentes en la población de búsqueda, p. ej., la mejor solución de la memoria remplaza la peor solución en la población actual, este remplazo se puede realizar cada generación o por un cierto número de generaciones [97, 103].

Algoritmo 3 Mecanismo_implícito_memoria

1: G=0
2: Crear una población inicial aleatoria $\vec{x}_{i,G}\,\forall i, i = 1, \ldots, NP$ y la representación multiploide.
3: **while** $G \leq G_{max}$ **do**
4: Evaluar cada $\vec{x}_{i,G}\,\forall i, i = 1, \ldots, NP$
5: **if** Se detecta un cambio **then**
6: Ajustar la dominancia para acomodar el cambio actual
7: **end if**
8: Seleccionar los alelos dominantes de acuerdo a su nivel de dominancia
9: Evolucionar $\vec{x}_{i,G}\,\forall i, i = 1, \ldots, NP$ {usando las mutaciones ajustadas}
10: $G = G + 1$
11: **end while**

Algoritmo 4 Mecanismo_explícito_memoria

1: G=0
2: Crear una población inicial aleatoria $\vec{x}_{i,G}\,\forall i, i = 1, \ldots, NP$
3: **while** $G \leq G_{max}$ **do**
4: Evaluar cada $\vec{x}_{i,G}\,\forall i, i = 1, \ldots, NP$
5: Actualizar la memoria
6: Evolucionar $\vec{x}_{i,G}\,\forall i, i = 1, \ldots, NP$
7: Usar la información desde la memoria para actualizar la nueva población
8: $G = G + 1$
9: **end while**

Uno de los objetivos principales de la población es el tratar de mantener soluciones prometedoras debido a las características del problema de optimización dinámica antes de detectar un cambio. Esta solución puede ser utilizada posteriormente, si se presentan condiciones similares en el proceso de búsqueda [66, 3, 5, 4]. La condición previamente mencionada muestra la importancia del enfoque de memoria también llamado población de memoria, porque condiciones de búsqueda presentadas previamente pueden presentarse más adelante en el proceso de búsqueda. Sin embargo, el mecanismo de memoria tiene algunas

desventajas [36, 10]: (i) este mecanismo puede no ser bueno cuando el cambio en el entorno no se presenta de manera frecuente, (ii) sólo es útil cuando las condiciones de búsqueda re-aparecen en los siguientes cambios en el entorno, (iii) la información obtenida en la memoria puede ser obsoleta y redundante, y (iv) en estos mecanismos pueden ser difícil mantener la diversidad en la población.

Enfoque de múltiples poblaciones

En este mecanismo se manejan múltiples sub-poblaciones concurrentemente; puede ser visto como una combinación de diferentes tareas como: (i) introducción/mantenimiento de diversidad, (ii) memoria, y (iii) adaptación. Cada sub-población pude manejar de forma separada cada una de estas tareas. Por ejemplo, en [7], una población se puede enfocar en la diversidad, otra en la exploración en el espacio de búsqueda. El detalle de este enfoque es mostrado en el Algoritmo 5.

Algoritmo 5 Mecanismo_multi_población

1: G=0
2: Crear sub-poblaciones aleatoriamente $\vec{x}_{i,G}\,\forall i, i = 1, \ldots, NP_{search}$ {P_{search} mediante la localización del óptimo global}
3: Crear sub-poblaciones aleatoriamente $\vec{x}_{i,G}\,\forall i, i = 1, \ldots, NP_{track}$ {P_{track} mediante el rastreo de cambios en el paisaje de aptitud}
4: **while** $G \leq G_{max}$ **do**
5: Evaluar cada sub-población
6: Sub-población en P_{search} encontrar el óptimo global
7: Sub-población en P_{track} rastrear cualquier cambio
8: Re-asignar/dividir/fusionar las sub-poblaciones con el fin de que puedan cubrir un área mas grande del espacio de búsqueda
9: Actualizar cada sub-poblaciónn en P_{search} teniendo como base la experiencia de sub-poblaciones en P_{track}
10: Evolucionar cada sub-población en P_{search} y P_{track}
11: $G = G + 1$
12: **end while**

En este enfoque se requieren realizar dos tareas: (i) asignar diferente tareas a cada una de las sub-poblaciones, y (ii) dividir las sub-poblaciones apropiadamente con la finalidad de que las sub-poblaciones cubran una gran área para promover la diversidad. Un ejemplo de este enfoque son los multi-cúmulos en optimización por cúmulo de partículas (PSO). Esta es una de las meta-heurísticas más populares que utiliza el enfoque de multi-poblaciones [45, 9].

El mecanismo de multi-población tiene las siguientes ventajas: (i)pueden mantener la diversidad, porque inician una nueva búsqueda cuando ocurre un nuevo cambio, [7], ii) pueden mantener la información anterior y obtienen buen desempeño en caso de cambios recurrentes [6], iii) estos enfoques son

comúnmente utilizados para resolver problemas multimodales (o también llamados "moving peaks problems") [34]. Sin embargo, estos enfoques tienen las siguientes desventajas: (i) múltiples sub-poblaciones pueden hacer lenta la búsqueda, ii) las sub-poblaciones necesitan garantizar la diversidad, porque si son similares, puede afectar el desempeño del algoritmo.

Parte III

Contribuciones

Capítulo 3

Algoritmo propuesto

En este capítulo se describirá el algoritmo evolutivo dinámico que se propuso e implementó, el cual esta basado en dos variantes de Evolución Diferencial. También se explican los mecanismos que fueron agregados a DE para hacer frente a entornos dinámicos.

3.1 Evolución Diferencial

Evolución diferencial (DE por sus siglas en inglés) es un algoritmo de búsqueda estocástico, que opera con una población de soluciones llamadas vectores [72]. La población es representada como se muestra en la Ecuación 3.1:

$$\vec{x}_{i,G},\ i = 1, \ldots, NP \tag{3.1}$$

donde $\vec{x}_{i,G}$ representa el vector i en la generación G y NP es el tamaño de la población. Cada vector $\vec{x}_{i,G}$ (llamado vector target) genera un descendiente $\vec{u}_{i,G}$ (llamado vector trial) mediante el uso de un vector mutante $\vec{v}_{i,G}$. El vector mutante es obtenido como en la Ecuación 3.2, donde $\vec{x}_{r0,G}$, $\vec{x}_{r1,G}$, y $\vec{x}_{r2,G}$ son vectores seleccionados aleatoriamente de la población actual ($r0 \neq r1 \neq r2 \neq i$); $\vec{x}_{r0,G}$ se le conoce como el vector base y $\vec{x}_{r1,G}$, y $\vec{x}_{r2,G}$ son vectores diferencia. $F > 0$ es el factor de escala definido por el usuario.

$$\vec{v}_{i,G} = \vec{x}_{r0,G} + F(\vec{x}_{r1,G} - \vec{x}_{r2,G}) \tag{3.2}$$

Después de que el vector mutante $\vec{v}_{i,G}$ es generado, éste se combina con el vector target $\vec{x}_{i,G}$ para generar el vector trial $\vec{u}_{i,G}$ a través de la aplicación de un operador de cruza como se muestra en la Ecuación 3.3.

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & \text{Si}(rand_j \leq CR) \text{ o } (j = J_{rand}) \\ x_{i,j,G} & \text{en otro caso} \end{cases} \tag{3.3}$$

donde $CR \in [0, 1]$ define la similaridad entre el vector trial y el vector mutante, $rand_j$ genera un número real aleatorio con una distribución uniforme entre 0 y 1, $j \in \{1, \ldots, n\}$ es la j-ésima variable del vector,

$J_{rand} \in [1, n]$ es un número entero que previene que el trial sea una copia del vector target.

Finalmente, el mejor vector entre el vector target y vector trial, basado en el valor de la función objetivo, es seleccionado para permanecer en la población y pasar a la siguiente generación como se muestra en la Ecuación 3.4 (asumiendo minimización):

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G} & \text{si}(f(\vec{u}_{i,G}) \leq f(\vec{x}_{i,G})), \\ \vec{x}_{i,G} & \text{en otro caso} \end{cases} \tag{3.4}$$

Esta variante de DE es conocida como DE/rand/1/bin, donde "rand" significa el criterio utilizado para la selección del vector base $\vec{x}_{r0,G}$, "1" indica el número de vectores diferencia empleados y "bin" es el tipo de cruza (en este caso es binomial, como en la Ecuación 3.3).

Otra variante de DE es DE/best/1/bin, donde la única diferencia con respecto a la variante previa (DE/rand/1/bin) es que el mejor vector de la población actual, representado como$\vec{x}_{best,G}$, es el vector base para todas las mutaciones diferenciales (véase Ecuación 3.5). Estás dos variantes son las utilizadas en esta propuesta, para más detalles de otras variantes de DE, en particular para optimización restringida, se pueden encontrar en [54].El pseudocódigo de DE es presentado en el Algoritmo 6.

$$\vec{v}_{i,G} = \vec{x}_{best,G} + F(\vec{x}_{r1,G} - \vec{x}_{r2,G}) \tag{3.5}$$

3.2 Evolución diferencial con combinación de variantes (DECV)

El algoritmo propuesto en esta tesis está basado en Evolución Diferencial con Combinación de Variantes (DECV por sus siglas e inglés) [54], donde la variante DE/rand/1/bin es usada al principio de la búsqueda, y después de contar con un porcentaje de vectores factibles (PFV), definido por el usuario, la variante DE/best/1/bin es usada en su lugar. DECV fue inicialmente propuesto para resolver problemas de optimización con restricciones estáticos (SCOPs) [54]. Las reglas de factibilidad propuestas propuestas por Deb [22] son utilizadas en DECV como criterio de selección en la Ecuación 3.4 y también cada vez que es seleccionado el mejor vector en la variante DE/best/1/bin. Las tres reglas son las siguientes:

1. Entre dos vectores factibles, el vector con mejor valor de función objetivo es seleccionado.

2. Si un vector es factible y el otro no es factible, entonces el vector factible es seleccionado.

3. Si ambos vectores no son factibles, entonces el vector con el menor valor de suma de violación de restricciones es seleccionado.

El pseudocódigo completo es detallado en el Algoritmo 7.

Algoritmo 6 Evolución Diferencial (DE/rand/1/bin)

G=0
Crear una población inicial aleatoria$\vec{x}_{i,G}\ \forall i, i = 1, \ldots, NP$
Evaluar $f(\vec{x}_{i,G})\ \forall i, i = 1, \ldots, NP$
for $G \leftarrow 1$ hasta MAX_GEN **do**
  for $i \leftarrow 1$ hasta NP **do**
    Seleccionar aleatoriamente $r0 \neq r1 \neq r2 \neq i$
    $J_{rand} = randint[1, D]$
    for $j \leftarrow 1$ hasta D **do**
      if $rand_j \leq Cr$ o $j = J_{rand}$ **then**
        $u_{i,j,G} = x_{r1,j,G} + F(x_{r2,j,G} - x_{r3,j,G})$
      else
        $u_{i,j,G} = x_{i,j,G}$
      end if
    end for
    if $f(\vec{u}_{i,G}) \leq f(\vec{x}_{i,G})$ **then**
      $\vec{x}_{i,G+1} = \vec{u}_{i,G}$
    else
      $\vec{x}_{i,G+1} = \vec{x}_{i,G}$
    end if
  end for
end for

Algoritmo 7 DECV

1: G=0
2: Crear una población inicial aleatoriamente $\vec{x}_{i,G}\ \forall i, i = 1, \ldots, NP$
3: Evalúa cada $\vec{x}_{i,G}\ \forall i, i = 1, \ldots, NP$
4: $eval = eval + NP$
5: **while** $eval \leq Max\_eval$ **do**
6: Calcular porcentajeFactibles
7: **for** $i \leftarrow 1$ to NP **do**
8: **if** porcentajeFactibles $\leq$ PFV **then**
9: Generar $\vec{u}_{i,G}$ con Ecuaciones 3.2 y 3.3
10: **else**
11: Generar $\vec{u}_{i,G}$ con Ecuaciones 3.5 y 3.3
12: **end if**
13: **if** $f(\vec{u}_{i,G})$ es mejor que $f(\vec{x}_{i,G})$ basado en las reglas de factibilidad **then**
14: $\vec{x}_{i,G+1} = \vec{u}_{i,G}$
15: **else**
16: $\vec{x}_{i,G+1} = \vec{x}_{i,G}$
17: **end if**
18: **end for**
19: $G = G + 1$
20: **end while**

3.3 DDECV + Repair

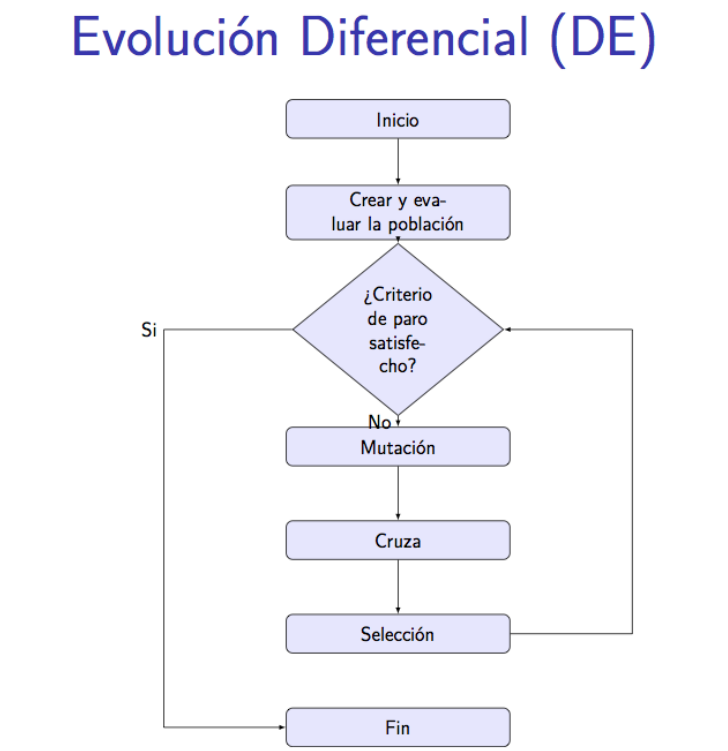
Para hacer frente los DCOPs, la propuesta de esta tesis llamada "Dynamic DECV + Repair" (DDECV + Repair) le fueron agregados mecanismos como: detección de cambios, el cual es capaz de detectar modificaciones en la función objetivo y/o restricciones. Una vez detectado un cambio, DDECV + Repair utiliza las dos variantes de DECV para promover la exploración/explotación en el espacio dinámico restringido de búsqueda. Debido a que es importante promover la exploración después de un cambio, para poder así acercarse lo más posible a una región factible diferente, se aplica a los vectores no factibles un mecanismo de reparación basado en la mutación diferencial y re-muestreo. Finalmente, un conjunto de vectores generados aleatoriamente llamados inmigrantes son agregados para incrementar la diversidad en la población. En la Figura 3.1a se muestra el diagrama de flujo del algoritmo tomado como base (DE) y en la Figura 3.1b se muestra el mismo algoritmo con los mecanismos agregados a la propuesta (DDECV + Repair). En las siguientes secciones se detallan los elementos de DDECV + Repair [4, 5].

3.3.1 Detección del cambio

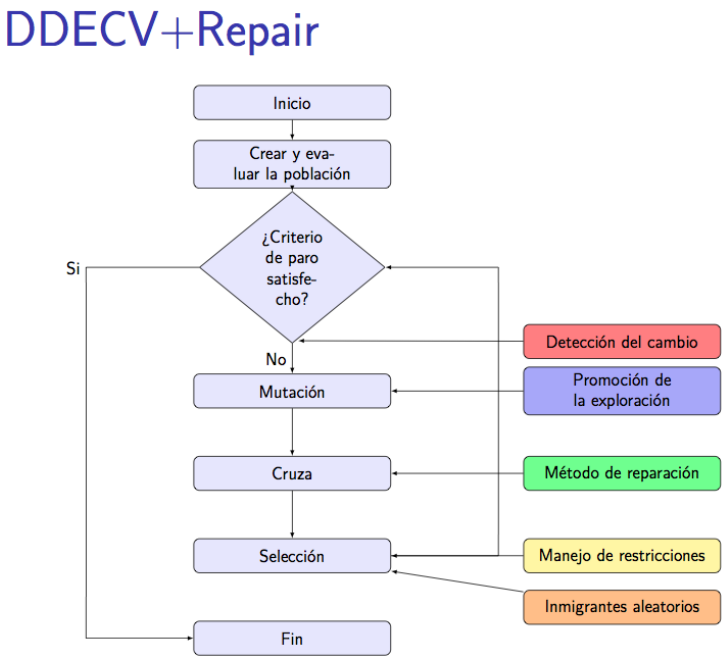
Una detección oportuna del cambio en la función objetivo y/o restricciones de un DCOP, proporciona un buen punto de partida para lidiar con un espacio de búsqueda dinámico [25, 77]. Por lo tanto, DDECV + Repair usa una solución (p. ej., vector) para re-evaluación, también conocido como detección basada en sensor [76]. En cada generación, antes de que el primer vector target y el vector target a la mitad de la población actual generen sus correspondientes trial, son evaluados nuevamente y se comparan sus valores de la función objetivo y restricciones contra sus valores anteriores. Si algún valor es diferente, un indicador es activado y el mejor vector de la población actual es guardado en la memoria, llamado población de memoria. Además, todos los vectores en la población actual y los de la población de memoria son re-evaluados para actualizarse y evitar información obsoleta. La población de memoria mantiene soluciones prometedoras, basándose en las características del DCOP antes del cambio detectado, las cuales se podrían presentar más adelante en la búsqueda dinámica [67]. La intención de usar el primer vector y el localizado a la mitad de la población para detectar un cambio es disminuir la posibilidad de perder un cambio; por esta razón, el mecanismo opera dos veces durante una generación y por lo tanto sólo se calculan dos evaluaciones extras. El pseudocódigo del mecanismo de detección del cambio [5] es detallado en el Algoritmo 8.

Promoción de la exploración

DDECV + Repair, como en la versión original de DECV, comienza usando DE/rand/1/bin. Sin embargo, una vez que un cambio es detectado (véase el Algoritmo 8), entonces es activado el mecanismo de exploración de la siguiente manera: La variante DE es cambiada por la variante DE/best/1/bin, cuyo uso durará varias generaciones definidas por el usuario (Gen_{best}), y el valor de F es incrementado durante el



(a)



(b)

Figura 3.1: Diagrama de flujo del algoritmo propuesto en esta tesis: (a) Elementos de evolución diferencial y (b) Los mecanismos agregados a DE para hacer frente a los cambio (algoritmo DDECV + Repair).

Algoritmo 8 Mecanismo_deteccion_cambio

Require: $\vec{x}_{i,t-1}$

1: Evaluar $\vec{x}_{i,t}$ en el tiempo t
2: **if** algún valor no es el mismo que su evaluación anterior **then**
3: Copiar el mejor vector de la población $\vec{x}_{best,t}$ en la población de memoria
4: Re-evaluar todos los vectores en la población actual y en la memoria
5: $eval = eval + current\_population\_size + memory\_population\_size$
6: **end if**

periodo que dura esta variante en uso, para favorecer movimientos largos y así promover la exploración. Además, considerando que la variante DE/best/1/bin es usada, el mejor vector puede ser seleccionado de la población actual o de la población de memoria (El mejor vector encontrado en entornos previos). La idea de usar la variante DE/best/1/bin con un valor mayor de F promueve la diversidad en un espacio de búsqueda restringido. Esta conclusión fue expuesta en [54] y es adoptada en este trabajo. Los detalles del mecanismo que promueve la exploración se muestran en el Algoritmo 9. En la Figura 3.2 se presenta el funcionamiento de este mecanismo.

Algoritmo 9 Mecanismo_promover_exploracion

1: **if** Contador para usar DE/best/1/bin $< Gen_{best}$ **then**
2: Generar $\vec{u}_{i,t}$ con la Ecuación 3.5 con amplio valor de F y posteriormente con la Ecuación 3.3
3: **else**
4: Generar $\vec{u}_{i,t}$ con las Ecuaciones 3.2 y 3.3
5: **end if**
6: **return** $\vec{u}_{i,t}$

3.3.2 Método de reparación

El proceso de reparación, en el contexto de optimización restringida, es el proceso de convertir una solución no factible en una solución factible. Para poder realizar este proceso se requieren soluciones factibles como referencia. [55, 64, 68, 69, 70]. Sin embargo, el método de reparación usado en DDECV + Repair, no utiliza soluciones factibles para poder funcionar. Es un simple enfoque de re-muestreo basado en el operador de mutación de DE, el cual trabaja de la siguiente forma [5]:

Después de que cada vector trial es generado, si este vector no es factible, entonces tres vectores temporales son generados de manera aleatoria con una distribución uniforme con la finalidad de aplicar el operador de mutación diferencial (véase Ecuación 3.2) de forma similar como el vector de mutación es creada en DE. La factibilidad entonces es verificada al vector obtenido. El proceso se repite hasta que es generado un vector factible o es alcanzado el número máximo de intentos de reparación (Repair_Limit). Independientemente de la factibilidad del vector obtenido después del proceso de reparación, este es

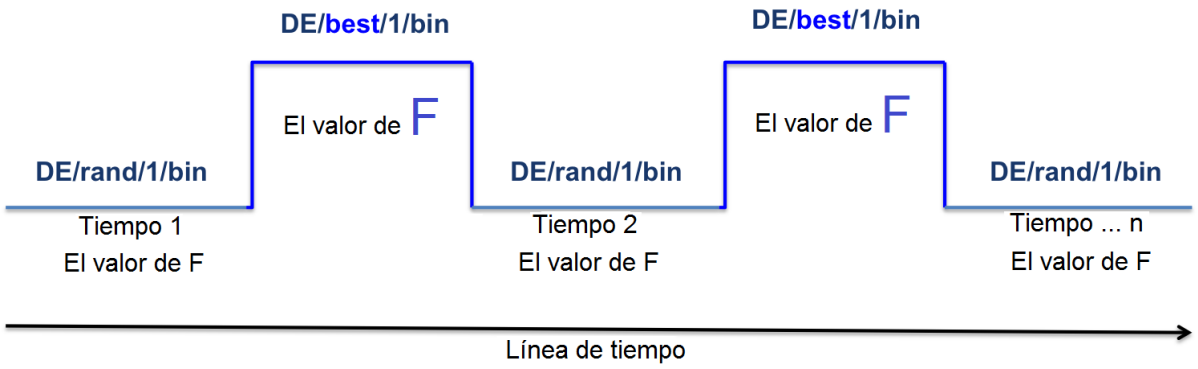


Figura 3.2: Se ilustra el desempeño del mecanismo que promueve la exploración. Después de la detección de un cambio, entonces se cambia de variante (DE/rand/1/bin por DE/best/1/bin). Además el valor de F se aumenta con la finalidad de incrementar la habilidad de exploración del algoritmo propuesto.

considerado como el vector trial para incrementar la diversidad en la población. Debido a que sólo las restricciones son evaluadas por el método de reparación, estas evaluaciones no son agregadas al número total de evaluaciones requeridas por DDECV + Repair. En el Algoritmo 10 se muestran los detalles del método de reparación. En la Figura 3.3 se muestra gráficamente el funcionamiento del método de reparación.

Algoritmo 10 Método_reparación

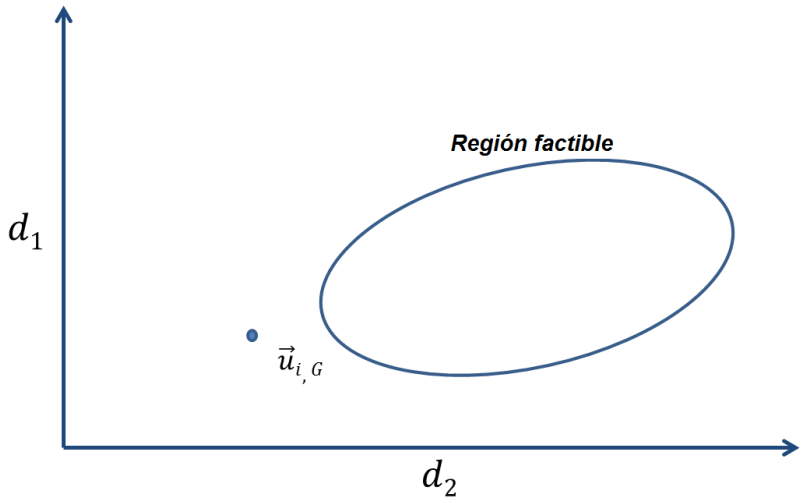
Require: $\vec{u}_{i,G}$ {Vector trial}

1: $counter = 0$
2: **while** $\vec{u}_{i,G}$ no es factible y $counter \leq$ Repair_Limit **do**
3: Generar aleatoriamente tres vectores ($\vec{u}_{r0,G}$, $\vec{u}_{r1,G}$ and $\vec{u}_{r2,G}$)
4: $\vec{u}_{i,G} = \vec{u}_{r0,G} + F(\vec{u}_{r1,G} - \vec{u}_{r2,G})$
5: $counter = counter + 1$
6: **end while**
7: Return $\vec{u}_{i,G}$

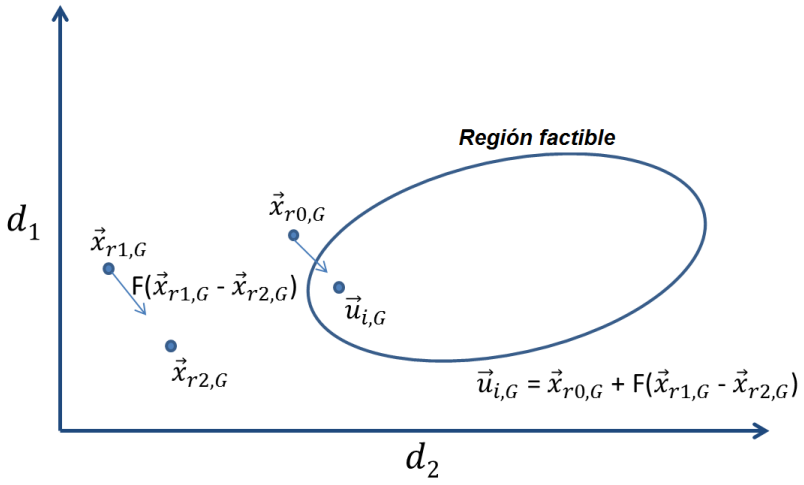
En una versión previa de DDECV + Repair presentada en [3], el método de reparación no estaba incluido, en su lugar era utilizado un mecanismo para promover la convergencia, el cual utilizaba un buscador local (hill-climber [31]). El mecanismo de convergencia fue eliminado y en su lugar se utiliza el método de reparación.

3.3.3 Inmigrantes aleatorios

Para favorecer la diversidad en la población actual de DDECV + Repair, un número de inmigrantes IB (vectores generados de manera aleatoria con una distribución uniforme) se agregan en la población al



(a)



(b)

Figura 3.3: Detalle del funcionamiento del método de reparación. En la Figura (a) se muestra un vector trail no factible (entonces el método de reparación es activado) y en la Figura (b) se muestra un nuevo vector trial factible, el cual fue generado a través de los tres vectores aleatorios.

final de cada generación. IB significa "Inmigrantes antes de un cambio ". Además, durante un periodo de tiempo, el mecanismo de promoción de la exploración está trabajando (controlado por el parámetro Gen_{best}), tal que número de inmigrantes es incrementado (IA, "Inmigrantes después de un cambio "). En ambos casos, los inmigrantes remplazan a lo peores vectores en la población actual.

El seudocódigo de DDECV + Repair es mostrado en el Algoritmo 11.

Algoritmo 11 DDECV + Repair

```
1: G=0
2: Crear aleatoriamente una población inicial x_{i,G} ∀i, i = 1, ..., NP
3: Evaluar cada x_{i,G} ∀i, i = 1, ..., NP
4: eval = eval + NP
5: while eval ≤ Max_eval do
6:    for i ← 1 hasta NP do
7:       if i = 1 o i = NP/2 then
8:          Mecanismo_deteccion_cambio (x_{i,G}) {Algoritmo 8}
9:          eval = eval + 1
10:      end if
11:      u_{i,G} = Mecanismo_promover_exploracion {Algoritmo 9}
12:      if u_{i,G} no es factible then
13:         Metodo_reparacion (u_{i,G}) {Algoritmo 10}
14:      end if
15:      eval = eval + 1
16:      if f(u_{i,G}) es mejor que f(x_{i,G}) basado en las reglas de factibilidad then
17:         x_{i,G+1} = u_{i,G}
18:      else
19:         x_{i,G+1} = x_{i,G}
20:      end if
21:   end for
22:   Agregar IA o IB inmigrantes a la población actual y evaluarlos
23:   eval = eval + IA(or + IB)
24:   G = G + 1
25: end while
```

Capítulo 4

Evaluación del algoritmo propuesto

En este capítulo se presenta el diseño experimental que fue llevado a cabo, mediante el cual se pudo medir el desempeño del enfoque propuesto en el capítulo anterior. Además se discuten los resultados obtenidos de DDECV + Repair en contra de ocho algoritmos encontrados en la literatura especializada en DCOPs.

4.1 Experimentos y Resultados

4.1.1 Diseño experimental

En esta documento de tesis se realizó un análisis profundo de DDECV + Repair (enfoque propuesto) para evaluar su desempeño y comportamiento. Se analizaron los siguientes aspectos: (1) el rol de cada uno de lo los elemento presentes en DDECV + Repair, (2) ¿cómo se afecta el desempeño de DDECV + Repair ante diferentes frecuencias y severidades en el cambio?, (3) la capacidad DDECV + Repair para detectar y reaccionar después de un cambio, así como el manejo de la diversidad, y (4) el desempeño de DDECV + Repair ante el dinamismo en diferentes partes de un problema. Por lo tanto, cuatro experimentos fueron diseñados:

- Una comparación de DDECV + Repair contra sus propias versiones, cada una de las cuales sin uno de los elementos para hacer frente a los cambios (promoción de la exploración, método de reparación de soluciones, inmigrantes aleatorios y población de memoria).

- Una comparación de DDECV + Repair contra algoritmos recientes que resuelven DCOPS, realizando variaciones en la frecuencia y la severidad del cambio.

- Una comparación de DDECV + Repair contra los algoritmos recientes que resuelven DCOPs mediante la medición de los cambios detectados, tasa de recuperación y balance entre soluciones factibles y no factibles en la población.

Tabla 4.1: Valores de los parámetros para los problemas de prueba tomados de [64].

	Número de ejecuciones	50
Benchmark	Número de cambios	5/k
problemas	Frecuencias del cambio	250, 500, 1000, 2000 y 4000 Evaluaciones.
parámetros	Severidad del cambio en la función objetivo k	0.25 (pequeño) 0.5 (mediano) y 1.0 (grande)
	Severidad del cambio en las restricciones S	10 (pequeño), 20 (mediano) y 50 (grande)

- Una comparación de DDECV + Repair contra dos recientes algoritmos ("Intelligent Constraint Handling Evolutionary Algorithm" ICHEA y "Dynamic Constrained T-Cell" DCTC) para analizar la presencia de dinamismo en diferentes partes del problema, p. ej., función objetivo dinámica y restricciones estáticas, función objetiva estática y restricciones dinámicas, y función objetivo y restricciones dinámicas.

Los cuatro experimentos resuelven el conjunto de dieciocho funciones de prueba presentado para DCOPs [64] en el Capítulo 2 Sección 2.3. Los parámetros usados para DDECV + Repair se encuentran en la Tabla 4.2. Estos valores fueron tomados de [5] y obtenidos mediante el uso de la herramienta para calibración de parámetros llamada Irace tool [48]. Los parámetros usados por el conjunto de prueba se detalla en la Tabla 4.1, donde se consideraron diferentes cambios de frecuencia y severidades.

Para evaluar el desempeño de DDECV + Repair se usaron siete medidas de desempeño: (i) offline error, (ii) offline error factible, (iii) mejor error antes de un cambio, (iv) conteo de cambios_detectados, (v) porcentaje de individuos no factibles seleccionados, (vi) tasa de recuperación (RR) y (vii) tasa de recuperación absoluta (ARR). Estas medidas de desempeño fueron descritas en el Capítulo 2 Sección 2.3.

Los resultados obtenidos por DDECV + Repair fueron comparados con los obtenidos por los siguiente ochos algoritmos encontrados en la literatura especializada en DCOPs: (i) GAElit, (ii) HyperMElit, (iii) RIGAElit, (iv) GA + Repair, (v) DE + Repair, (vi) GSA + Repair, (vii) ICHEA y (vii) T-Cell dinámico restringido (DCTC). Estos algoritmos fueron descritos previamente en este Capítulo.

Algoritmos encontrados en la literatura especializada para resolver DCOPs

En esta sección se describen brevemente ocho algoritmos de la literatura especializada en DCOPs, los cuales son utilizados en la comparación con el algoritmo propuesto. Además, en una investigación reciente sobre algoritmos resuelven DCOPs, el algoritmo genético (GA por sus siglas en inglés) es el más popular. Sin embargo han surgido nuevas propuestas basadas en otros algoritmos bio-inspirados,las cuales serán brevemente descritos a continuación.

GAElit es un Algoritmo Genético tradicional con elitismo [1]. En este algoritmo para la selección de

[1]La mejor solución basada en la función objetivo de cada generación pasa a la siguiente generación para promover la

Tabla 4.2: Valores de los parámetros de DDECV + Repair tomados de [3, 5].

Tamaño de la población	25
Cruza	CR = 0.8399
F antes del cambio	F = 0.9644
F después del cambio	FA = 1.0820
Immigrantes antes del cambio	IB = 5
Immigrantes después del cambio	IA = 3
Gen_{best}	16
Repair_Limit (número máximo de intentos de reparación de una solución)	100

padres se utiliza ordenamiento no línael (non-linear ranking), así como cruza aritmética y mutación uniforme. Para hacer frente a las restricciones se emplea penalización estática de funciones. Además, para evitar la información obsoleta se agregó el mecanismo de detección del cambio (ver Algoritmo 8) cuando se aplica el elitismo.

HyperMElit es similar a GAElit, pero con un mecanismo que cambia entre dos porcentajes de mutación: (1) porcentaje bajo (mutación estándar) y (2) porcentaje alto (hipermutación), con la finalidad de incrementar la diversidad. Si la mejor solución empeora, la hipermutación se aplica por un número determinado de generaciones definido por el usuario [16].

RIGAElit también es similar a GAElit. Sin embargo, después de aplicar el operador de mutación, una fracción de la población actual es remplazada con soluciones generadas aleatoriamente (llamadas inmigrantes aleatorios). Esta fracción es determinada por un porcentaje de inmigrantes aleatorios (también llamados porcentaje de reemplazo). En los experimentos realizados en esta tesis, la tercera parte de la población actual es remplazada por inmigrantes aleatorios con la finalidad de mantener la diversidad durante el proceso de búsqueda [30].

GA + Repair fue propuesto en [68]. Este algoritmo es similar a GAElit pero con un método de reparación basado en el empleado en GENOCOP III [55]. El método de reparación convierte soluciones no factibles en la población (llamada población de búsqueda) en factibles utilizando soluciones factibles como referencia. Estas soluciones son llamadas población de referencia donde sólo se permiten soluciones factibles. Para evitar datos obsoletos se aplica un mecanismo de detección de cambios (ver Algoritmo 8).

DE + Repair es basado en la variante DE/rand/1/bin [72] con un mecanismo de detección de cambio,

convergencia [32].

en el cual las soluciones son re-evaluadas con la finalidad de detectar cambios en el entorno. Se utiliza como manejador de restricciones una modificación del método de reparación propuesto por Michalewicz y Nazhiyath [55], cuya aplicación se basa en la cercanía en el espacio de la variable entre la solución referencia (factible) y la solución no factible que se va a reparar [69].

GSA + Repair es basado en al algoritmo de búsqueda gravitacional ("Gravitational Search Algorithm" GSA) [75] y utiliza el mismo mecanismo de detección del cambio y método de reparación que DE + Repair [70].

Algoritmo evolutivo inteligente con manejo de restricciones (ICHEA) es una variación de un algoritmo evolutivo. Este algoritmo usa un operador de cruza entre matrimonios, el cual emplea conocimiento de las restricciones en lugar de hacer una búsqueda ciega de la solución. Esta cruza tiene la intención de generar una solución genérica que satisface más restricciones que sus padres los cuales son seleccionados de dos diferentes regiones factibles [87]. El algoritmo favorece a las soluciones descendientes que satisfacen más restricciones usando las reglas de Deb [22] para ello. Para evitar la pérdida de diversidad en la población, este algoritmo tiene un manejador de estancamiento de soluciones óptimas locales, el cual trabaja como el algoritmo de búsqueda tabú [88].

Célula T dinámica con restricciones (DCTC) es un algoritmo inspirado en el modelo de la célula T, que funciona con cuatro poblaciones, correspondientes a los grupos en los que se dividen las células T: (1) células vírgenes para proporcionar diversidad, (2) células efectoras CD4 para explorar el espacio de búsqueda, (3) células efectoras CD8 para usar representación con números reales y (4) células de memoria para explorar el vecindario de la mejor solución encontrada. Además, utiliza un mecanismo de detección del cambio mediante la re-evaluación de soluciones. Finalmente, las reglas de Deb [22] son utilizadas para hacer frente a las restricciones [7].

4.1.2 Resultados

Experimento 1. Análisis de elementos de DDECV + Repair

El primer experimento compara DDECV + Repair contra sus propias versiones, donde en cada una de ellas, un solo elemento a la vez es desactivado, de la siguiente manera:

- **DDE rand + Repair**: Versión sin mecanismo de promoción de la exploración, es decir, sin la combinación de variantes y la población de memoria, donde sólo la variante DE/rand/1/bin es usada.

- **DDE best + Repair**: Versión sin mecanismo de promoción de la exploración, es decir, sin la combinación de variantes y la población de memoria, donde sólo la variante DE/best/1/bin es usada.

- **DDECV - Repair**: Versión con todos los elementos con excepción del método de reparación.
- **DDECV - Imm + Repair**: Versión todos los elementos con excepción de los inmigrantes aleatorios.
- **DDECV - Mem + Repair**: Versión todos los elementos con excepción de la población de memoria (la mejor solución en los cambios previos).

Los dieciocho problemas fueron resueltos con cada una de las seis versiones y la medida de desempeño calculada fue offline error. La frecuencia en el cambio de 1000 evaluaciones y la severidad del cambio es media (p. ej., k=0.50 y S=20). Estos son los valores más utilizados en la literatura especializada en DCOPs [64]. Los resultados obtenidos se presentan en la Tabla 4.3. La validación estadística fue realizada con la prueba no paramétrica de Kruskal-Wallis (KW) con un 95% de confianza (utilizada para comparar entre múltiples algoritmos) y la prueba post-hoc Bergmann-Hommels. Este tipo de pruebas es sugerida en [24]. Se adoptaron pruebas no paramétricas fueron adoptadas porque las muestras de las ejecuciones no se ajustan a una distribución Gausiana basado en la prueba de Kolmogorov - Smirnov. La prueba estadística KW indica que no existe diferencia significativa entre las diferentes versiones de DDECV + Repair considerando los dieciocho problemas de prueba.

Para tener una evidencia más sólida y precisa, cada problema de prueba se analizó por separado. De tal manera, la prueba no paramétrica por pares de la suma de rangos Wilcoxon con un 95% de confianza fue aplicada por cada problema de prueba en comparaciones por pares entre DDECV + Repair y cada una de sus cinco versiones. Los resultados se muestran en la Tabla 4.4.

Con base en los resultados de la Tabla 4.4, DDECV + Repair supera a DDE_rand + Repair en catorce problemas de prueba (g24_u, g24_1, g24_2, g24_3, g24_3b, g24_4, g24_5, g24_6a, g24_6b, g24_6c, g24_6d, g24_7, g24_8a y g24_8b), mientras que DDE_rand + Repair fue mejor en sólo un problema de prueba no restringido (g24_2u). En tres problemas de prueba (g24_f, g24_uf y g24_3f), todos estáticos y el segundo no restringido no se observaron diferencias significativas entre DDECV + Repair y sus subversiones.

DDECV + Repair superó a DDE_best + Repair en nueve problemas de prueba (g24_u, 24_3, g24_3b, g24_4, g24_6b, g24_6d, g24_7, g24_8a y g24_8b). Por otra parte, DDE_best + Repair tiene un mejor desempeño en seis problemas de prueba (g24_f, g24_uf, g24_2u, g24_3f, g24_6a y g24_6c). No fueron observadas diferencias en tres problemas de prueba (g24_1, g24_2 y g24_5). Es importante remarcar que que los problemas donde DDE_best + Repair tuvo un mejor desempeño son aquellos no restringidos o con restricciones estáticas

Con respecto a DDECV − Repair, los resultados indican que DDECV + Repair superó en trece problemas de prueba (g24_1, g24_f, g24_2, g24_3, g24_3b, g24_3f, g24_4, g24_5, g24_6a, g24_6c, g24_6d, g24_7 y g24_8b). En contraste, DDECV − Repair fue mejor en sólo un problema de prueba no restringido (g24_u). No se observaron diferencias significativas en cuatro problemas de prueba (g24_uf, g24_2u, g24_6b y g24_8a), donde los primeros dos problemas son no restringidos. Estos resultados realzan la importancia

Tabla 4.3: Promedio y desviación estándar de los valores de offline error obtenidos por DDECV + Repair y sus cinco versiones incompletas con frecuencia del cambio de 1000 evaluaciones y una severidad media de cambio (k=0.50 y S=20). Los mejores resultados se resaltan en negritas, aunque de acuerdo a la prueba estadística de Kruskal-Wallis con un 95% de confianza, no se observaron diferencias significativas.

Algoritmos	**Funciones**		
	G24_u	**G24_1**	**G24_f**
DDE_rand+Repair	0.046(±0.007)	0.08(±0.014)	0.023(±0.008)
DDE_best+Repair	0.044(±0.005)	0.063(±0.013)	**0.013(±0.004)**
DDECV−Repair	**0.037(±0.005)**	0.094(±0.024)	0.031(±0.01)
DDECV−Imm+Repair	0.101(±0.025)	0.067(±0.014)	0.02(±0.007)
DDECV−Mem+Repair	0.04(±0.006)	0.075(±0.015)	0.025(±0.007)
DDECV+Repair	0.039(±0.007)	**0.061(±0.01)**	0.021(±0.006)
	G24_uf	**G24_2**	**G24_2u**
DDE_rand+Repair	0.012(±0.004)	0.081(±0.011)	0.035(±0.003)
DDE_best+Repair	**0.006(±0.002)**	0.065(±0.011)	0.033(±0.002)
DDECV−Repair	0.011(±0.005)	0.088(±0.016)	0.031(±0.001)
DDECV−Imm+Repair	0.01(±0.004)	0.067(±0.014)	0.46(±0.249)
DDECV−Mem+Repair	0.011(±0.004)	0.062(±0.011)	**0.031(±0.001)**
DDECV+Repair	0.009(±0.002)	**0.062(±0.006)**	0.036(±0.001)
	G24_3	**G24_3b**	**G24_3f**
DDE_rand+Repair	0.074(±0.006)	0.146(±0.019)	0.013(±0.003)
DDE_best+Repair	0.054(±0.005)	0.118(±0.013)	**0.009(±0.003)**
DDECV−Repair	0.061(±0.009)	0.131(±0.021)	0.025(±0.008)
DDECV−Imm+Repair	**0.042(±0.003)**	0.095(±0.012)	0.011(±0.003)
DDECV−Mem+Repair	0.045(±0.005)	0.101(±0.014)	0.012(±0.004)
DDECV+Repair	0.046(±0.006)	**0.084(±0.006)**	0.01(±0.002)
	G24_4	**G24_5**	**G24_6a**
DDE_rand+Repair	0.144(±0.016)	0.092(±0.012)	0.044(±0.005)
DDE_best+Repair	0.117(±0.013)	0.079(±0.008)	0.034(±0.004)
DDECV−Repair	0.131(±0.02)	0.122(±0.023)	0.065(±0.026)
DDECV−Imm+Repair	0.094(±0.009)	0.38(±0.162)	**0.032(±0.005)**
DDECV−Mem+Repair	0.099(±0.014)	0.082(±0.012)	0.033(±0.007)
DDECV+Repair	**0.088(±0.011)**	**0.078(±0.008)**	0.036(±0.005)
	G24_6b	**G24_6c**	**G24_6d**
DDE_rand+Repair	0.055(±0.01)	0.051(±0.007)	0.099(±0.009)
DDE_best+Repair	0.041(±0.008)	0.039(±0.006)	0.094(±0.009)
DDECV−Repair	0.049(±0.012)	0.051(±0.014)	0.115(±0.024)
DDECV−Imm+Repair	**0.037(±0.006)**	**0.035(±0.005)**	0.111(±0.014)
DDECV−Mem+Repair	0.045(±0.01)	0.042(±0.009)	0.082(±0.011)
DDECV+Repair	0.041(±0.01)	0.041(±0.01)	**0.079(±0.006)**
	G24_7	**G24_8a**	**G24_8b**
DDE_rand+Repair	0.129(±0.018)	0.215(±0.02)	0.145(±0.042)
DDE_best+Repair	0.107(±0.012)	0.138(±0.019)	0.09(±0.034)
DDECV−Repair	0.169(±0.027)	0.159(±0.023)	0.13(±0.035)
DDECV−Imm+Repair	**0.099(±0.013)**	0.283(±0.064)	0.076(±0.024)
DDECV−Mem+Repair	0.114(±0.015)	0.18(±0.021)	0.098(±0.029)
DDECV+Repair	0.107(±0.011)	**0.138(±0.015)**	**0.074(±0.025)**

Tabla 4.4: Resultados de la prueba no paramétrica por pares de la suma de rangos Wilcoxon con un 95% de confianza entre DDECV + Repair contra cada uno de sus cinco versiones incompletas basadas en los resultados del offline error en la tabla 4.3. "+" significa que DDECV + Repair superó la versión en la columna correspondiente. "-" significa que la versión en la columna correspondiente superó a DDECV + Repair. Una diferencia no significativa entre DDECV + Repair y la versión en la columna correspondiente se indica con "=".

Funciones	**Algoritmos**				
	DDE_rand+Repair	**DDE_best+Repair**	**DDECV−Repair**	**DDECV−Imm+Repair**	**DDECV−Mem+Repair**
g24_u	+	+	−	+	+
g24_1	+	=	+	=	+
g24_f	=	−	+	−	+
g24_uf	=	−	=	=	+
g24_2	+	=	+	=	+
g24_2u	−	−	=	+	−
g24_3	+	+	+	−	+
g24_3b	+	+	+	+	+
g24_3f	=	−	+	+	+
g24_4	+	+	+	+	+
g24_5	+	=	+	+	+
g24_6a	+	−	+	=	−
g24_6b	+	+	=	−	+
g24_6c	+	−	+	−	+
g24_6d	+	+	+	+	+
g24_7	+	+	+	−	=
g24_8a	+	+	=	+	+
g24_8b	+	+	+	+	+

del método de reparación en DDECV + Repair.

Además, DDECV + Repair obtiene mejores resultados con respecto a DDECV − Imm + Repair en nueve problemas de prueba (g24_u, g24_2u, g24_3b, g24_3f, g24_4, g24_5, g24_6d, g24_8a y g24_8b), mientras DDECV − Imm + Repair supera los resultados de DDECV + Repair en cinco problemas de prueba (g24_f, g24_3, g24_6b, g24_6c y g24_7). En cuatro problemas de prueba (g24_1, g24_uf, g24_2, y g24_6a) no fueron observadas diferencias significativas.

La característica principal de tres problemas donde la versión sin inmigrantes fue mejor que DDECV + Repair (g24_f, g24_3 y g24_7) es que el óptimo global está en las fronteras de la región factible, mientras que en los otros dos problemas de prueba (g24_6b y g24_6c), el óptimo factible global está en la fronteras del espacio de búsqueda. Por lo tanto, la adición de estas soluciones generadas aleatoriamente dentro de la población actual parece afectar la convergencia de las soluciones situadas en la frontera de la región factible o del espacio de búsqueda.

Finalmente, DDECV + Repair supera a DDECV − Mem + Repair en quince problemas de prueba (g24_u, g24_1, g24_f, g24_uf, g24_2, g24_3, g24_3b, g24_3f, g24_4, g24_5, g24_6b, g24_6c, g24_6d, g24_8a y g24_8b), mientras que DDECV − Mem + Repair tiene un mejor desempeño en dos problemas de prueba (g24_2u y g24_6a), donde el primer problema no es restringido y el segundo tiene restricciones estáticas. No fueron observadas diferencias significativas en sólo un problema de prueba (g24_7), donde la principal

característica es que el óptimo global esta en la frontera de la región factible. Los resultados sugieren la importancia de la población de memoria, porque las condiciones previas de la búsqueda aparecen después en el proceso de búsqueda.

Los resultados generales de este primer experimento derivan las siguientes conclusiones:

- El mecanismo más importante en DDECV + Repair es precisamente el mecanismo de reparación. Sin embargo, incluso sin él, el algoritmo puede proveer resultados competitivos pero sólo en problemas dinámicos no restringidos.
- La ausencia del mecanismo de promoción de diversidad también puede afectar el desempeño de DDECV + Repair, pero su impacto es poco significativo. De hecho, incluso sin él, al usar sólo DE/best/1/bin, en problemas con función objetivo dinámica pero con restricciones estáticas los resultados siguen siendo competitivos.
- Los inmigrantes aleatorios tienen un efecto positivo sobre DDECV + Repair. Sin embargo la presencia de soluciones aleatorias en cada generación puede afectar el desempeño del algoritmo cuando el óptimo global está localizado en las fronteras de la región factible o del espacio de búsqueda.
- Otro importante mecanismo en DDECV + Repair es la memoria de población. Sin ella, el desempeño de DDECV + Repair es particularmente afectado.

Experimento 2. Análisis de la frecuencia y severidad del cambio

El segundo experimento analiza los efectos de diferentes frecuencias del cambio y también diferentes severidades del cambio en el desempeño de DDECV + Repair. En este experimento, para la comparación fueron utilizados seis algoritmos previamente descritos y sus resultados fueron tomados de sus correspondientes documentos [16, 17, 62, 63, 69, 70] y se resuelven los dieciocho problemas, como en el Experimento 1.

La primera comparación de este segundo experimento se enfoca en la frecuencia del cambio. Los resultados fueron validados con la prueba no paramétrica de Kruskal-Wallis (KW) con un 95% de confianza, aplicada a los resultados (valores de offline error) obtenidos por DDECV + Repair y los algoritmos comparados (GAElit, RIGAElit, HyperMElit, GA + Repair, DE + Repair y GSA + Repair) con cinco diferentes cambios de frecuencia y una severidad del cambio mediana (k=0.5 y S=20). En la Tabla 4.10 se presentan las frecuencias de 250, 500, 2000, y 4000 evaluaciones, y en la Tabla 4.11 se muestra la frecuencia de 1000 evaluaciones, la cual es separada porque DE + Repair y GSA + Repair sólo reportan resultados para esta frecuencia del cambio.

Los detalles estadísticos descriptivos completos de estas comparaciones (valores de offline error) se encuentran en las Tablas 4.5, 4.6, 4.7, 4.8, y 4.9, para 250, 500, 1000, 2000, y 4000 evaluaciones, respectivamente.

Tabla 4.5: Promedio y desviación estándar de los valores de offline error obtenidos por DDECV + Repair y los algoritmos comparados con una frecuencia del cambio de 250 evaluaciones y una severidad media del cambio (k=0.5 y S=20). Los mejores resultados se resaltan en negritas.

Algoritmos	**Funciones**		
	G24_u	**G24_1**	**G24_f**
GAElit	0.265(±0.05)	0.781(±0.168)	0.326(±0.161)
RIGAElit	0.316(±0.056)	0.706(±0.12)	0.351(±0.129)
HyperMelit	0.274(±0.045)	0.667(±0.12)	0.34(±0.109)
GA+Repair	0.674(±0.099)	0.414(±0.099)	0.125(±0.052)
DDECV+Repair	**0.179(±0.013)**	**0.269(±0.021)**	**0.097(±0.027)**
	G24_uf	**G24_2**	**G24_2u**
GAElit	0.177(±0.055)	0.469(±0.081)	0.234(±0.066)
RIGAElit	0.202(±0.06)	0.386(±0.055)	0.195(±0.035)
HyperMelit	0.185(±0.069)	0.434(±0.061)	0.198(±0.045)
GA+Repair	0.463(±0.152)	0.412(±0.06)	0.46(±0.071)
DDECV+Repair	**0.039(±0.018)**	**0.245(±0.011)**	**0.189(±0.01)**
	G24_3	**G24_3b**	**G24_3f**
GAElit	0.646(±0.179)	0.857(±0.188)	0.35(±0.188)
RIGAElit	0.655(±0.126)	0.797(±0.094)	0.413(±0.12)
HyperMelit	0.562(±0.13)	0.732(±0.108)	0.364(±0.163)
GA+Repair	0.115(±0.027)	0.382(±0.098)	0.059(±0.022)
DDECV+Repair	**0.11(±0.013)**	**0.28(±0.03)**	**0.038(±0.011)**
	G24_4	**G24_5**	**G24_6a**
GAElit	0.844(±0.112)	0.452(±0.07)	1.179(±0.232)
RIGAElit	0.773(±0.114)	0.398(±0.062)	0.775(±0.138)
HyperMelit	0.738(±0.105)	0.398(±0.048)	0.83(±0.095)
GA+Repair	**0.252(±0.058)**	0.263(±0.046)	0.785(±0.192)
DDECV+Repair	0.277(±0.029)	**0.25(±0.031)**	**0.191(±0.014)**
	G24_6b	**G24_6c**	**G24_6d**
GAElit	0.814(±0.088)	0.77(±0.097)	0.842(±0.099)
RIGAElit	0.624(±0.071)	0.649(±0.084)	0.75(±0.1)
HyperMelit	0.67(±0.057)	0.708(±0.056)	0.723(±0.081)
GA+Repair	0.738(±0.101)	0.673(±0.076)	0.622(±0.118)
DDECV+Repair	**0.237(±0.028)**	**0.227(±0.021)**	**0.437(±0.029)**
	G24_7	**G24_8a**	**G24_8b**
GAElit	0.631(±0.141)	0.404(±0.033)	0.926(±0.115)
RIGAElit	0.771(±0.089)	0.526(±0.046)	0.913(±0.086)
HyperMelit	0.619(±0.105)	0.46(±0.023)	0.923(±0.053)
GA+Repair	0.211(±0.047)	**0.389(±0.05)**	0.511(±0.126)
DDECV+Repair	**0.169(±0.03)**	0.48(±0.039)	**0.349(±0.021)**

Tabla 4.6: Valores promedio y desviación estándar del offline error obtenidos por DDECV + Repair y los algoritmos comparados con una frecuencia del cambio de 500 evaluaciones y una severidad media del cambio (k=0.5 y S=20). Los mejores resultados se remarcan en negritas.

Algoritmos	**Funciones**		
	G24_u	**G24_1**	**G24_f**
GAElit	0.184(±0.035)	0.641(±0.057)	0.175(±0.083)
RIGAElit	0.235(±0.025)	0.496(±0.046)	0.266(±0.051)
HyperMElit	0.163(±0.026)	0.52(±0.065)	0.209(±0.053)
GA+Repair	0.5(±0.059)	0.264(±0.024)	0.077(±0.011)
DDECV+Repair	**0.086(±0.009)**	**0.123(±0.015)**	**0.04(±0.009)**
	G24_uf	**G24_2**	**G24_2u**
GAElit	0.091(±0.022)	0.372(±0.05)	0.132(±0.017)
RIGAElit	0.125(±0.02)	0.325(±0.037)	0.146(±0.024)
HyperMElit	0.091(±0.012)	0.364(±0.043)	0.115(±0.016)
GA+Repair	0.358(±0.018)	0.298(±0.036)	0.354(±0.029)
DDECV+Repair	**0.019(±0.005)**	**0.137(±0.008)**	**0.089(±0.008)**
	G24_3	**G24_3b**	**G24_3f**
GAElit	0.375(±0.049)	0.631(±0.084)	0.252(±0.058)
RIGAElit	0.436(±0.048)	0.545(±0.051)	0.264(±0.048)
HyperMElit	0.404(±0.05)	0.557(±0.088)	0.244(±0.051)
GA+Repair	**0.063(±0.008)**	0.184(±0.019)	0.035(±0.008)
DDECV+Repair	0.064(±0.008)	**0.143(±0.012)**	**0.024(±0.007)**
	G24_4	**G24_5**	**G24_6a**
GAElit	0.646(±0.075)	0.367(±0.029)	1.038(±0.157)
RIGAElit	0.542(±0.047)	0.287(±0.035)	0.534(±0.05)
HyperMElit	0.573(±0.075)	0.324(±0.039)	0.694(±0.071)
GA+Repair	**0.143(±0.015)**	0.196(±0.024)	0.616(±0.074)
DDECV+Repair	0.145(±0.012)	**0.139(±0.014)**	**0.08(±0.016)**
	G24_6b	**G24_6c**	**G24_6d**
GAElit	0.631(±0.057)	0.666(±0.052)	0.664(±0.075)
RIGAElit	0.436(±0.039)	0.443(±0.029)	0.512(±0.057)
HyperMElit	0.535(±0.039)	0.543(±0.051)	0.584(±0.041)
GA+Repair	0.567(±0.048)	0.518(±0.038)	0.475(±0.038)
DDECV+Repair	**0.097(±0.01)**	**0.093(±0.011)**	**0.193(±0.015)**
	G24_7	**G24_8a**	**G24_8b**
GAElit	0.441(±0.053)	0.356(±0.028)	0.807(±0.056)
RIGAElit	0.565(±0.068)	0.405(±0.028)	0.758(±0.064)
HyperMElit	0.43(±0.062)	0.355(±0.028)	0.71(±0.071)
GA+Repair	**0.134(±0.017)**	0.341(±0.032)	0.38(±0.068)
DDECV+Repair	0.135(±0.023)	**0.262(±0.031)**	**0.176(±0.033)**

Tabla 4.7: Valores promedio y desviación estándar del offline error obtenidos por DDECV + Repair y los algoritmos comparados con una frecuencia del cambio de 1000 evaluaciones y una severidad media del cambio (k=0.5 y S=20). Los mejores resultados se remarcan en negritas.

Algoritmos	**Funciones**		
	G24_u	**G24_1**	**G24_f**
GAElit	0.106(±0.035)	0.459(±0.057)	0.154(±0.083)
RIGAElit	0.149(±0.025)	0.346(±0.046)	0.178(±0.051)
HyperMelit	0.111(±0.026)	0.384(±0.065)	0.149(±0.053)
GA+Repair	0.468(±0.059)	0.226(±0.024)	0.041(±0.011)
DE+Repair	0.099(±0.01)	0.151(±0.024)	0.039(±0.022)
GSA+Repair	0.049(±0.004)	0.132(±0.015)	0.029(±0.012)
DDECV+Repair	**0.039(±0.007)**	**0.061(±0.01)**	**0.021(±0.006)**
	G24_uf	**G24_2**	**G24_2u**
GAElit	0.063(±0.022)	0.288(±0.05)	0.073(±0.017)
RIGAElit	0.069(±0.02)	0.246(±0.037)	0.091(±0.024)
HyperMelit	0.053(±0.012)	0.253(±0.043)	0.068(±0.016)
GA+Repair	0.218(±0.018)	0.281(±0.036)	0.294(±0.029)
DE+Repair	0.057(±0.019)	0.191(±0.014)	0.141(±0.012)
GSA+Repair	0.047(±0.009)	0.182(±0.019)	0.196(±0.012)
DDECV+Repair	**0.009(±0.002)**	**0.062(±0.006)**	**0.036(±0.001)**
	G24_3	**G24_3b**	**G24_3f**
GAElit	0.289(±0.049)	0.457(±0.084)	0.158(±0.058)
RIGAElit	0.308(±0.048)	0.386(±0.051)	0.167(±0.048)
HyperMelit	0.243(±0.05)	0.394(±0.088)	0.128(±0.051)
GA+Repair	0.156(±0.008)	0.171(±0.019)	0.025(±0.008)
DE+Repair	0.091(±0.012)	0.121(±0.019)	0.013(±0.009)
GSA+Repair	**0.028(±0.004)**	**0.076(±0.009)**	**0.009(±0.007)**
DDECV+Repair	0.046(±0.006)	0.084(±0.006)	0.01(±0.002)
	G24_4	**G24_5**	**G24_6a**
GAElit	0.453(±0.075)	0.266(±0.029)	0.674(±0.157)
RIGAElit	0.421(±0.047)	0.24(±0.035)	0.333(±0.05)
HyperMelit	0.426(±0.075)	0.248(±0.039)	0.491(±0.071)
GA+Repair	0.211(±0.015)	0.236(±0.024)	0.431(±0.074)
DE+Repair	0.121(±0.021)	0.121(±0.011)	0.047(±0.009)
GSA+Repair	**0.073(±0.012)**	0.153(±0.013)	**0.033(±0.003)**
DDECV+Repair	0.088(±0.011)	**0.078(±0.008)**	0.036(±0.005)
	G24_6b	**G24_6c**	**G24_6d**
GAElit	0.408(±0.057)	0.441(±0.052)	0.51(±0.075)
RIGAElit	0.309(±0.039)	0.325(±0.029)	0.342(±0.057)
HyperMelit	0.39(±0.039)	0.394(±0.051)	0.456(±0.041)
GA+Repair	0.427(±0.048)	0.39(±0.038)	0.354(±0.038)
DE+Repair	0.101(±0.012)	0.79(±0.01)	0.91(±0.011)
GSA+Repair	0.047(±0.003)	0.045(±0.004)	**0.037(±0.007)**
DDECV+Repair	**0.041(±0.01)**	**0.041(±0.01)**	0.079(±0.006)
	G24_7	**G24_8a**	**G24_8b**
GAElit	0.316(±0.053)	0.266(±0.028)	0.662(±0.056)
RIGAElit	0.416(±0.068)	0.304(±0.028)	0.598(±0.064)
HyperMelit	0.315(±0.062)	0.279(±0.028)	0.608(±0.071)
GA+Repair	0.181(±0.017)	0.496(±0.032)	0.391(±0.068)
DE+Repair	0.033(±0.009)	0.217(±0.033)	0.227(±0.039)
GSA+Repair	**0.018(±0.002)**	0.202(±0.041)	0.192(±0.034)
DDECV+Repair	0.107(±0.011)	**0.138(±0.015)**	**0.074(±0.025)**

Tabla 4.8: Valores promedio y desviación estándar del offline error obtenidos por DDECV + Repair y los algoritmos comparados con una frecuencia del cambio de 2000 evaluaciones y una severidad media del cambio (k=0.5 y S=20). Los mejores resultados se remarcan en negritas.

Algoritmos	**Funciones**		
	G24_u	**G24_1**	**G24_f**
GAElit	0.065(±0.011)	0.332(±0.074)	0.092(±0.052)
RIGAElit	0.11(±0.014)	0.235(±0.038)	0.106(±0.037)
HyperMelit	0.072(±0.015)	0.289(±0.053)	0.084(±0.042)
GA+Repair	0.262(±0.04)	0.055(±0.012)	0.023(±0.006)
DDECV+Repair	**0.023(±0.003)**	**0.036(±0.01)**	**0.011(±0.004)**
	G24_uf	**G24_2**	**G24_2u**
GAElit	0.032(±0.01)	0.183(±0.024)	0.049(±0.008)
RIGAElit	0.047(±0.015)	0.168(±0.023)	0.057(±0.011)
HyperMelit	0.028(±0.008)	0.172(±0.037)	0.044(±0.012)
GA+Repair	0.164(±0.054)	0.147(±0.022)	0.171(±0.04)
DDECV+Repair	**0.005(±0.001)**	**0.035(±0.007)**	**0.018(±0.001)**
	G24_3	**G24_3b**	**G24_3f**
GAElit	0.164(±0.033)	0.32(±0.058)	0.072(±0.032)
RIGAElit	0.208(±0.026)	0.262(±0.024)	0.1(±0.026)
HyperMelit	0.168(±0.029)	0.288(±0.048)	0.082(±0.036)
GA+Repair	**0.019(±0.004)**	**0.044(±0.009)**	0.01(±0.003)
DDECV+Repair	0.036(±0.002)	0.063(±0.009)	**0.006(±0.001)**
	G24_4	**G24_5**	**G24_6a**
GAElit	0.333(±0.074)	0.196(±0.026)	0.408(±0.05)
RIGAElit	0.309(±0.037)	0.174(±0.022)	0.236(±0.026)
HyperMelit	0.287(±0.067)	0.182(±0.019)	0.287(±0.036)
GA+Repair	**0.044(±0.009)**	0.111(±0.023)	0.3(±0.054)
DDECV+Repair	0.057(±0.005)	**0.041(±0.004)**	**0.02(±0.004)**
	G24_6b	**G24_6c**	**G24_6d**
GAElit	0.274(±0.028)	0.282(±0.033)	0.318(±0.059)
RIGAElit	0.21(±0.025)	0.213(±0.027)	0.242(±0.027)
HyperMelit	0.234(±0.019)	0.249(±0.034)	0.281(±0.03)
GA+Repair	0.306(±0.03)	0.287(±0.042)	0.263(±0.024)
DDECV+Repair	**0.026(±0.005)**	**0.022(±0.004)**	**0.044(±0.003)**
	G24_7	**G24_8a**	**G24_8b**
GAElit	0.217(±0.047)	0.232(±0.023)	0.499(±0.048)
RIGAElit	0.303(±0.043)	0.269(±0.017)	0.496(±0.042)
HyperMelit	0.253(±0.036)	0.237(±0.013)	0.463(±0.052)
GA+Repair	**0.05(±0.015)**	0.247(±0.02)	0.136(±0.035)
DDECV+Repair	0.057(±0.005)	**0.075(±0.015)**	**0.041(±0.012)**

Tabla 4.9: Valores promedio y desviación estándar del offline error obtenidos por DDECV + Repair y los algoritmos comparados con una frecuencia del cambio de 4000 evaluaciones y una severidad media del cambio (k=0.5 y S=20). Los mejores resultados se remarcan en negritas.

Algoritmos	**Funciones**		
	G24_u	**G24_1**	**G24_f**
GAElit	0.038(±0.009)	0.192(±0.052)	0.069(±0.036)
RIGAElit	0.061(±0.013)	0.155(±0.016)	0.073(±0.034)
HyperMElit	0.046(±0.01)	0.195(±0.025)	0.066(±0.044)
GA+Repair	0.332(±0.017)	0.032(±0.008)	0.012(±0.003)
DDECV+Repair	**0.01(±0.001)**	**0.019(±0.005)**	**0.005(±0.002)**
	G24_uf	**G24_2**	**G24_2u**
GAElit	0.016(±0.003)	0.141(±0.021)	0.028(±0.006)
RIGAElit	0.03(±0.007)	0.12(±0.013)	0.036(±0.01)
HyperMElit	0.015(±0.005)	0.138(±0.034)	0.025(±0.008)
GA+Repair	0.117(±0.025)	0.096(±0.01)	0.114(±0.029)
DDECV+Repair	**0.002(±0.001)**	**0.021(±0.005)**	**0.01(±0.001)**
	G24_3	**G24_3b**	**G24_3f**
GAElit	0.119(±0.034)	0.2(±0.05)	0.057(±0.016)
RIGAElit	0.148(±0.019)	0.188(±0.026)	0.054(±0.02)
HyperMElit	0.104(±0.019)	0.192(±0.026)	0.052(±0.02)
GA+Repair	**0.01(±0.002)**	**0.024(±0.004)**	0.005(±0.001)
DDECV+Repair	0.029(±0.003)	0.038(±0.006)	**0.003(±0.001)**
	G24_4	**G24_5**	**G24_6a**
GAElit	0.193(±0.029)	0.141(±0.018)	0.226(±0.04)
RIGAElit	0.218(±0.039)	0.132(±0.017)	0.148(±0.016)
HyperMElit	0.209(±0.034)	0.144(±0.024)	0.169(±0.022)
GA+Repair	**0.025(±0.004)**	0.086(±0.017)	0.217(±0.028)
DDECV+Repair	0.037(±0.006)	**0.024(±0.003)**	**0.01(±0.002)**
	G24_6b	**G24_6c**	**G24_6d**
GAElit	0.166(±0.02)	0.164(±0.02)	0.183(±0.024)
RIGAElit	0.141(±0.019)	0.143(±0.019)	0.152(±0.019)
HyperMElit	0.151(±0.017)	0.151(±0.017)	0.17(±0.023)
GA+Repair	0.205(±0.025)	0.212(±0.04)	0.191(±0.02)
DDECV+Repair	**0.013(±0.003)**	**0.012(±0.002)**	**0.024(±0.002)**
	G24_7	**G24_8a**	**G24_8b**
GAElit	0.148(±0.03)	0.205(±0.011)	0.344(±0.04)
RIGAElit	0.21(±0.04)	0.232(±0.01)	0.363(±0.042)
HyperMElit	0.158(±0.037)	0.215(±0.013)	0.332(±0.034)
GA+Repair	**0.026(±0.007)**	0.217(±0.014)	0.068(±0.019)
DDECV+Repair	0.034(±0.005)	**0.041(±0.014)**	**0.018(±0.006)**

En la Tabla 4.10 se puede observar que DDECV + Repair supera a todos los algoritmos en todas las frecuencias del cambio, con la excepción de GA + Repair con 250 evaluaciones. Los resultados de la prueba post-hoc de Bonferroni Dunn para 250 y 500 evaluaciones se muestran en la Figura 4.1 y para 2000 y 4000 evaluaciones en la Figura 4.2. En ambos casos se confirman los resultados encontrados. Con respecto a la frecuencia del cambio de 1000 evaluaciones (Tabla 4.11), DDECV + Repair supera a GAElit, RIGAElit, HyperMElit y GA + Repair. Por otra parte, el desempeño de DDECV + Repair tienen un desempeño similar a DE + Repair y GSA + Repair. En la Figura 4.3, con la prueba post-hoc de Bonferroni-Dunn aplicada a los resultados, confirma lo mencionado previamente.

La segunda comparación de este experimento tiene el objetivo de analizar el impacto de la severidad del cambio en DDECV + Repair. Los resultados fueron validados con la prueba no paramétrica de Kruskal-Wallis (KW) con un 95% de confianza aplicada a los resultados de offline error obtenidos por DDECV + Repair y los algoritmos comparados (GAElit, RIGAElit, HyperMElit, y GA+Repair) con severidad baja en el cambio (k=0.25 y S=10), y alta (k=1.0 y S=50), ambos con 1000 evaluaciones como frecuencia del cambio. Los resultados se muestran en la Tabla 4.14. DE + Repair y GSA + Repair fueron omitidos en estos resultados, porque no se encontraros sus resultados. Los detalles completos de estas comparaciones con sus respectivos valores de offline error están en las Tablas 4.12 y 4.13, para severidad del cambio baja y alta respectivamente.

Basado en los resultados de la Tabla 4.14, DDECV + Repair supera a los algoritmos comparados en ambas severidades del cambio. La única excepción fue GA + Repair cuando la severidad del cambio es baja, donde, ambos algoritmos tienen un desempeño similar. Los resultados de la prueba post-hoc Bonferroni-Dunn se muestran en las Figuras 4.4a y 4.4b, y confirman lo antes discutido.

El segundo experimento proporcionó los siguientes resultados a la investigación:

- DDECV + Repair fue robusto en diferentes frecuencias del cambio. Sin embargo, cuando la frecuencia del cambio es media (i.e., 1000 evaluaciones), sólo se obtuvieron resultados comparables contra los dos algoritmos con métodos de reparación (DE + Repair y GSA + Repair).

- DDECV + Repair no fue sensible a la severidad baja o alta del cambio presentada tanto en la función objetivo como en las restricciones de un DCOP.

Experimento 3. Detección del cambio, recuperación y análisis de la diversidad

El tercer experimento estudia la habilidad de DDECV + Repair para detectar los cambios en la función objetivo y/o restricciones y su capacidad de recuperación después de ellos. Además, se analiza su manejo de diversidad (balance entre soluciones factibles y no factibles en la población).

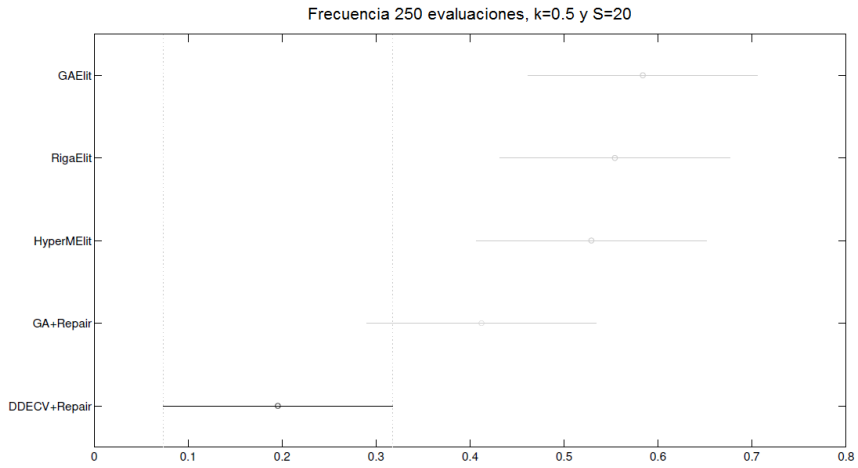
En la Tabla 4.15 se presenta el promedio y la desviación estándar del número de cambios detectados de manera exitosa en cada problema dinámico de prueba en 50 ejecuciones independientes por DDECV + Repair e HyperMElit (los problemas de prueba estáticos no fueron considerados en este experimento).

Tabla 4.10: Resultados de la prueba de Kruskal-Wallis con un 95% de confianza sobre los valores de offline error en las tablas 4.5, 4.6, 4.8 y 4.9, obtenidos por DDECV+Repair, GAElit, RIGAElit, HyperMElit y GA+Repair con frecuencias del cambio diferentes y severidad media del cambio (k=0.5 y S=20). "$X^{(+)}$" significa que el algoritmo en la columna correspondiente, superó al algoritmo X. "$X^{(-)}$" significa que el algoritmo en la columna correspondiente, fue superado por el algoritmo X. Si el algoritmo X no aparece en la columna Y, no existe diferencia significativa entre X y Y.

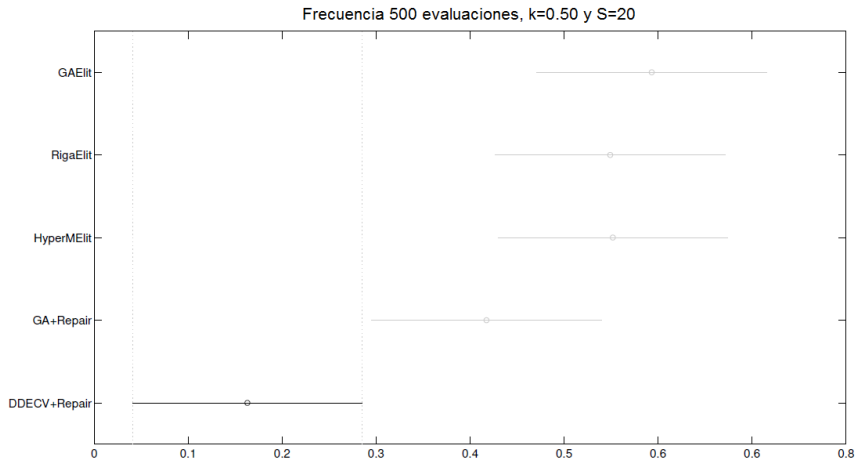
Frecuencia	**Algoritmos**				
	GAElit(1)	**RIGAElit(2)**	**HyperMElit(3)**	**GA+Repair(4)**	**DDECV+Repair(5)**
250	$5^{(-)}$	$5^{(-)}$	$5^{(-)}$		$1^{(+)}$, $2^{(+)}$ y $3^{(+)}$
500	$5^{(-)}$	$5^{(-)}$	$5^{(-)}$	$5^{(-)}$	$1^{(+)}$, $2^{(+)}$, $3^{(+)}$ y $4^{(+)}$
2000	$5^{(-)}$	$5^{(-)}$	$5^{(-)}$	$5^{(-)}$	$1^{(+)}$, $2^{(+)}$, $3^{(+)}$ y $4^{(+)}$
4000	$5^{(-)}$	$5^{(-)}$	$5^{(-)}$	$5^{(-)}$	$1^{(+)}$, $2^{(+)}$, $3^{(+)}$ y $4^{(+)}$

Tabla 4.11: Resultados de la prueba de Kruskal-Wallis con un 95% de confianza sobre los valores de offline error en la tabla 4.7, obtenidos por DDECV+Repair, GAElit, RIGAElit, HyperMElit, GA+Repair DE + Repair y GSA + Repair con frecuencia del cambio de 1000 evaluaciones y severidad media del cambio (k=0.5 y S=20). "$X^{(+)}$" significa que el algoritmo en la columna correspondiente, superó al algoritmo X. "$X^{(-)}$" significa que el algoritmo en la columna correspondiente, fue superado por el algoritmo X. Si el algoritmo X no aparece en la columna Y, no existe diferencia significativa entre X y Y.

GAElit (1)	**RIGAElit (2)**	**HyperMElit (3)**	**GA + Repair (4)**	**DE + Repair (5)**	**GSA + Repair (6)**	**DDECV + Repair(7)**
$6^{(-)}$ y $7^{(-)}$	$6^{(-)}$ y $7^{(-)}$	$6^{(-)}$ y $7^{(-)}$	$6^{(-)}$ y $7^{(-)}$		$1^{(+)}$, $2^{(+)}$, $3^{(+)}$, $4^{(+)}$	$1^{(+)}$, $2^{(+)}$, $3^{(+)}$, $4^{(+)}$

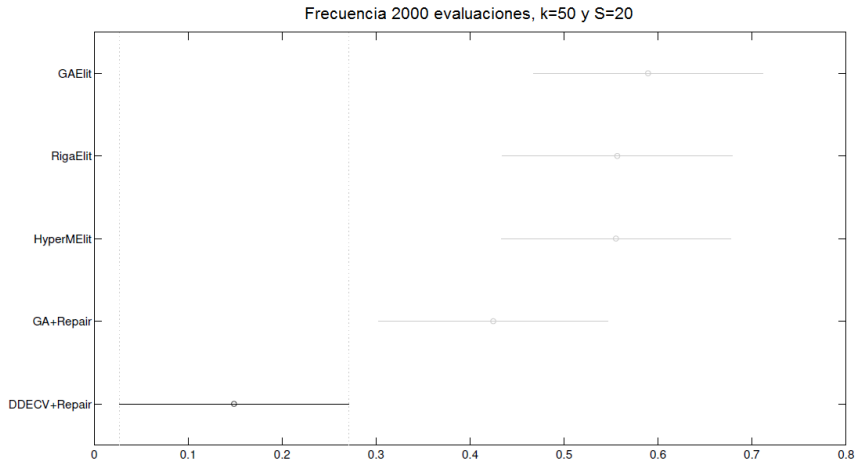


(a)

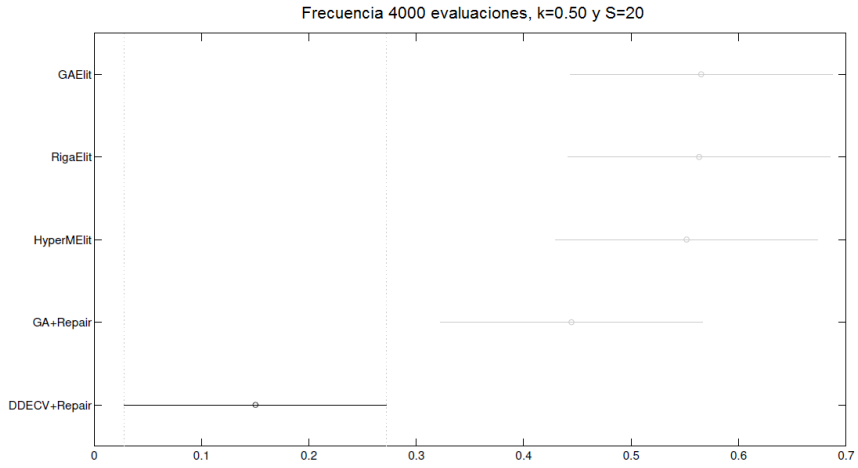


(b)

Figura 4.1: Resultados de la prueba post-hoc Bonferroni-Dunn basados en los valores de offline error obtenidos por DDECV + Repair y los algoritmos comparados con severidad media del cambio (k=0. 5 y S=20), considerando dos frecuencias de cambio: (a) 250 evaluaciones y (b) 500 evaluaciones.



(a)



(b)

Figura 4.2: Resultados de la prueba post-hoc Bonferroni-Dunn basados en los valores de offline error obtenidos por DDECV + Repair y los algoritmos comparados con severidad media del cambio (k=0.5 y S=20), considerando dos frecuencias de cambio: (a) 2000 evaluaciones y (b) 4000 evaluaciones.

Tabla 4.12: Valores promedio y desviación estándar del offline error obtenidos por DDECV + Repair y los algoritmos comparados con una frecuencia del cambio de 1000 evaluaciones y una severidad baja del cambio (k=0.25 y S=10). Los mejores resultados se remarcan en negritas.

Algoritmos	**Funciones**		
	G24_u	**G24_1**	**G24_f**
GAElit	0.101(±0.021)	0.319(±0.06)	0.113(±0.06)
RIGAElit	0.155(±0.018)	0.246(±0.043)	0.119(±0.039)
HyperMelit	0.094(±0.016)	0.273(±0.047)	0.119(±0.05)
GA+Repair	0.348(±0.055)	0.067(±0.011)	0.034(±0.009)
DDECV+Repair	**0.027(±0.005)**	**0.052(±0.013)**	**0.023(±0.006)**
	G24_uf	**G24_2**	**G24_2u**
GAElit	0.035(±0.011)	0.157(±0.03)	0.05(±0.013)
RIGAElit	0.049(±0.011)	0.152(±0.025)	0.073(±0.013)
HyperMelit	0.033(±0.012)	0.151(±0.029)	0.051(±0.013)
GA+Repair	0.157(±0.033)	0.141(±0.023)	0.237(±0.032)
DDECV+Repair	**0.01(±0.004)**	**0.047(±0.006)**	**0.022(±0.001)**
	G24_3	**G24_3b**	**G24_3f**
GAElit	0.18(±0.047)	0.313(±0.038)	0.101(±0.05)
RIGAElit	0.212(±0.033)	0.301(±0.041)	0.115(±0.032)
HyperMelit	0.179(±0.037)	0.303(±0.059)	0.095(±0.029)
GA+Repair	**0.015(±0.003)**	**0.038(±0.008)**	0.018(±0.004)
DDECV+Repair	0.057(±0.005)	0.076(±0.012)	**0.011(±0.003)**
	G24_4	**G24_5**	**G24_6a**
GAElit	0.367(±0.06)	0.192(±0.033)	0.652(±0.104)
RIGAElit	0.344(±0.025)	0.191(±0.023)	0.358(±0.029)
HyperMelit	0.33(±0.051)	0.19(±0.018)	0.452(±0.047)
GA+Repair	**0.062(±0.009)**	0.131(±0.024)	0.442(±0.064)
DDECV+Repair	0.077(±0.012)	**0.075(±0.012)**	**0.035(±0.006)**
	G24_6b	**G24_6c**	**G24_6d**
GAElit	0.427(±0.035)	0.435(±0.031)	0.504(±0.052)
RIGAElit	0.325(±0.025)	0.326(±0.021)	0.344(±0.031)
HyperMelit	0.374(±0.024)	0.392(±0.039)	0.427(±0.045)
GA+Repair	0.428(±0.047)	0.409(±0.044)	0.366(±0.039)
DDECV+Repair	**0.043(±0.01)**	**0.038(±0.007)**	**0.083(±0.011)**
	G24_7	**G24_8a**	**G24_8b**
GAElit	0.259(±0.042)	**0.164(±0.021)**	0.427(±0.063)
RIGAElit	0.357(±0.048)	0.271(±0.023)	0.426(±0.058)
HyperMelit	0.27(±0.049)	0.171(±0.018)	0.388(±0.05)
GA+Repair	**0.063(±0.015)**	0.191(±0.031)	0.113(±0.027)
DDECV+Repair	0.362(±0.083)	0.2(±0.02)	**0.07(±0.015)**

Tabla 4.13: Valores promedio y desviación estándar del offline error obtenidos por DDECV + Repair y los algoritmos comparados con una frecuencia del cambio de 1000 evaluaciones y una severidad alta del cambio (k=1.0 y S=50). Los mejores resultados se remarcan en negritas.

Algoritmos	**Funciones**		
	G24_u	**G24_1**	**G24_f**
GAElit	0.106(±0.02)	0.574(±0.114)	0.265(±0.146)
RIGAElit	0.142(±0.031)	0.481(±0.044)	0.231(±0.06)
HyperMelit	0.111(±0.031)	0.502(±0.069)	0.221(±0.094)
GA+Repair	0.375(±0.068)	0.222(±0.045)	0.055(±0.022)
DDECV+Repair	**0.082(±0.013)**	**0.086(±0.012)**	**0.022(±0.007)**
	G24_uf	**G24_2**	**G24_2u**
GAElit	0.093(±0.039)	0.474(±0.094)	0.186(±0.048)
RIGAElit	0.101(±0.032)	0.392(±0.06)	0.266(±0.046)
HyperMelit	0.09(±0.032)	0.428(±0.077)	0.187(±0.058)
GA+Repair	0.309(±0.095)	0.441(±0.068)	0.633(±0.1)
DDECV+Repair	**0.009(±0.004)**	**0.056(±0.006)**	**0.037(±0.001)**
	G24_3	**G24_3b**	**G24_3f**
GAElit	0.324(±0.088)	0.615(±0.114)	0.201(±0.099)
RIGAElit	0.413(±0.095)	0.482(±0.07)	0.237(±0.06)
HyperMelit	0.318(±0.071)	0.54(±0.097)	0.227(±0.06)
GA+Repair	0.056(±0.012)	0.17(±0.041)	0.025(±0.011)
DDECV+Repair	**0.026(±0.003)**	**0.096(±0.008)**	**0.011(±0.003)**
	G24_4	**G24_5**	**G24_6a**
GAElit	0.62(±0.06)	0.673(±0.079)	0.719(±0.183)
RIGAElit	0.528(±0.098)	0.592(±0.062)	0.353(±0.052)
HyperMelit	0.55(±0.098)	0.607(±0.047)	0.44(±0.074)
GA+Repair	0.102(±0.046)	0.408(±0.048)	0.452(±0.093)
DDECV+Repair	**0.096(±0.007)**	**0.087(±0.01)**	**0.033(±0.007)**
	G24_6b	**G24_6c**	**G24_6d**
GAElit	0.411(±0.047)	0.491(±0.066)	0.495(±0.076)
RIGAElit	0.284(±0.036)	0.307(±0.034)	0.369(±0.082)
HyperMelit	0.347(±0.041)	0.357(±0.046)	0.426(±0.056)
GA+Repair	0.432(±0.071)	0.438(±0.059)	0.387(±0.083)
DDECV+Repair	**0.044(±0.01)**	**0.04(±0.008)**	**0.082(±0.008)**
	G24_7	**G24_8a**	**G24_8b**
GAElit	0.378(±0.109)	0.364(±0.037)	1.069(±0.108)
RIGAElit	0.487(±0.109)	0.339(±0.034)	0.938(±0.106)
HyperMelit	0.387(±0.06)	0.383(±0.039)	0.94(±0.135)
GA+Repair	**0.09(±0.025)**	0.384(±0.047)	0.511(±0.069)
DDECV+Repair	0.106(±0.015)	**0.075(±0.013)**	**0.111(±0.016)**

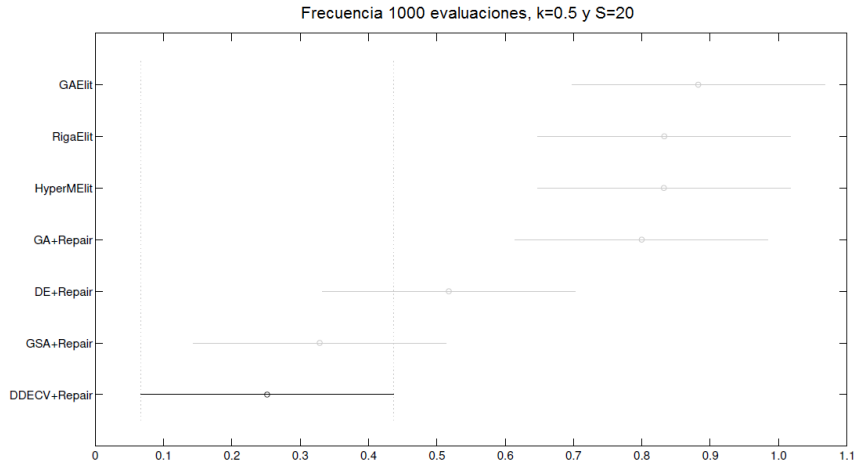
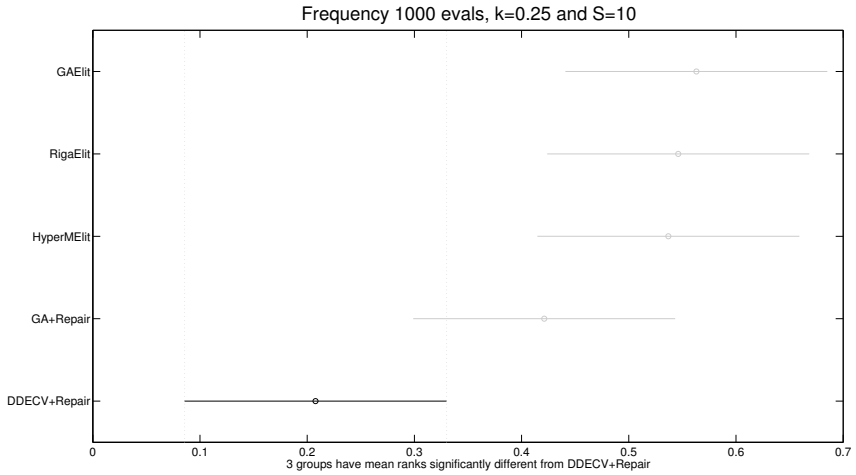


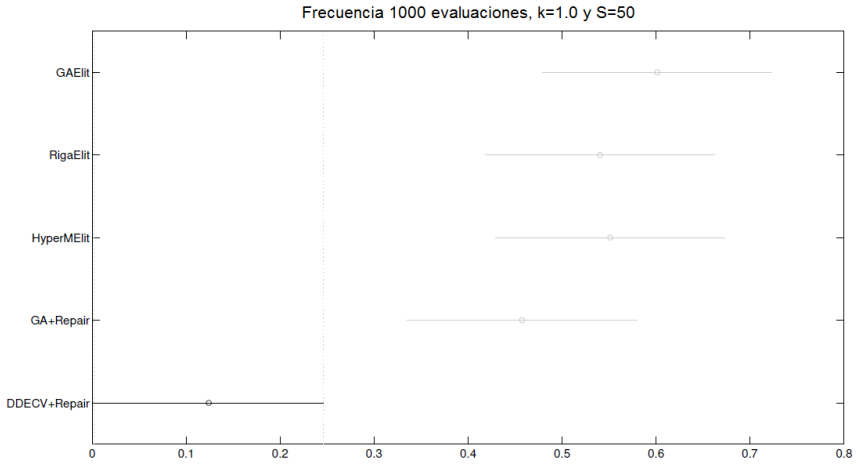
Figura 4.3: Resultados de la prueba post-hoc Bonferroni-Dunn basados en los valores de offline error obtenidos por DDECV + Repair y los algoritmos comparados considerando 1000 evaluaciones como frecuencia del cambio y severidad media del cambio (k=0.5 y S=20).

Tabla 4.14: Resultados de la prueba de Kruskal-Wallis con un 95% de confianza sobre los valores de offline error en las tablas 4.12 y 4.13, obtenidos por DDECV + Repair, GAElit, RIGAElit, HyperMElit y GA + Repair con severidad baja (k=0.25 y S=10), y alta (k=1.0 y S=50) del cambio, ambos con 1000 evaluaciones como frecuencia del cambio. "$X^{(+)}$" significa que el algoritmo en la columna correspondiente, superó al algoritmo X. "$X^{(-)}$" significa que el algoritmo en la columna correspondiente, fue superado por el algoritmo X. Si el algoritmo X no aparece en la columna Y, no existe diferencia significativa entre X y Y.

Severidad del cambio		**Algoritmos**				
k	**S**	**GAElit(1)**	**RIGAElit(2)**	**HyperMElit(3)**	**GA + Repair(4)**	**DDECV + Repair(5)**
0.25	10	$5^{(-)}$	$5^{(-)}$	$5^{(-)}$		$1^{(+)}$, $2^{(+)}$ y $3^{(+)}$
1.00	50	$5^{(-)}$	$5^{(-)}$	$5^{(-)}$	$5^{(-)}$	$1^{(+)}$, $2^{(+)}$, $3^{(+)}$ y $4^{(+)}$



(a)



(b)

Figura 4.4: Resultados de la prueba post-hoc Bonferroni-Dunn basados en los valores de offline error obtenidos por DDECV + Repair y los algoritmos comparados considerando 1000 evaluaciones como frecuencia del cambio y niveles de severidad: (a) bajo (k=0.25 y S=10) y (b) alto (k=1.0 y S=50).

Debido a que entre los algoritmos comparados, HyperM es el único algoritmo que usa un mecanismos de detección del cambio diferente (disminución del valor de la función objetivo [2]) fue seleccionado para esta comparación. La frecuencia del cambio está dada cada 1000 evaluaciones y la severidad del cambio fue media (k=0.5 y S=20).

El número total de cambios en una sola ejecución es de 10, los resultados sugieren que la re-evaluación de soluciones usada por DDECV + Repair es más efectiva con respecto a la de la disminución del valor de la función objetivo cuando se resuelven DCOPs (p. ej., en el problema de prueba G24_3, HyperMElit fue incapaz de detectar cambios en las restricciones dinámicas debido a que la función objetivo es estática). Además, el óptimo cambia entre regiones desconectadas. Este desempeño fue observado con otras frecuencias y severidades del cambio.

Después de obtener un mayor conocimiento de la efectividad de DDECV + Repair para detectar cambios, las Figuras 4.5a, 4.5b, 4.5c, 4.6a y 4.6b presentan la tasa de recuperación (RR) y la tasas absoluta de recuperación (ARR) en diagramas para los siguientes algoritmos: GAElit, RIGAElit, HyperMElit, GA + Repair y DDECV + Repair, todos ellos con cinco frecuencias del cambio (250, 500, 1000, 2000 y 4000 evaluaciones) y una severidad baja del cambio (k=0.5 y S=20). Estos resultados indican que DDECV + Repair se recupera más rápido que los algoritmos comparados, independientemente de la frecuencia del cambio. El mismo comportamiento fue observado en las Figuras 4.7a y 4.7b, con una frecuencia del cambio de 1000 evaluaciones y una severidad baja del cambio (k=0.25 y S=10) y una severidad alta del cambio (k=1.0 y S=50), fueron consideradas respectivamente. Se observó que la habilidad de recuperación mostrada por DDECV + Repair, así como por los algoritmos comparados, parece ser más afectada cuando ocurren cambios más rápidos(p.ej., 250 evaluaciones Figura 4.5a) que con una severidad grande en el cambio (p.ej., k=1.0 y S=50 en Figura 4.7b).

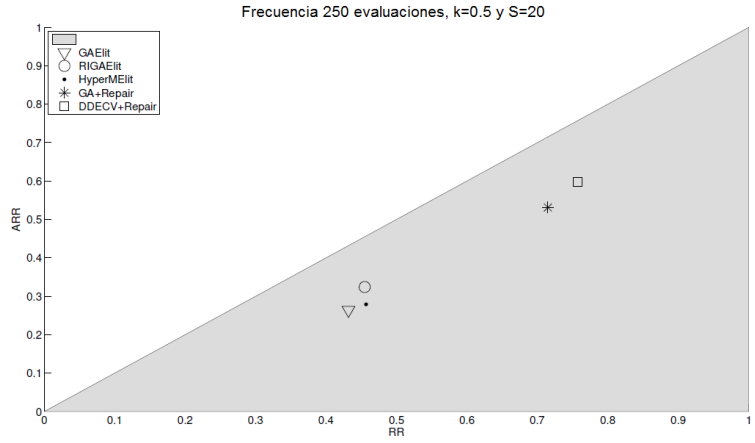
Finalmente para DDECV + Repair y los algoritmos comparados, el manejo de la diversidad, en el contexto de un espacio de búsqueda restringido (p.ej., el porcentaje de soluciones factibles y no factibles en la población) fue evaluado mediante el cálculo del porcentaje de soluciones no factibles en la población. Los promedios de las soluciones no factibles en la población considerando todo el conjunto de problemas de prueba con cinco diferentes frecuencias del cambio (250, 500, 1000, 2000, y 4000 evaluaciones) y una severidad media del cambio (k=0.5 y S=20) son mostrados en la Tabla 4.16. La misma información es presentada en la Tabla 4.17, con 1000 evaluaciones, pero ahora considerando una severidad baja (k=0.25 y S=10) y alta (k=1.0 y S=50) del cambio.

En ambas tablas, independientemente de la frecuencia y severidad del cambio, se observa el mismo comportamiento, donde los dos algoritmos con mecanismo de reparación mantienen porcentajes más bajos de soluciones no factibles que los otros enfoques sin este mecanismo. Además, es importante destacar que de entre los dos algoritmos con mecanismo de reparación (GA + Repair y DDECV + Repair), este último mantiene una mayor proporción de soluciones no factibles (aproximandamente entre 8% y 14%).

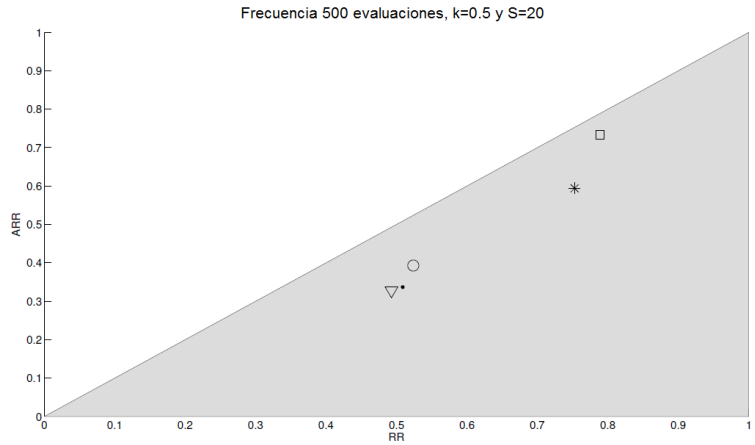
[2]El conjunto de funciones de prueba que se está resolviendo en este trabajo es de minimización, en este caso no se contempla la disminución del valor de valor de la función objetivo, por el contrario el aumento de la misma

Tabla 4.15: Promedio del número de cambios detectados por DDECV + Repair e HyperMElit con una frecuencia del cambio de 1000 evaluaciones y una severidad media del cambio (k=0.5 y S=20). Los problemas de prueba estáticos fueron descartados en esta comparación. El número total de cambios es de 10 para todo el conjunto de problemas de prueba.

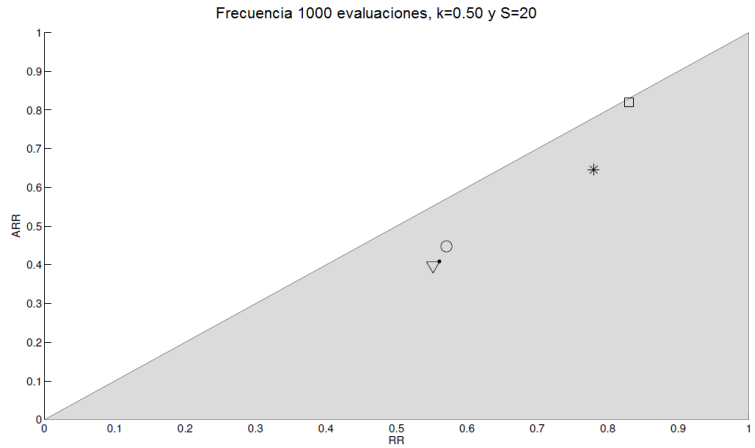
Funciones	**Cambios detectados**	
	HyperM	**DDECV + Repair**
g24_u	6.0 (±0.0)	10.0 (±0.0)
g24_1	6.0 (±0.0)	10.0 (±0.0)
g24_2	4.7 (±0.8)	10.0 (±0.0)
g24_2u	4.5 (±0.5)	10.0 (±0.0)
g24_3	0	10.0 (±0.0)
g24_3b	6.0 (±0.0)	10.0 (±0.0)
g24_4	5.9 (±0.3)	10.0 (±0.0)
g24_5	6.5 (±0.5)	10.0 (±0.0)
g24_6a	5.0 (±0.0)	10.0 (±0.0)
g24_6b	5.0 (±0.0)	10.0 (±0.0)
g24_6c	5.0 (±0.0)	10.0 (±0.0)
g24_6d	5.0 (±0.0)	10.0 (±0.0)
g24_7	9.3 (±0.7)	10.0 (±0.0)
g24_8a	9.0 (±0.9)	10.0 (±0.0)
g24_8b	7.3(±0.9)	10.0 (±0.0)



(a)

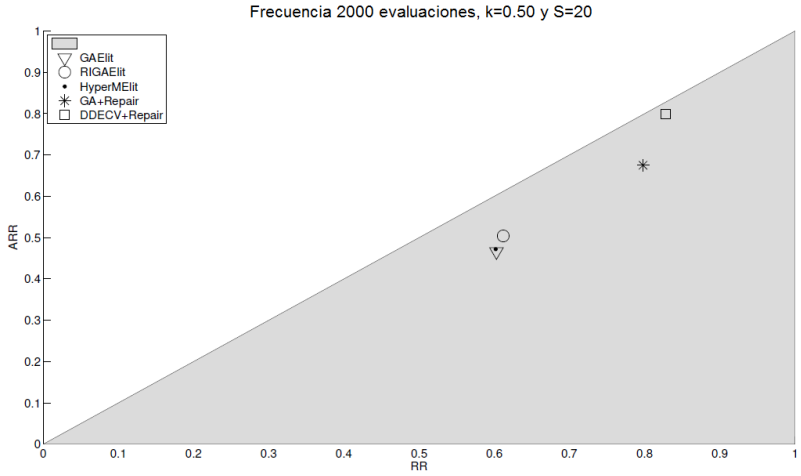


(b)

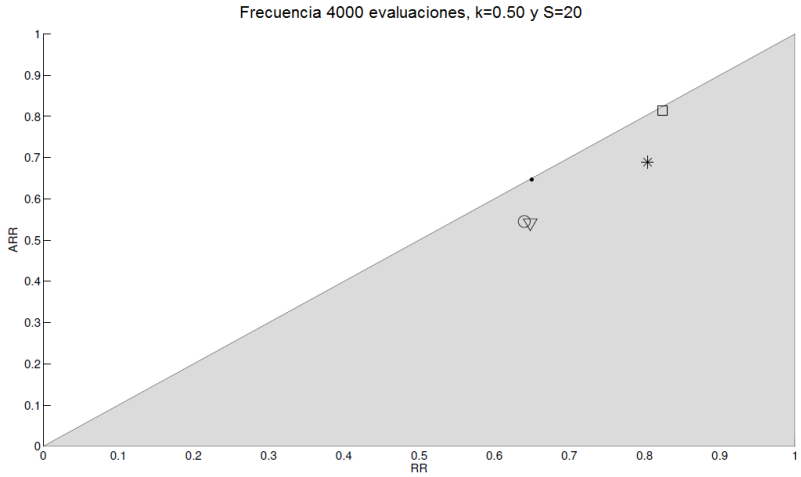


(c)

Figura 4.5: Mapeo de los valores de RR/ARR de GAElit, RIGAElit, HyperMElit, GA + Repair y DDECV + Repair. El diagrama RR-ARR para tres frecuencias del cambio: (a) 250 evaluaciones, (b) 500 evaluaciones, y (c) 1000 evaluaciones. Si un punto se encuentra en la línea diagonal el algoritmo ha sido capaz de recuperarse del cambio y también converge al nuevo óptimo global. El diagrama RR-ARR fue propuesto en [64].

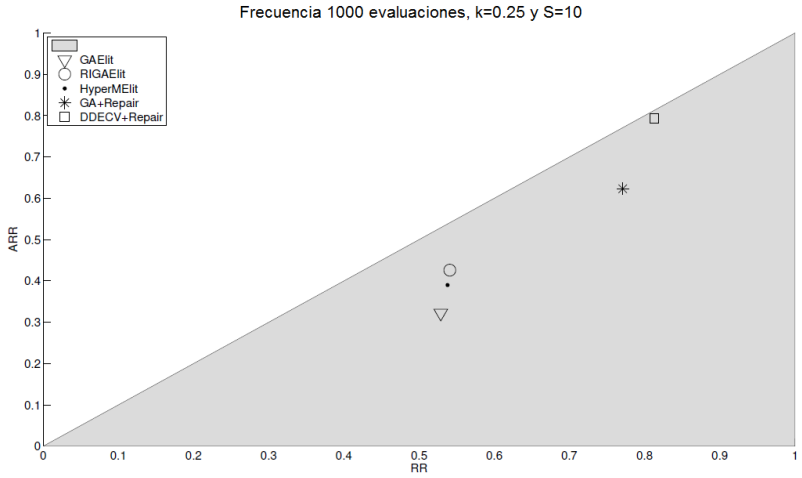


(a)

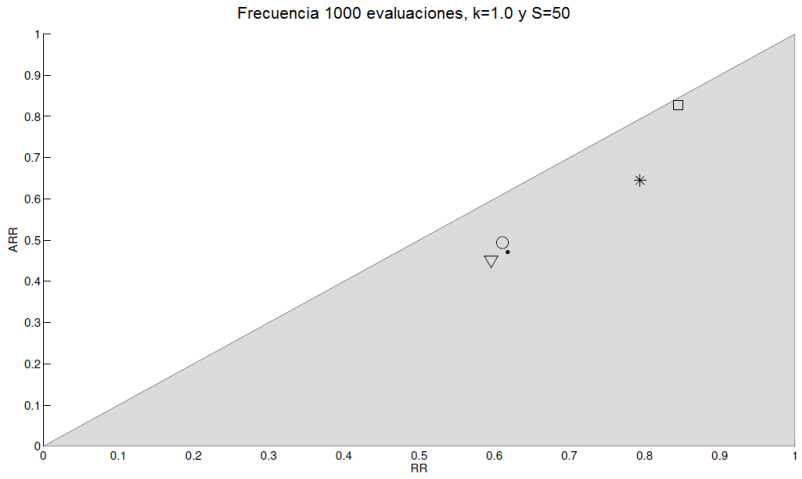


(b)

Figura 4.6: Mapeo de los valores de RR/ARR de GAElit, RIGAElit, HyperMElit, GA + Repair y DDECV + Repair. El diagrama RR-ARR para dos frecuencias del cambio: (a) 2000 evaluaciones y (b) 4000 evaluaciones. Si un punto se encuentra cerca del lado derecho del gráfico, indica una recuperación rápida. Por otra parte, si un punto se encuentra en la línea diagonal, el algoritmo ha sido capaz de recuperarse de un cambio y también converge hacia el nuevo óptimo global. El diagrama RR-ARR fue propuesto en [64].



(a)



(b)

Figura 4.7: Mapeo de los valores de RR/ARR de GAElit, RIGAElit, HyperMElit, GA + Repair y DDECV + Repair con una frecuencia del cambio de 1000 evaluaciones y : (a) severidad baja del cambio (k=0.25 y S=10) y (b) severidad alta del cambio (k=1.0 y S=50). Si un punto está cerca del lado derecho del gráfico, indica una recuperación rápida. Por otra parte, si un punto se encuentra en la línea diagonal, el algoritmo ha sido capaz de recuperarse de un cambio y también converge hacia el nuevo óptimo global. El diagrama RR-ARR fue propuesto en [64].

Tabla 4.16: Promedio del porcentaje de individuos no factibles seleccionados con frecuencia del cambio de 250, 500, 1000, 2000 y 4000 evaluaciones. La severidad media del cambio (k=0.5 y S=20). Sólo los problemas restringidos son incluidos en el cálculo de esta medida.

Frequency	**Algorithms**				
	GAElit	**RIGAElit**	**HyperMElit**	**GA+Repair**	**DDECV+Repair**
250	53.7%	39.7%	54.5%	5.0%	13.2%
500	52.4%	39.8%	53.1%	2.0%	12.9%
1000	52.3%	39.5%	52.7%	1.0%	10.7%
2000	51.9%	39.5%	52.2%	0.0	9.5%
4000	52.3%	39.4%	52.2%	0.0	8.8%

Tabla 4.17: Promedio del porcentaje de individuos no factibles seleccionados con frecuencia del cambio cada 1000 evaluaciones y con severidad baja y alta del cambio (k=0.25, 1.0 y S=20, 50). Sólo los problemas restringidos son incluidos en el cálculo de esta medida.

Severidad del cambio		**Algoritmos**				
k	**S**	**GAElit**	**RIGAElit**	**HyperMElit**	**GA+Repair**	**DDECV+Repair**
0.25	10	50.8%	40.9%	52.8%	3.6%	13.5%
1.00	50	53.4%	39.6%	53.7%	3.6%	10.9%

Dicha proporción, considerando los resultados globales obtenidos en los comparativos por DDECV + Repair, parece ser adecuado para un algoritmo que trata de resolver DCOPs. De hecho, este hallazgo está de acuerdo con la importancia de mantener algunas soluciones factibles para favorecer el éxito en una búsqueda restringida [53].

De este tercer experimento se puede resumir lo siguiente:

- El mecanismo de detección del cambio en DDECV + Repair basado en re-evaluación de soluciones fue particularmente efectivo en el conjunto de problemas dinámicos de prueba resueltos en esta tesis doctoral.
- DDECV + Repair tienen una recuperación rápida después del cambio. Además, se constató que cambios rápidos afectan más la habilidad de recuperación del algoritmo.
- De manera similar a la optimización estática con restricciones, se debe de mantener una baja proporción de soluciones no factibles para obtener resultados competitivos al resolver DCOPs.

Experimento 4. Análisis de dinamismo en diferente partes del problema y mejor error antes de un cambio

El cuarto experimento estudia el desempeño de DDECV + Repair al hacer frente al dinamismo en diferentes partes del problema. Los algoritmos ICHEA y DCTC descritos previamente fueron usados para comparación contra el algoritmo propuesto y sus resultados fueron tomados de sus respectivos documentos [88, 7].

Doce de los dieciocho problemas de prueba fueron resueltos como en el experimento 1. Debido a que las 6 seis funciones son estáticas no fueron utilizadas, además ICHEA y DCTC sólo reportan resultados de doce funciones.

La primera comparación de este cuarto experimento fue realizado entre DDECV + Repair e ICHEA[87] y la medida Mejor_error_antes_de_un_cambio fue calculada. Sólo son consideradas 4 funciones debido a que ICHEA sólo reporta resultados para cuatro funciones [87]. La frecuencia del cambio fue cada 1000 evaluaciones y la severidad del cambio fue media (p.ej., k=0.50 y S=20). Los resultados obtenidos están en la Tabla 4.18. La validación fue hecha con la prueba no paramétrica por pares de la suma de rangos Wilcoxon con un 95% de confianza. Esta prueba indica que no existe diferencia significativa entre DDECV + Repair e ICHEA considerando los cuatro problemas (g24_u, g24_1, g24_3 y g24_4). Por lo tanto, DDECV + Repair tiene un desempeño similar con respecto a ICHEA para alcanzar la solución óptima antes de que ocurra un cambio.

La segunda comparación de este cuarto experimento estudia los efectos del dinamismo en diferentes partes del problema con diferentes frecuencias del cambio y también diferentes severidades del cambio en el desempeño proporcionado por DDECV + Repair y DCTC [7]. La medida usada en estas comparaciones fue Offline error factible. Las frecuencias del cambio fueron 250, 500 y 1000 evaluaciones y diferentes severidades en el cambio (k=0.25, 0.5 y 1.0 y S=10, 20 y 50). La validación estadística se realizó con la prueba no paramétrica por pares de la suma de rangos Wilcoxon con un 95% de confianza. Estas comparaciones fueron divididas por clase de problemas de prueba de la siguiente manera:

Función objetivo y restricciones dinámicas. En la Tabla 4.19 se muestran los valores de la media y la desviación estándar de offline error factible para DDECV + Repair y DCTC en tres problemas de prueba (g24_3b, g24_4 y g24_5) con diferente frecuencias del cambio (250, 500 y 1000 evaluaciones) y diferente severidades del cambio (k=0.25, 0.5 y 1.0 y S=10, 20 y 50). Veintisiete combinaciones fueron llevadas a cabo por cada problema de prueba. La prueba estadística indica que hay diferencias significativas. Basado en estos resultados, DDECV + Repair supera a DCTC en todas las combinaciones de dos problemas de prueba (g24_3b y g24_4). Por otra parte, DDECV + Repair superó en diecinueve combinaciones de un problema de prueba (g24_5), mientras que DCTC fue mejor en ocho combinaciones.

Función objetivo dinámica y restricciones estáticas. En la Tabla 4.20 se incluyen los valores de

la media y desviación estándar de offline error factible para DDECV + Repair y DCTC en tres problemas de prueba (g24_1, g24_2 y g24_8b) con diferente frecuencias del cambio (250, 500 y 1000 evaluaciones) y diferentes severidades del cambio en la función objetivo (k= 0.25,0.50 y 1.0). Nueve combinaciones fueron calculados por cada problema de prueba. La prueba estadística indicó que hay diferencias significativas. De acuerdo a estos resultados, DCTC fue mejor en todas las combinaciones de 2 problemas de prueba (g24_1 y g24_2) y también en sólo ocho combinaciones de un problema de prueba (g24_8b). Por otra parte, DDECV + Repair supera en sólo una combinación cuando la frecuencia del cambio es de 250 evaluaciones y la severidad del cambio en la función objetivo alta (k=1.0). En la Tabla 4.21, se presentan los valores de la media y la desviación estándar de offline error factible para DDECV + Repair y DCTC en tres problemas (g24_6a, g24_6c y g24_6d) con diferentes frecuencias del cambio (250, 500 y 1000 evaluaciones) sin severidad en el cambio. Los resultados sugieren que DDECV + Repair tiene desempeño similar a DCTC, porque no fueron observadas diferencias significativas.

Función objetivo estática y restricciones dinámicas . En la Tabla 4.22 se incluyen los valores del promedio y la desviación estándar de offline error factible para DDECV + Repair y DCTC en dos problemas de prueba (g24_3 y g24_7) con diferentes cambios de frecuencias (250, 500 y 1000 evaluaciones) y diferentes severidades del cambio en las restricciones (S=10, 20 y 50). Nueve combinaciones fueron calculadas por cada problema de prueba. Estos resultados indican que DDECV + Repair tuvo un desempeño similar con respecto a DCTC, porque no fueron observaron diferencias significativas.

El cuarto experimento generó las siguientes conclusiones:

- DDECV + Repair superó a los algoritmos comparados donde el conjunto de problemas tuvo dinamismo en la función objetivo y las restricciones.
- En los problemas de prueba con dinamismo en la función objetivo y restricciones estáticas, DCTC superó a DDECV + Repair.
- Resultados similares fueron obtenidos entre DDECV + Repair y DCTC en problemas de prueba con función objetivo estática y restricciones dinámicas.

Tabla 4.18: Valores promedio y desviación estándar del *mejor_error_antes_de_un_cambio* obtenidos por DDECV + Repair e ICHEA con una frecuencia del cambio de 1000 evaluaciones y una severidad media del cambio (k=0.50 y S=20). Los mejores resultados se remarcan en negritas.

Algoritmos	**Funciones**			
	G24_u	**G24_1**	**G24_3**	**G24_4**
ICHEA	0.0051(±0.004)	0.0333(±0.005)	0.0187(±0.003)	0.0799(±0.006)
DDECV+Repair	**0.000(±0.000)**	**0.012(±0.010)**	**0.004(±0.001)**	**0.008(±0.006)**

Tabla 4.19: Valores promedio y desviación estándar de *offline error factible* obtenidos por DDECV + Repair y DCTC con distintas frecuencias de cambio (250, 500 y 1000 evaluaciones), y distintas severidades de cambio (k=0.25, 0.5 y 1.0 y S=10, 20 y 50), para los problemas de prueba con función objetivo y restricciones dinámicas. Los mejores resultados se remarcan en negritas.

Evals	**Algoritmos**	**Funciones**			**Funciones**			**Funciones**		
		G24_3b	**G24_4**	**G24_5**	**G24_3b**	**G24_4**	**G24_5**	**G24_3b**	**G24_4**	**G24_5**
	k=0.25		*S=10*			*S=20*			*S=50*	
250	DCTC	0.59(±0.15)	0.43(±0.07)	**0.21(±0.02)**	0.54(±0.15)	0.41(±0.05)	0.18(±0.02)	0.56(±0.14)	0.28(±0.04)	**0.11(±0.02)**
	DDECV+Repair	**0.227(±0.042)**	**0.222(±0.04)**	0.235(±0.027)	**0.213(±0.018)**	**0.207(±0.033)**	**0.18(±0.026)**	**0.174(±0.019)**	**0.172(±0.019)**	0.171(±0.028)
500	DCTC	0.54(±0.16)	0.36(±0.05)	0.18(±0.01)	0.49(±0.13)	0.35(±0.02)	0.15(±0.01)	0.49(±0.1)	0.23(±0.03)	**0.07(±0.01)**
	DDECV+Repair	**0.138(±0.02)**	**0.138(±0.022)**	**0.143(±0.022)**	**0.133(±0.018)**	**0.131(±0.019)**	**0.116(±0.02)**	**0.094(±0.013)**	**0.098(±0.012)**	0.101(±0.019)
1000	DCTC	0.47(±0.12)	0.32(±0.02)	0.17(±0.02)	0.43(±0.08)	0.32(±0.02)	0.12(±0.02)	0.39(±0.06)	0.19(±0.01)	**0.06(±0.01)**
	DDECV+Repair	**0.083(±0.014)**	**0.081(±0.012)**	**0.096(±0.014)**	**0.08(±0.01)**	**0.081(±0.011)**	**0.074(±0.011)**	**0.053(±0.008)**	**0.054(±0.003)**	0.064(±0.007)
	k=0.5		*S=10*			*S=20*			*S=50*	
250	DCTC	0.55(±0.13)	0.71(±0.08)	0.34(±0.04)	0.57(±0.13)	0.62(±0.05)	0.31(±0.04)	1.15(±0.13)	0.28(±0.06)	**0.12(±0.04)**
	DDECV+Repair	**0.292(±0.038)**	**0.3(±0.035)**	**0.231(±0.028)**	**0.291(±0.033)**	**0.288(±0.027)**	**0.248(±0.027)**	**0.249(±0.022)**	**0.244(±0.021)**	0.262(±0.027)
500	DCTC	0.49(±0.14)	0.63(±0.05)	0.28(±0.02)	0.51(±0.11)	0.55(±0.04)	0.26(±0.03)	1.03(±0.1)	0.2(±0.04)	**0.07(±0.03)**
	DDECV+Repair	**0.159(±0.023)**	**0.161(±0.021)**	**0.143(±0.02)**	**0.15(±0.015)**	**0.149(±0.015)**	**0.148(±0.018)**	**0.117(±0.01)**	**0.119(±0.01)**	0.145(±0.018)
1000	DCTC	0.41(±0.12)	0.57(±0.02)	0.25(±0.01)	0.45(±0.09)	0.5(±0.02)	0.23(±0.01)	0.98(±0.09)	0.15(±0.02)	**0.03(±0.02)**
	DDECV+Repair	**0.087(±0.016)**	**0.086(±0.014)**	**0.087(±0.014)**	**0.088(±0.01)**	**0.089(±0.01)**	**0.082(±0.01)**	**0.061(±0.006)**	**0.063(±0.005)**	0.082(±0.013)
	k=1.0		*S=10*			*S=20*			*S=50*	
250	DCTC	1.67(±0.26)	1.53(±0.13)	0.46(±0.08)	1.09(±0.29)	1.33(±0.09)	**0.28(±0.11)**	1.68(±0.2)	1.33(±0.11)	0.31(±0.09)
	DDECV+Repair	**0.458(±0.041)**	**0.465(±0.04)**	**0.278(±0.026)**	**0.46(±0.03)**	**0.474(±0.037)**	0.295(±0.029)	**0.421(±0.026)**	**0.42(±0.026)**	**0.308(±0.031)**
500	DCTC	1.59(±0.23)	1.41(±0.06)	0.38(±0.04)	0.93(±0.18)	1.26(±0.08)	0.2(±0.05)	1.54(±0.16)	1.2(±0.07)	0.25(±0.07)
	DDECV+Repair	**0.236(±0.015)**	**0.232(±0.021)**	**0.16(±0.018)**	**0.225(±0.013)**	**0.225(±0.014)**	**0.179(±0.022)**	**0.195(±0.014)**	**0.195(±0.014)**	**0.184(±0.019)**
1000	DCTC	1.44(±0.18)	1.36(±0.05)	0.34(±0.03)	0.83(±0.14)	1.17(±0.05)	0.15(±0.03)	1.45(±0.08)	1.13(±0.05)	0.2(±0.03)
	DDECV+Repair	**0.108(±0.011)**	**0.113(±0.011)**	**0.09(±0.011)**	**0.093(±0.008)**	**0.094(±0.008)**	**0.098(±0.011)**	**0.093(±0.008)**	**0.094(±0.008)**	**0.098(±0.011)**

Tabla 4.20: Valores promedio y desviación estándar de *offline error factible* obtenidos por DDECV + Repair y DCTC con distintas frecuencias de cambio (250, 500 y 1000 evaluaciones), y distintas severidades de cambio (k=0.25, 0.5 y 1.0), para los problemas de prueba con función objetivo dinámica y restricciones estáticas. Los mejores resultados se remarcan en negritas.

Evals	**Algoritmos**	**Funciones**			**Funciones**			**Funciones**		
		G24_1	**G24_2**	**G24_8b**	**G24_1**	**G24_2**	**G24_8b**	**G24_1**	**G24_2**	**G24_8b**
		k=0.25			*k=0.50*			*k=1.0*		
250	DCTC	**0.03(±0.01)**	**0.08(±0.03)**	**0.11(±0.04)**	**0.05(±0.03)**	**0.12(±0.03)**	**0.25(±0.08)**	**0.12(±0.04)**	**0.18(±0.1)**	0.58(±0.19)
	DDECV+Repair	0.186(±0.034)	0.159(±0.016)	0.217(±0.029)	0.27(±0.032)	0.164(±0.017)	0.377(±0.045)	0.46(±0.54)	0.252(±0.022)	**0.532(±0.065)**
500	DCTC	**0.0(±0.0)**	**0.05(±0.02)**	**0.04(±0.02)**	**0.01(±0.01)**	**0.06(±0.03)**	**0.12(±0.06)**	**0.03(±0.02)**	**0.12(±0.07)**	**0.29(±0.13)**
	DDECV+Repair	0.113(±0.028)	0.099(±0.012)	0.127(±0.024)	0.132(±0.02)	0.131(±0.014)	0.188(±0.027)	0.215(±0.026)	0.129(±0.015)	0.297(±0.048)
1000	DCTC	**0.0(±0.0)**	**0.03(±0.01)**	**0.01(±0.01)**	**0.0(±0.01)**	**0.03(±0.03)**	**0.03(±0.06)**	**0.0(±0.0)**	**0.04(±0.04)**	**0.07(±0.07)**
	DDECV+Repair	0.058(±0.012)	0.049(±0.006)	0.069(±0.018)	0.066(±0.015)	0.063(±0.012)	0.09(±0.028)	0.084(±0.011)	0.061(±0.008)	0.12(±0.019)

Tabla 4.21: Valores promedio y desviación estándar de *offline error factible* obtenidos por DDECV + Repair y DCTC con distintas frecuencias de cambio (250, 500 y 1000 evaluaciones), para los problemas de prueba con funcion objetivo dinámica y restricciones estáticas. Los mejores resultados se remarcan en negritas.

Evals	**Algorithms**	**Funciones**		
		G24_6a	**G24_6c**	**G24_6d**
250	DCTC	0.26(±0.38)	**0.12(±0.05)**	**0.14(±0.18)**
	DDECV+Repair	**0.183(±0.018)**	0.214(±0.031)	0.424(±0.03)
500	DCTC	**0.06(±0.12)**	**0.06(±0.03)**	**0.04(±0.14)**
	DDECV+Repair	0.082(±0.01)	0.095(±0.015)	0.186(±0.016)
1000	DCTC	**0.02(±0.02)**	**0.04(±0.03)**	**0.0(±0.0)**
	DDECV+Repair	0.035(±0.007)	0.041(±0.008)	0.079(±0.009)

Tabla 4.22: Valores promedio y desviación estándar de *offline error factible* obtenidos por DDECV + Repair y DCTC con distintas frecuencias de cambio (250, 500 y 1000 evaluaciones) y distintas severidades de cambio (S=10, 20 y 50), para los problemas de prueba con función objetivo estática y restricciones dinámicas. Los mejores resultados se remarcan en negritas.

Evals	**Algoritmos**	**Funciones**		**Funciones**		**Funciones**	
		G24_3	**G24_7**	**G24_3**	**G24_7**	**G24_3**	**G24_7**
		S=10		*S=20*		*S=50*	
250	DCTC	0.16(±0.15)	**0.15(±0.02)**	0.15(±0.21)	**0.11(±0.03)**	0.12(±0.07)	**0.1(±0.03)**
	DDECV+Repair	**0.130(±0.019)**	0.240(±0.038)	**0.105(±0.018)**	0.16(±0.03)	**0.072(±0.008)**	0.124(±0.032)
500	DCTC	0.13(±0.14)	**0.12(±0.02)**	0.1(±0.13)	**0.07(±0.02)**	0.1(±0.11)	**0.06(±0.02)**
	DDECV+Repair	**0.083(±0.01)**	0.175(±0.018)	**0.063(±0.009)**	0.15(±0.022)	**0.041(±0.008)**	0.112(±0.025)
1000	DCTC	0.11(±0.03)	**0.1(±0.02)**	0.05(±0.03)	**0.05(±0.01)**	0.05(±0.04)	**0.04(±0.01)**
	DDECV+Repair	**0.059(±0.005)**	0.11(±0.015)	**0.044(±0.008)**	0.115(±0.015)	**0.026(±0.004)**	0.108(±0.013)

Parte IV

Conclusiones

Capítulo 5

Conclusiones y Trabajo Futuro

En este capítulo se presentan las conclusiones obtenidas a partir de los experimentos planteados y resultados obtenidos, así como posibles direcciones de investigación derivadas de este trabajo de investigación.

5.1 Conclusiones

En esta tesis doctoral se presentó un algoritmo basado en evolución diferencial dinámica con combinación de variantes dinámica con un método de reparación (DDECV + Repair) para resolver problemas de optimización dinámica con restricciones (DCOPs). Se llevaron a cabo experimentos, para analizar el desempeño de DDECV + Repair en cuatro áreas: (1) El papel que juega cada uno de los elementos que componen a DDECV + Repair en su desempeño, los elementos son los siguientes: mecanismo que promueve la exploración, el método de reparación, los inmigrantes aleatorios y la población de memoria, (2) la sensibilidad de DECV + Repair ante diferentes frecuencias y severidades del cambio, (3) la habilidad de DDECV + Repair para detectar un cambio y recuperarse después de él, además del manejo de diversidad durante el proceso de búsqueda y (4) el desempeño de DDECV + Repair para resolver problemas con dinamismo en ambas (función objetivo y restricciones) o dinamismo sólo en una de ellas.

Se diseñaron cuatro experimentos para analizar lo expuesto anteriormente; para ello se usaron siete medidas de desempeño, así como ocho algoritmos encontrados en la literatura especializada que resuelven DCOPs, los cuales fueron probados en un conjunto de problemas propuesto recientemente. De acuerdo a los resultados obtenidos, validados estadísticamente, se encontró que el método de reparación es el mecanismo más importante en DDECV + Repair, mientras que la presencia de inmigrantes aleatorios tiene un efecto positivo, aunque puede afectar a los resultados en los problemas donde el óptimo global factible está localizado en los límites de la región factible o del espacio de búsqueda. El mecanismo de memoria juega un papel importante en DDECV + repair porque la solución óptima en ocasiones retorna a localizaciones cercanas a donde se ubicó previamente. Además, la sensibilidad de DDECV + Repair ante diferentes frecuencias y severidades del cambio fue baja, y de éstas dos características,los cambios rápidos

(frecuencia del cambio) pueden afectar la capacidad del algoritmo para recuperarse rápido después de un cambio. Por otro lado, el mecanismo de detección del cambio basado en la re-evaluación de soluciones demostró ser eficaz. También se mostró que al igual que en optimización estática con restricciones, mantener un porcentaje bajo de soluciones no factibles durante la búsqueda ayuda a DDECV + Repair a alcanzar resultados competitivos cuando resuelve DCOPs. Además, DDECV + Repair tuvo un mejor desempeño donde el dinamismo se presenta en ambas (función objetivo y restricciones).

Finalmente, derivado de los resultados obtenidos por el algoritmo propuesto presentado en el capítulo anterior, podemos concluir que la hipótesis propuesta (ver Capítulo 1) en este trabajo doctoral es válida, debido a que el algoritmo evolutivo propuesto, el cual está basado en evolución diferencial con mecanismos para hacer frente al dinamismos en la función objetivo y/o restricciones provee resultados competitivos con respecto a los algoritmos del estado del arte que resuelven DCOPs.

5.2 Trabajo futuro

A continuación se proponen algunas probables extensiones y líneas de investigación para la problemática planteada en esta tesis doctoral.

- Analizar el comportamiento de DDECV + Repair con frecuencias y severidades del cambio aleatorias.
- La técnica de inmigrantes aleatorios va a ser revisada para analizar el efecto negativo en la resolución de problemas donde el óptimo global factible se encuentra en la fronteras de la región factible o del espacio de búsqueda.
- Se diseñarán otras modificaciones a DDECV + Repair con la finalidad de mejorar su desempeño al hacer frente al dinamismo presente en la función objetivo y restricciones de manera separada.
- DDECV + Repair será probado en aplicaciones del mundo real.
- Se buscaran otros problemas de prueba con regiones factibles pequeñas para probar el método de reparación en estas situaciones.
- Analizar el desempeño de DDECV + Repair utilizando diferentes métodos de reparación.

Apéndice A: Funciones dinámicas de prueba

En este apéndice, introduciremos un conjunto de problemas de optimización dinámica unimodales llamados G24, basados en algunos problemas con restricciones estáticas. En todos los problemas del conjunto de prueba, las funciones objetivo o al menos una de las funciones de restricción se modificaron a partir de la función objetivo estática y/o de las funciones de restricción de una función estática propuesta en [28] y llamada G24 en la competencia sobre optimización de parámetros reales con restricciones CEC 2006.

La forma general para cada problema en el conjunto G24 es la siguiente:

$$\text{minimizar } f(\vec{x}) \tag{5.1}$$

Sujeto a:

$$g_i(\vec{x}) \leq 0,\ g_i(\vec{x}) \in G,\ i = 1, \ldots, m \tag{5.2}$$

donde la función objetivo $f(\vec{x})$ puede ser una de las formas de función establecidas en la ecuación 5.1, cada restricción $gi(\vec{x})$ puede ser una de las formas de función dadas en la ecuación 5.2, y G es el conjunto de m funciones de restricción para ese problema de prueba en particular. Las descripciones detalladas de $f(\vec{x})$ y $gi(\vec{x})$ para cada problema se describen en la Tabla 5.1 y en la Tabla 5.2.

La ecuación 5.1 describe las formas de función general para las funciones objetivo en el conjunto G24. De esas formas de función, se utiliza $f^{(2)}$ para el diseño de la función objetivo para G24_8a y G24_8b, y se utiliza $f^{(1)}$ para el diseño de las funciones objetivo para todos los demás problemas. Se modifica $f^{(1)}$ de una función estática propuesta en [28] y $f^{(2)}$ es una nueva función diseñada en [60].

$$\begin{aligned} f^{(1)} &= -(X_1 + X_2) \\ f^{(2)} &= -3\exp\left(-\sqrt{\sqrt{(X_1)^2 + (X_2)^2}}\right) \end{aligned} \tag{5.3}$$

$$g^{(1)} = -2Y_1^4 + 8Y_1^3 - 8Y_1^2 + Y_2 - 2 \tag{5.4}$$

$$g^{(2)} = -4Y_1^4 + 32Y_1^3 - 88Y_1^2 + 96Y_1 + Y_2 - 36 \tag{5.5}$$

$$g^{(3)} = 2Y_1 + 3Y_2 - 9 \tag{5.6}$$

$$g^{(4)} = \begin{cases} -1, & \text{si } (0 \leq Y_1(x_1,t) \leq 1) \text{ o } (2 \leq Y_1(x_1,t) \leq 3) \\ 1, & \text{de otra manera} \end{cases} \tag{5.7}$$

$$g^{(5)} = \begin{cases} -1, & \text{si } (0 \leq Y_1(x_1,t) \leq 0.5) \text{ o } (2 \leq Y_1(x_1,t) \leq 2.5) \\ 1, & \text{de otra manera} \end{cases} \tag{5.8}$$

$$g^{(6)} = \begin{cases} -1, & \text{si } [(0 \leq Y_1(x_1,t) \leq 1) \text{ y } (2 \leq Y_2(x_2,t) \leq 3)] \text{ o } (2 \leq Y_1(x_1,t) \leq 3) \\ 1, & \text{de otra manera} \end{cases} \tag{5.9}$$

Tabla 5.1: La forma de función objetivo de cada problema de prueba [60].

problema de prueba	**función objetivo**
G24_8a & G24_8a	$f(x) = f^{(2)}$
Todos los demás problemas	$f(x) = f^{(1)}$

Tabla 5.2: El conjunto de formas de función de restricción de cada problema de prueba [60].

problemas de prueba	**Conjunto de restricciones G**
G24_u, G24_uf, G24_2u and G24_8a	$G = \{\emptyset\}$
G24_6a	$G = \{g^{(3)}, g^{(6)}\}$
G24_6b	$G = \{g^{(3)}\}$
G24_6c	$G = \{g^{(3)}, g^{(4)}\}$
G24_6d	$G = \{g^{(5)}, g^{(6)}\}$
G24_6a	$G = \{g^{(3)}, g^{(6)}\}$
Todos los demás problemas	$G = \{g^{(1)}, g^{(2)}\}$

Tabla 5.3: Parámetros dinámicos para todos los problemas de prueba en el conjunto de pruebas G24 [64]

.

Problema	**Configuración de parámetros**
G24_u	$p_1(t) = sin(k\pi t + \frac{\pi}{2})$, $p_2(t) = 1$, $q_i(t) = 0$

Tabla 5.4: Valores de óptimos globales factibles de los problemas de prueba de optimización dinámica con restricciones en diferentes tiempos con una severidad media de cambio (k=0.5 and S=20) y el porcentaje estimado de la región factible se muestra entre paréntesis. La discrepancia con respecto a los valores de Nguyen [60] se indica con un asterisco.

tiempo t	**Funciones**					
	G24_u	**G24_1**	**G24_f**	**G24_uf**	**G24_2**	**G24_2u**
0	-7.0 (100.0%)	-5.5080133 (44.24%)	-5.5080133 (44.34%)	-7.0 (100.0%)	-3.0 (44.13%)	-3.0 (100.0%)
1	-4.0 (100.0%)	-3.4421046 (44.15%)	-5.5080133 (44.32%)	-7.0 (100.0%)	-4.8257126 (44.2%)	-6.1213203 (100.0%)
2	-4.0 (100.0%)	-2.8305013 (44.14%)	-5.5080133 (44.18%)	-7.0 (100.0%)	-3.4421046 (44.27%) *	-4.0 (100.0%) *
3	-4.0 (100.0%)	-3.4421046 (44.29%)	-5.5080133 (44.23%)	-7.0 (100.0%)	0.0 (44.27%)	0.0 (100.0%)
4	-7.0 (100.0%)	-5.5080133 (44.19%)	-5.5080133 (44.1%)	-7.0 (100.0%)	0.0 (44.22%)	0.0 (100.0%)
5	-4.0 (100.0%)	-3.4421046 (44.3%)	-5.5080133 (44.19%)	-7.0 (100.0%)	0.0 (44.22%)	0.0 (100.0%)
6	-4.0 (100.0%)	-2.8305013 (44.22%)	-5.5080133 (44.17%)	-7.0 (100.0%)	0.0 (44.27%)	0.0 (100.0%)
7	-4.0 (100.0%)	-3.4421046 (44.2%)	-5.5080133 (44.16%)	-7.0 (100.0%)	-2.1213203 (44.2%)	-2.1213203 (100.0%)
8	-7.0 (100.0%)	-5.5080133 (44.21%)	-5.5080133 (44.27%)	-7.0 (100.0%)	-3.0 (44.16%) *	-3.0 (100.0%) *
9	-4.0 (100.0%)	-3.4421046 (44.15%)	-5.5080133 (44.14%)	-7.0 (100.0%)	-4.8257126 (44.24%) *	-6.1213203 (100.0%) *
10	-4.0 (100.0%)	-2.8305013 (44.21%)	-5.5080133 (44.15%)	-7.0 (100.0%)	-3.4421046 (44.09%)	-4.0 (100.0%)
11	-4.0 (100.0%)	-3.4421046 (44.26%)	-5.5080133 (44.1%)	-7.0 (100.0%)	0.0 (44.19%)	0.0 (100.0%)
	G24_3	**G24_3b**	**G24_3f**	**G24_4**	**G24_5**	**G24_6a**
0	-3.5080133 (7.1%)	-3.5080133 (7.11%)	-3.5080133 (7.11%)	-5.5080133 (44.29%)	-3.0 (44.19%)	-4.0 (16.68%)
1	-3.7080133 (10.11%)	-1.6421046 (10.12%)	-3.5080133 (7.09%)	-3.2421046 (39.56%)	-4.6257126 (39.59%) *	-3.0 (16.68%)
2	-3.9080133 (13.2%)	-1.2305013 (13.32%)	-3.5080133 (7.14%)	-2.4305013 (35.33%)	-3.0421046 (35.21%)	-4.0 (16.73%)
3	-4.1080133 (16.59%)	-2.0421046 (16.65%)	-3.5080133 (7.09%)	-2.8421046 (31.31%)	0.0 (31.28%)	-3.0 (16.68%)
4	-4.3080133 (20.04%)	-4.3080133 (19.98%)	-3.5080133 (7.13%)	-4.7080133 (27.36%)	0.0 (27.36%)	-4.0 (16.76%)
5	-4.5080133 (23.67%)	-2.4421046 (23.57%)	-3.5080133 (7.16%)	-2.4421046 (23.53%)	0.0 (23.53%)	-3.0 (16.65%)
6	-4.7080133 (27.31%)	-2.0305013 (27.33%)	-3.5080133 (7.13%)	-1.6305013 (19.98%)	0.0 (20.02%)	-4.0 (16.71%)
7	-4.9080133 (31.2%)	-2.8421046 (31.23%)	-3.5080133 (7.12%)	-2.0421046 (16.61%)	-1.8660941 (16.55%) *	-3.0 (16.67%)
8	-5.1080133 (35.22%)	-5.1080133 (35.33%)	-3.5080133 (7.15%)	-3.9080133 (13.31%)	-2.6062545 (13.33%)	-4.0 (16.66%)
9	-5.3080133 (39.53%)	-3.2421046 (39.67%)	-3.5080133 (7.09%)	-1.6421046 (10.16%)	-3.0257126 (10.18%)	-3.0 (16.77%)
10	-5.5080133 (44.26%)	-2.8305013 (44.28%)	-3.5080133 (7.14%)	-0.8305013 (7.11%)	-1.4421046 (7.1%)	-4.0 (16.65%)
11	-5.7080133 (49.2%)	-3.6421046 (49.21%)	-3.5080133 (7.12%)	-1.2421046 (4.75%)	0.1223965 (4.8%)	-3.0 (16.67%)
	G24_6b	**G24_6c**	**G24_6d**	**G24_7**	**G24_8a**	**G24_8b**
0	-4.0 (49.94%)	-4.0 (33.31%)	-6.5 (20.9%)	-5.5080133 (44.22%)	-3.0 (100.0%)	-1.7953334 (44.27%)
1	-3.0 (49.96%)	-3.0 (33.4%)	-3.0 (20.77%)	-5.3080133 (39.65%)	-1.7331301 (100.0%)	-0.9964644 (44.24%)
2	-4.0 (50.04%)	-4.0 (33.24%)	-6.5 (20.79%)	-5.1080133 (35.35%)	-3.0 (100.0%)	-3.0 (44.14%)
3	-3.0 (49.99%)	-3.0 (33.3%)	-3.0 (20.78%)	-4.9080133 (31.16%)	-3.0 (100.0%)	-2.2413935 (44.19%)
4	-4.0 (50.01%)	-4.0 (33.4%)	-6.5 (20.9%)	-4.7080133 (27.38%)	-3.0 (100.0%)	-1.7953334 (44.28%)
5	-3.0 (50.03%)	-3.0 (33.34%)	-3.0 (20.76%)	-4.5080133 (23.7%)	-1.7331301 (100.0%)	-0.9964644 (44.29%)
6	-4.0 (50.03%)	-4.0 (33.38%)	-6.5 (20.87%)	-4.3080133 (20.07%)	-3.0 (100.0%)	-3.0 (44.16%)
7	-3.0 (50.07%)	-3.0 (33.29%)	-3.0 (20.81%)	-4.1080133 (16.55%)	-3.0 (100.0%)	-2.2413935 (44.27%)
8	-4.0 (50.02%)	-4.0 (33.32%)	-6.5 (20.82%)	-3.9080133 (13.32%)	-3.0 (100.0%)	-1.7953334 (44.12%)
9	-3.0 (49.96%)	-3.0 (33.33%)	-3.0 (20.91%)	-3.7080133 (10.12%)	-1.7331301 (100.0%)	-0.9964644 (44.24%)
10	-4.0 (50.02%)	-4.0 (33.32%)	-6.5 (20.78%)	-3.5080133 (7.14%)	-3.0 (100.0%)	-3.0 (44.19%)
11	-3.0 (50.01%)	-3.0 (33.27%)	-3.0 (20.8%)	-3.3080133 (4.75%)	-3.0 (100.0%)	-2.2413935 (44.26%)

Tabla 5.5: Valores de óptimos globales factibles de los problemas de prueba de optimización dinámica con restricciones en diferentes tiempos con una severidad baja de cambio (k=0.25 and S=10) y el porcentaje estimado de la región factible se muestra entre paréntesis.

tiempo t	Funciones					
	G24_u	**G24_1**	**G24_f**	**G24_uf**	**G24_2**	**G24_2u**
0	-7.0 (100.0%)	-5.5080133 (44.14%)	-5.5080133 (44.23%)	-7.0 (100.0%)	-3.0 (44.26%)	-3.0 (100.0%)
1	-6.1213203 (100.0%)	-4.8257126 (44.22%)	-5.5080133 (44.23%)	-7.0 (100.0%)	-5.3306891 (44.23%)	-6.7716386 (100.0%)
2	-4.0 (100.0%)	-3.4421046 (44.22%)	-5.5080133 (44.18%)	-7.0 (100.0%)	-4.8257126 (44.19%)	-6.1213203 (100.0%)
3	-4.0 (100.0%)	-3.0096358 (44.16%)	-5.5080133 (44.22%)	-7.0 (100.0%)	-3.1390028 (44.17%)	-3.9764774 (100.0%)
4	-4.0 (100.0%)	-2.8305013 (44.19%)	-5.5080133 (44.16%)	-7.0 (100.0%)	-2.4339355 (44.13%)	-2.8284271 (100.0%)
5	-4.0 (100.0%)	-3.0096358 (44.14%)	-5.5080133 (44.22%)	-7.0 (100.0%)	0.0 (44.27%)	0.0 (100.0%)
6	-4.0 (100.0%)	-3.4421046 (44.21%)	-5.5080133 (44.17%)	-7.0 (100.0%)	0.0 (44.21%)	0.0 (100.0%)
7	-6.1213203 (100.0%)	-4.8257126 (44.12%)	-5.5080133 (44.26%)	-7.0 (100.0%)	0.0 (44.24%)	0.0 (100.0%)
8	-7.0 (100.0%)	-5.5080133 (44.06%)	-5.5080133 (44.13%)	-7.0 (100.0%)	0.0 (44.3%)	0.0 (100.0%)
9	-6.1213203 (100.0%)	-4.8257126 (44.24%)	-5.5080133 (44.26%)	-7.0 (100.0%)	0.0 (44.14%)	0.0 (100.0%)
10	-4.0 (100.0%)	-3.4421046 (44.3%)	-5.5080133 (44.14%)	-7.0 (100.0%)	0.0 (44.23%)	0.0 (100.0%)
11	-4.0 (100.0%)	-3.0096358 (44.18%)	-5.5080133 (44.21%)	-7.0 (100.0%)	0.0 (44.15%)	0.0 (100.0%)
12	-4.0 (100.0%)	-2.8305013 (44.07%)	-5.5080133 (44.16%)	-7.0 (100.0%)	0.0 (44.27%)	0.0 (100.0%)
13	-4.0 (100.0%)	-3.0096358 (44.18%)	-5.5080133 (44.19%)	-7.0 (100.0%)	-1.1480503 (44.14%)	-1.1480503 (100.0%)
14	-4.0 (100.0%)	-3.4421046 (44.16%)	-5.5080133 (44.19%)	-7.0 (100.0%)	-2.1213203 (44.21%)	-2.1213203 (100.0%)
15	-6.1213203 (100.0%)	-4.8257126 (44.27%)	-5.5080133 (44.2%)	-7.0 (100.0%)	-4.39973 (44.21%)	-5.6000657 (100.0%)
16	-7.0 (100.0%)	-5.5080133 (44.29%)	-5.5080133 (44.17%)	-7.0 (100.0%)	-4.5770542 (44.23%)	-5.8284271 (100.0%)
17	-6.1213203 (100.0%)	-4.8257126 (44.15%)	-5.5080133 (44.28%)	-7.0 (100.0%)	-5.3306891 (44.14%)	-6.7716386 (100.0%)
18	-4.0 (100.0%)	-3.4421046 (44.31%)	-5.5080133 (44.23%)	-7.0 (100.0%)	-4.8257126 (44.27%)	-6.1213203 (100.0%)
19	-4.0 (100.0%)	-3.0096358 (44.27%)	-5.5080133 (44.16%)	-7.0 (100.0%)	-3.1390028 (44.16%)	-3.9764774 (100.0%)
	G24_3	**G24_3b**	**G24_3f**	**G24_4**	**G24_5**	**G24_6a**
0	-3.5080133 (7.16%)	-3.5080133 (7.18%)	-3.5080133 (7.11%)	-5.5080133 (44.16%)	-3.0 (44.16%)	-4.0 (16.67%)
1	-3.9080133 (13.25%)	-3.2257126 (13.35%)	-3.5080133 (7.15%)	-4.4257126 (35.3%)	-4.9306891 (35.3%)	-3.0 (16.64%)
2	-4.3080133 (19.99%)	-2.2421046 (19.98%)	-3.5080133 (7.13%)	-2.6421046 (27.36%)	-4.0257126 (27.29%)	-4.0 (16.64%)
3	-4.7080133 (27.27%)	-2.2096358 (27.39%)	-3.5080133 (7.09%)	-1.8096358 (20.04%)	-2.2904747 (19.97%)	-3.0 (16.76%)
4	-5.1080133 (35.22%)	-2.4305013 (35.37%)	-3.5080133 (7.15%)	-1.2305013 (13.35%)	-1.3025646 (13.27%)	-4.0 (16.67%)
5	-5.5080133 (44.23%)	-3.0096358 (44.31%)	-3.5080133 (7.09%)	-1.0096358 (7.12%)	0.0 (7.12%)	-3.0 (16.61%)
6	-5.9080133 (54.25%)	-3.8421046 (54.22%)	-3.5080133 (7.12%)	-1.0421046 (2.99%)	0.1813757 (3.05%)	-4.0 (16.62%)
7	-6.3080133 (64.06%)	-5.6257126 (64.15%)	-3.5080133 (7.11%)	-2.0257126 (0.89%)	0.3637734 (0.87%)	-3.0 (16.62%)
8	-6.4041534 (73.32%)	-6.4041534 (73.28%)	-3.5080133 (7.15%)	-0.8537078 (0.1%)	0.5252334 (0.1%)	-4.0 (16.64%)
9	-6.4747666 (81.22%)	-5.7499242 (81.17%)	-3.5080133 (7.13%)	-0.4271261 (0.0%)	0.5580671 (0.0%)	-3.0 (16.65%)
10	-6.5411961 (87.05%)	-4.0 (87.11%)	-3.5080133 (7.1%)	0.0 (0.0%)	0.4142136 (0.0%)	-4.0 (16.68%)
11	-6.6062545 (90.91%)	-4.0 (90.89%)	-3.5080133 (7.07%)	0.4020849 (0.0%)	0.2176068 (0.0%)	-3.0 (16.67%)
12	-6.6725158 (94.17%)	-4.0 (94.18%)	-3.5080133 (7.09%)	0.5524348 (0.0%)	0.0 (0.0%)	-4.0 (16.71%)
13	-6.7434961 (96.88%)	-4.0 (96.87%)	-3.5080133 (7.06%)	0.3797631 (0.0%)	-0.2055263 (0.0%)	-3.0 (16.68%)
14	-6.8269052 (98.92%)	-4.0 (98.92%)	-3.5080133 (7.14%)	0.0 (0.0%)	-0.3694131 (0.0%)	-4.0 (16.7%)
15	-7.0 (100.0%)	-6.1213203 (100.0%)	-3.5080133 (7.11%)	-0.3595229 (0.0%)	-0.4697393 (0.0%)	-3.0 (16.63%)
16	-7.0 (100.0%)	-7.0 (100.0%)	-3.5080133 (7.1%)	-0.4950379 (0.0%)	-0.4950379 (0.0%)	-4.0 (16.7%)
17	-7.0 (100.0%)	-6.1213203 (100.0%)	-3.5080133 (7.13%)	-0.3409381 (0.0%)	-0.4454571 (0.0%)	-3.0 (16.69%)
18	-7.0 (100.0%)	-4.0 (100.0%)	-3.5080133 (7.15%)	0.0 (0.0%)	-0.332169 (0.0%)	-4.0 (16.72%)
19	-7.0 (100.0%)	-4.0 (100.0%)	-3.5080133 (7.09%)	0.323708 (0.0%)	-0.1751895 (0.0%)	-3.0 (16.65%)
	G24_6b	**G24_6c**	**G24_6d**	**G24_7**	**G24_8a**	**G24_8b**
0	-4.0 (49.95%)	-4.0 (33.3%)	-6.5 (20.85%)	-5.5080133 (44.26%)	-3.0 (100.0%)	-1.7953334 (44.27%)
1	-3.0 (49.97%)	-3.0 (33.4%)	-3.0 (20.95%)	-5.1080133 (35.25%)	-2.4017398 (100.0%)	-1.1577345 (44.24%)
2	-4.0 (49.98%)	-4.0 (33.29%)	-6.5 (20.83%)	-4.7080133 (27.35%)	-1.7331301 (100.0%)	-0.9964644 (44.24%)
3	-3.0 (50.07%)	-3.0 (33.37%)	-3.0 (20.83%)	-4.3080133 (20.06%)	-2.4017398 (100.0%)	-1.3334966 (44.22%)
4	-4.0 (49.97%)	-4.0 (33.28%)	-6.5 (20.84%)	-3.9080133 (13.33%)	-3.0 (100.0%)	-3.0 (44.17%)
5	-3.0 (49.98%)	-3.0 (33.29%)	-3.0 (20.83%)	-3.5080133 (7.1%)	-3.0 (100.0%)	-1.87681 (44.19%)
6	-4.0 (49.97%)	-4.0 (33.29%)	-6.5 (20.85%)	-3.1080133 (3.01%)	-3.0 (100.0%)	-2.2413935 (44.13%)
7	-3.0 (50.01%)	-3.0 (33.37%)	-3.0 (20.87%)	-2.7080133 (0.86%)	-3.0 (100.0%)	-1.9124641 (44.18%)
8	-4.0 (49.96%)	-4.0 (33.32%)	-6.5 (20.86%)	-0.8537078 (0.1%)	-3.0 (100.0%)	-1.7953334 (44.21%)
9	-3.0 (49.92%)	-3.0 (33.4%)	-3.0 (20.81%)	-0.6040475 (0.0%)	-2.4017398 (100.0%)	-1.1577345 (44.17%)
10	-4.0 (49.91%)	-4.0 (33.31%)	-6.5 (20.87%)	-0.5857864 (0.0%)	-1.7331301 (100.0%)	-0.9964644 (44.3%)
11	-3.0 (50.04%)	-3.0 (33.39%)	-3.0 (20.78%)	-0.5686339 (0.0%)	-2.4017398 (100.0%)	-1.3334966 (44.21%)
12	-4.0 (50.01%)	-4.0 (33.3%)	-6.5 (20.81%)	-0.5524348 (0.0%)	-3.0 (100.0%)	-3.0 (44.32%)
13	-3.0 (50.01%)	-3.0 (33.45%)	-3.0 (20.92%)	-0.5370662 (0.0%)	-3.0 (100.0%)	-1.87681 (44.23%)
14	-4.0 (49.96%)	-4.0 (33.3%)	-6.5 (20.81%)	-0.522429 (0.0%)	-3.0 (100.0%)	-2.2413935 (44.23%)
15	-3.0 (49.96%)	-3.0 (33.39%)	-3.0 (20.84%)	-0.5084421 (0.0%)	-3.0 (100.0%)	-1.9124641 (44.18%)
16	-4.0 (49.93%)	-4.0 (33.41%)	-6.5 (20.9%)	-0.4950379 (0.0%)	-3.0 (100.0%)	-1.7953334 (44.22%)
17	-3.0 (50.0%)	-3.0 (33.27%)	-3.0 (20.82%)	-0.4821593 (0.0%)	-2.4017398 (100.0%)	-1.1577345 (44.18%)
18	-4.0 (50.04%)	-4.0 (33.37%)	-6.5 (20.9%)	-0.4697579 (0.0%)	-1.7331301 (100.0%)	-0.9964644 (44.11%)
19	-3.0 (49.97%)	-3.0 (33.31%)	-3.0 (20.81%)	-0.4577922 (0.0%)	-2.4017398 (100.0%)	-1.3334966 (44.14%)

Tabla 5.6: Valores de óptimos globales factibles de los problemas de prueba de optimización dinámica con restricciones en diferentes tiempos con una severidad baja del cambio en la función objetivo (k=0.25) y una severidad media de cambio en las restricciones (S=20), el porcentaje estimado de la región factible se muestra entre paréntesis.

tiempo t	Funciones					
	G24_u	**G24_1**	**G24_f**	**G24_uf**	**G24_2**	**G24_2u**
0	-7.0 (100.0%)	-5.5080133 (44.2%)	-5.5080133 (44.17%)	-7.0 (100.0%)	-3.0 (44.23%)	-3.0 (100.0%)
1	-6.1213203 (100.0%)	-4.8257126 (44.17%)	-5.5080133 (44.22%)	-7.0 (100.0%)	-5.3306891 (44.19%)	-6.7716386 (100.0%)
2	-4.0 (100.0%)	-3.4421046 (44.33%)	-5.5080133 (44.26%)	-7.0 (100.0%)	-4.8257126 (44.25%)	-6.1213203 (100.0%)
3	-4.0 (100.0%)	-3.0096358 (44.15%)	-5.5080133 (44.16%)	-7.0 (100.0%)	-3.1390028 (44.16%)	-3.9764774 (100.0%)
4	-4.0 (100.0%)	-2.8305013 (44.15%)	-5.5080133 (44.22%)	-7.0 (100.0%)	-2.4339355 (44.11%)	-2.8284271 (100.0%)
5	-4.0 (100.0%)	-3.0096358 (44.15%)	-5.5080133 (44.18%)	-7.0 (100.0%)	0.0 (44.26%)	0.0 (100.0%)
6	-4.0 (100.0%)	-3.4421046 (44.16%)	-5.5080133 (44.15%)	-7.0 (100.0%)	0.0 (44.21%)	0.0 (100.0%)
7	-6.1213203 (100.0%)	-4.8257126 (44.25%)	-5.5080133 (44.14%)	-7.0 (100.0%)	0.0 (44.12%)	0.0 (100.0%)
8	-7.0 (100.0%)	-5.5080133 (44.21%)	-5.5080133 (44.2%)	-7.0 (100.0%)	0.0 (44.21%)	0.0 (100.0%)
9	-6.1213203 (100.0%)	-4.8257126 (44.22%)	-5.5080133 (44.22%)	-7.0 (100.0%)	0.0 (44.24%)	0.0 (100.0%)
10	-4.0 (100.0%)	-3.4421046 (44.25%)	-5.5080133 (44.16%)	-7.0 (100.0%)	0.0 (44.3%)	0.0 (100.0%)
11	-4.0 (100.0%)	-3.0096358 (44.17%)	-5.5080133 (44.23%)	-7.0 (100.0%)	0.0 (44.37%)	0.0 (100.0%)
12	-4.0 (100.0%)	-2.8305013 (44.21%)	-5.5080133 (44.25%)	-7.0 (100.0%)	0.0 (44.22%)	0.0 (100.0%)
13	-4.0 (100.0%)	-3.0096358 (44.19%)	-5.5080133 (44.21%)	-7.0 (100.0%)	-1.1480503 (44.18%)	-1.1480503 (100.0%)
14	-4.0 (100.0%)	-3.4421046 (44.15%)	-5.5080133 (44.19%)	-7.0 (100.0%)	-2.1213203 (44.23%)	-2.1213203 (100.0%)
15	-6.1213203 (100.0%)	-4.8257126 (44.19%)	-5.5080133 (44.25%)	-7.0 (100.0%)	-4.39973 (44.17%)	-5.6000657 (100.0%)
16	-7.0 (100.0%)	-5.5080133 (44.08%)	-5.5080133 (44.3%)	-7.0 (100.0%)	-4.5770542 (44.11%)	-5.8284271 (100.0%)
17	-6.1213203 (100.0%)	-4.8257126 (44.21%)	-5.5080133 (44.19%)	-7.0 (100.0%)	-5.3306891 (44.18%)	-6.7716386 (100.0%)
18	-4.0 (100.0%)	-3.4421046 (44.18%)	-5.5080133 (44.33%)	-7.0 (100.0%)	-4.8257126 (44.22%)	-6.1213203 (100.0%)
19	-4.0 (100.0%)	-3.0096358 (44.2%)	-5.5080133 (44.19%)	-7.0 (100.0%)	-3.1390028 (44.06%)	-3.9764774 (100.0%)
	G24_3	**G24_3b**	**G24_3f**	**G24_4**	**G24_5**	**G24_6a**
0	-3.5080133 (7.14%)	-3.5080133 (7.06%)	-3.5080133 (7.1%)	-5.5080133 (44.19%)	-3.0 (44.11%)	-4.0 (16.68%)
1	-3.7080133 (10.06%)	-3.0257126 (10.12%)	-3.5080133 (7.06%)	-4.6257126 (39.57%)	-5.1306891 (39.54%)	-3.0 (16.66%)
2	-3.9080133 (13.31%)	-1.8421046 (13.31%)	-3.5080133 (7.14%)	-3.0421046 (35.27%)	-4.4257126 (35.32%)	-4.0 (16.68%)
3	-4.1080133 (16.59%)	-1.6096358 (16.56%)	-3.5080133 (7.16%)	-2.4096358 (31.24%)	-2.7147387 (31.14%)	-3.0 (16.71%)
4	-4.3080133 (19.95%)	-1.6305013 (19.99%)	-3.5080133 (7.1%)	-2.0305013 (27.33%)	-1.8682501 (27.33%)	-4.0 (16.71%)
5	-4.5080133 (23.64%)	-2.0096358 (23.53%)	-3.5080133 (7.13%)	-2.0096358 (23.56%)	0.0 (23.52%)	-3.0 (16.66%)
6	-4.7080133 (27.3%)	-2.6421046 (27.37%)	-3.5080133 (7.14%)	-2.2421046 (20.0%)	0.0 (20.04%)	-4.0 (16.7%)
7	-4.9080133 (31.21%)	-4.2257126 (31.1%)	-3.5080133 (7.1%)	-3.4257126 (16.61%)	0.0 (16.58%)	-3.0 (16.62%)
8	-5.1080133 (35.39%)	-5.1080133 (35.29%)	-3.5080133 (7.11%)	-3.9080133 (13.29%)	0.0 (13.33%)	-4.0 (16.65%)
9	-5.3080133 (39.56%)	-4.6257126 (39.55%)	-3.5080133 (7.12%)	-3.0257126 (10.13%)	0.0 (10.14%)	-3.0 (16.68%)
10	-5.5080133 (44.17%)	-3.4421046 (44.23%)	-3.5080133 (7.09%)	-1.4421046 (7.06%)	0.0 (7.08%)	-4.0 (16.59%)
11	-5.7080133 (49.23%)	-3.2096358 (49.22%)	-3.5080133 (7.15%)	-0.8096358 (4.79%)	0.0662405 (4.77%)	-3.0 (16.61%)
12	-5.9080133 (54.26%)	-3.2305013 (54.23%)	-3.5080133 (7.1%)	-0.4305013 (3.01%)	0.0 (3.0%)	-4.0 (16.67%)
13	-6.1080133 (59.27%)	-3.5786728 (59.13%)	-3.5080133 (7.11%)	-0.4096358 (1.73%)	-0.9338233 (1.75%)	-3.0 (16.71%)
14	-6.3080133 (64.04%)	-4.0 (64.16%)	-3.5080133 (7.16%)	-0.6421046 (0.88%)	-1.6999932 (0.88%)	-4.0 (16.65%)
15	-6.3660254 (68.89%)	-5.6730326 (68.87%)	-3.5080133 (7.13%)	-1.8257126 (0.37%)	-2.2784097 (0.37%)	-3.0 (16.64%)
16	-6.4041534 (73.41%)	-6.4041534 (73.31%)	-3.5080133 (7.06%)	-0.8537078 (0.1%)	-0.7827971 (0.1%)	-4.0 (16.67%)
17	-6.4401979 (77.51%)	-5.7254805 (77.5%)	-3.5080133 (7.14%)	-0.4745734 (0.0%)	-0.6071523 (0.0%)	-3.0 (16.68%)
18	-6.4747666 (81.17%)	-4.0 (81.16%)	-3.5080133 (7.14%)	0.0 (0.0%)	-0.4271261 (0.0%)	-4.0 (16.6%)
19	-6.5083111 (84.4%)	-3.8776035 (84.46%)	-3.5080133 (7.1%)	0.4205639 (0.0%)	-0.2276075 (0.0%)	-3.0 (16.67%)
	G24_6b	**G24_6c**	**G24_6d**	**G24_7**	**G24_8a**	**G24_8b**
0	-4.0 (50.05%)	-4.0 (33.37%)	-6.5 (20.88%)	-5.5080133 (44.17%)	-3.0 (100.0%)	-1.7953334 (44.14%)
1	-3.0 (50.08%)	-3.0 (33.28%)	-3.0 (20.86%)	-5.3080133 (39.63%)	-2.4017398 (100.0%)	-1.1577345 (44.18%)
2	-4.0 (49.95%)	-4.0 (33.29%)	-6.5 (20.85%)	-5.1080133 (35.22%)	-1.7331301 (100.0%)	-0.9964644 (44.21%)
3	-3.0 (50.03%)	-3.0 (33.33%)	-3.0 (20.79%)	-4.9080133 (31.18%)	-2.4017398 (100.0%)	-1.3334966 (44.2%)
4	-4.0 (50.0%)	-4.0 (33.27%)	-6.5 (20.85%)	-4.7080133 (27.38%)	-3.0 (100.0%)	-3.0 (44.15%)
5	-3.0 (50.11%)	-3.0 (33.29%)	-3.0 (20.86%)	-4.5080133 (23.6%)	-3.0 (100.0%)	-1.87681 (44.19%)
6	-4.0 (50.0%)	-4.0 (33.41%)	-6.5 (20.88%)	-4.3080133 (20.05%)	-3.0 (100.0%)	-2.2413935 (44.19%)
7	-3.0 (50.0%)	-3.0 (33.33%)	-3.0 (20.85%)	-4.1080133 (16.57%)	-3.0 (100.0%)	-1.9124641 (44.21%)
8	-4.0 (49.97%)	-4.0 (33.38%)	-6.5 (20.79%)	-3.9080133 (13.28%)	-3.0 (100.0%)	-1.7953334 (44.16%)
9	-3.0 (50.03%)	-3.0 (33.38%)	-3.0 (20.84%)	-3.7080133 (10.17%)	-2.4017398 (100.0%)	-1.1577345 (44.19%)
10	-4.0 (49.98%)	-4.0 (33.29%)	-6.5 (20.85%)	-3.5080133 (7.12%)	-1.7331301 (100.0%)	-0.9964644 (44.24%)
11	-3.0 (50.05%)	-3.0 (33.39%)	-3.0 (20.86%)	-3.3080133 (4.74%)	-2.4017398 (100.0%)	-1.3334966 (44.18%)
12	-4.0 (50.01%)	-4.0 (33.27%)	-6.5 (20.88%)	-3.1080133 (2.99%)	-3.0 (100.0%)	-3.0 (44.16%)
13	-3.0 (50.0%)	-3.0 (33.38%)	-3.0 (20.82%)	-2.9080133 (1.74%)	-3.0 (100.0%)	-1.87681 (44.09%)
14	-4.0 (50.01%)	-4.0 (33.35%)	-6.5 (20.88%)	-2.7080133 (0.88%)	-3.0 (100.0%)	-2.2413935 (44.17%)
15	-3.0 (49.93%)	-3.0 (33.37%)	-3.0 (20.78%)	-2.5080133 (0.38%)	-3.0 (100.0%)	-1.9124641 (44.24%)
16	-4.0 (49.95%)	-4.0 (33.33%)	-6.5 (20.92%)	-0.8537078 (0.1%)	-3.0 (100.0%)	-1.7953334 (44.19%)
17	-3.0 (50.07%)	-3.0 (33.35%)	-3.0 (20.79%)	-0.6537078 (0.0%)	-2.4017398 (100.0%)	-1.1577345 (44.28%)
18	-4.0 (50.01%)	-4.0 (33.43%)	-6.5 (20.8%)	-0.6040475 (0.0%)	-1.7331301 (100.0%)	-0.9964644 (44.14%)
19	-3.0 (50.02%)	-3.0 (33.33%)	-3.0 (20.79%)	-0.5947671 (0.0%)	-2.4017398 (100.0%)	-1.3334966 (44.21%)

Tabla 5.7: Valores de óptimos globales factibles de los problemas de prueba de optimización dinámica con restricciones en diferentes tiempos con una severidad baja del cambio en la función objetivo (k=0.25) y una severidad alta de cambio en las restricciones (S=50), el porcentaje estimado de la región factible se muestra entre paréntesis.

tiempo t	**Funciones**					
	G24_u	**G24_1**	**G24_f**	**G24_uf**	**G24_2**	**G24_2u**
0	-7.0 (100.0%)	-5.508013 (44.21%)	-5.508013 (44.18%)	-7.0 (100.0%)	-3.0 (44.23%)	-3.0 (100.0%)
1	-6.12132 (100.0%)	-4.825713 (44.21%)	-5.508013 (44.18%)	-7.0 (100.0%)	-5.330689 (44.12%)	-6.771639 (100.0%)
2	-4.0 (100.0%)	-3.442105 (44.24%)	-5.508013 (44.27%)	-7.0 (100.0%)	-4.825713 (44.26%)	-6.12132 (100.0%)
3	-4.0 (100.0%)	-3.009636 (44.21%)	-5.508013 (44.25%)	-7.0 (100.0%)	-3.139003 (44.21%)	-3.976477 (100.0%)
4	-4.0 (100.0%)	-2.830501 (44.28%)	-5.508013 (44.26%)	-7.0 (100.0%)	-2.433935 (44.24%)	-2.828427 (100.0%)
5	-4.0 (100.0%)	-3.009636 (44.25%)	-5.508013 (44.24%)	-7.0 (100.0%)	0.0 (44.19%)	0.0 (100.0%)
6	-4.0 (100.0%)	-3.442105 (44.23%)	-5.508013 (44.23%)	-7.0 (100.0%)	0.0 (44.26%)	0.0 (100.0%)
7	-6.12132 (100.0%)	-4.825713 (44.13%)	-5.508013 (44.16%)	-7.0 (100.0%)	0.0 (44.17%)	0.0 (100.0%)
8	-7.0 (100.0%)	-5.508013 (44.2%)	-5.508013 (44.17%)	-7.0 (100.0%)	0.0 (44.16%)	0.0 (100.0%)
9	-6.12132 (100.0%)	-4.825713 (44.24%)	-5.508013 (44.2%)	-7.0 (100.0%)	0.0 (44.19%)	0.0 (100.0%)
10	-4.0 (100.0%)	-3.442105 (44.24%)	-5.508013 (44.22%)	-7.0 (100.0%)	0.0 (44.18%)	0.0 (100.0%)
11	-4.0 (100.0%)	-3.009636 (44.24%)	-5.508013 (44.19%)	-7.0 (100.0%)	0.0 (44.16%)	0.0 (100.0%)
12	-4.0 (100.0%)	-2.830501 (44.2%)	-5.508013 (44.25%)	-7.0 (100.0%)	0.0 (44.29%)	0.0 (100.0%)
13	-4.0 (100.0%)	-3.009636 (44.2%)	-5.508013 (44.16%)	-7.0 (100.0%)	-1.14805 (44.14%)	-1.14805 (100.0%)
14	-4.0 (100.0%)	-3.442105 (44.21%)	-5.508013 (44.21%)	-7.0 (100.0%)	-2.12132 (44.22%)	-2.12132 (100.0%)
15	-6.12132 (100.0%)	-4.825713 (44.3%)	-5.508013 (44.28%)	-7.0 (100.0%)	-4.39973 (44.24%)	-5.600066 (100.0%)
16	-7.0 (100.0%)	-5.508013 (44.18%)	-5.508013 (44.19%)	-7.0 (100.0%)	-4.577054 (44.27%)	-5.828427 (100.0%)
17	-6.12132 (100.0%)	-4.825713 (44.2%)	-5.508013 (44.32%)	-7.0 (100.0%)	-5.330689 (44.29%)	-6.771639 (100.0%)
18	-4.0 (100.0%)	-3.442105 (44.17%)	-5.508013 (44.23%)	-7.0 (100.0%)	-4.825713 (44.19%)	-6.12132 (100.0%)
19	-4.0 (100.0%)	-3.009636 (44.18%)	-5.508013 (44.18%)	-7.0 (100.0%)	-3.139003 (44.19%)	-3.976477 (100.0%)
	G24_3	**G24_3b**	**G24_3f**	**G24_4**	**G24_5**	**G24_6a**
0	-3.508013 (7.16%)	-3.508013 (7.1%)	-3.508013 (7.13%)	-5.508013 (44.21%)	-3.0 (44.15%)	-4.0 (16.64%)
1	-3.588013 (8.31%)	-2.905713 (8.29%)	-3.508013 (7.1%)	-4.745713 (42.32%)	-5.250689 (42.29%)	-3.0 (16.64%)
2	-3.668013 (9.5%)	-1.602105 (9.51%)	-3.508013 (7.13%)	-3.282105 (40.48%)	-4.665713 (40.58%)	-4.0 (16.69%)
3	-3.748013 (10.72%)	-1.249636 (10.74%)	-3.508013 (7.09%)	-2.769636 (38.7%)	-2.969297 (38.72%)	-3.0 (16.68%)
4	-3.828013 (12.03%)	-1.150501 (12.05%)	-3.508013 (7.11%)	-2.510501 (37.03%)	-2.207661 (36.97%)	-4.0 (16.71%)
5	-3.908013 (13.32%)	-1.409636 (13.34%)	-3.508013 (7.13%)	-2.609636 (35.29%)	0.0 (35.3%)	-3.0 (16.7%)
6	-3.988013 (14.6%)	-1.922105 (14.62%)	-3.508013 (7.13%)	-2.962105 (33.66%)	0.0 (33.62%)	-4.0 (16.68%)
7	-4.068013 (15.98%)	-3.385713 (16.03%)	-3.508013 (7.13%)	-4.265713 (31.94%)	0.0 (31.98%)	-3.0 (16.61%)
8	-4.148013 (17.34%)	-4.148013 (17.23%)	-3.508013 (7.09%)	-4.868013 (30.41%)	0.0 (30.46%)	-4.0 (16.6%)
9	-4.228013 (18.56%)	-3.545713 (18.61%)	-3.508013 (7.17%)	-4.105713 (28.83%)	0.0 (28.9%)	-3.0 (16.64%)
10	-4.308013 (20.05%)	-2.242105 (20.08%)	-3.508013 (7.08%)	-2.642105 (27.44%)	0.0 (27.34%)	-4.0 (16.67%)
11	-4.388013 (21.42%)	-1.889636 (21.4%)	-3.508013 (7.14%)	-2.129636 (25.79%)	0.0 (25.8%)	-3.0 (16.64%)
12	-4.468013 (22.89%)	-1.790501 (22.89%)	-3.508013 (7.13%)	-1.870501 (24.43%)	0.0 (24.36%)	-4.0 (16.67%)
13	-4.548013 (24.36%)	-2.049636 (24.36%)	-3.508013 (7.15%)	-1.969636 (22.85%)	-1.033272 (22.82%)	-3.0 (16.7%)
14	-4.628013 (25.77%)	-2.562105 (25.76%)	-3.508013 (7.19%)	-2.322105 (21.43%)	-1.89942 (21.39%)	-4.0 (16.67%)
15	-4.708013 (27.35%)	-4.025713 (27.31%)	-3.508013 (7.12%)	-3.625713 (20.05%)	-3.551202 (20.03%)	-3.0 (16.7%)
16	-4.788013 (28.85%)	-4.788013 (28.85%)	-3.508013 (7.11%)	-4.228013 (18.58%)	-3.671958 (18.62%)	-4.0 (16.64%)
17	-4.868013 (30.45%)	-4.185713 (30.4%)	-3.508013 (7.15%)	-3.465713 (17.28%)	-3.970689 (17.3%)	-3.0 (16.64%)
18	-4.948013 (32.01%)	-2.882105 (32.13%)	-3.508013 (7.1%)	-2.002105 (15.91%)	-3.385713 (15.92%)	-4.0 (16.67%)
19	-5.028013 (33.67%)	-2.529636 (33.56%)	-3.508013 (7.1%)	-1.489636 (14.61%)	-2.0642 (14.58%)	-3.0 (16.65%)
	G24_6b	**G24_6c**	**G24_6d**	**G24_7**	**G24_8a**	**G24_8b**
0	-4.0 (50.05%)	-4.0 (33.26%)	-6.5 (20.77%)	-5.508013 (44.25%)	-3.0 (100.0%)	-1.795333 (44.22%)
1	-3.0 (50.03%)	-3.0 (33.33%)	-3.0 (20.87%)	-5.428013 (42.23%)	-2.40174 (100.0%)	-1.157734 (44.18%)
2	-4.0 (49.98%)	-4.0 (33.38%)	-6.5 (20.86%)	-5.348013 (40.51%)	-1.73313 (100.0%)	-0.996464 (44.1%)
3	-3.0 (50.07%)	-3.0 (33.28%)	-3.0 (20.81%)	-5.268013 (38.71%)	-2.40174 (100.0%)	-1.333497 (44.19%)
4	-4.0 (50.06%)	-4.0 (33.33%)	-6.5 (20.84%)	-5.188013 (37.01%)	-3.0 (100.0%)	-3.0 (44.13%)
5	-3.0 (49.98%)	-3.0 (33.4%)	-3.0 (20.86%)	-5.108013 (35.27%)	-3.0 (100.0%)	-1.87681 (44.22%)
6	-4.0 (50.0%)	-4.0 (33.4%)	-6.5 (20.86%)	-5.028013 (33.63%)	-3.0 (100.0%)	-2.241393 (44.27%)
7	-3.0 (50.02%)	-3.0 (33.41%)	-3.0 (20.83%)	-4.948013 (32.04%)	-3.0 (100.0%)	-1.912464 (44.27%)
8	-4.0 (50.01%)	-4.0 (33.3%)	-6.5 (20.84%)	-4.868013 (30.41%)	-3.0 (100.0%)	-1.795333 (44.21%)
9	-3.0 (50.01%)	-3.0 (33.41%)	-3.0 (20.82%)	-4.788013 (28.95%)	-2.40174 (100.0%)	-1.157734 (44.27%)
10	-4.0 (50.03%)	-4.0 (33.36%)	-6.5 (20.83%)	-4.708013 (27.32%)	-1.73313 (100.0%)	-0.996464 (44.22%)
11	-3.0 (50.01%)	-3.0 (33.29%)	-3.0 (20.9%)	-4.628013 (25.8%)	-2.40174 (100.0%)	-1.333497 (44.29%)
12	-4.0 (50.0%)	-4.0 (33.29%)	-6.5 (20.82%)	-4.548013 (24.34%)	-3.0 (100.0%)	-3.0 (44.23%)
13	-3.0 (49.92%)	-3.0 (33.32%)	-3.0 (20.76%)	-4.468013 (22.88%)	-3.0 (100.0%)	-1.87681 (44.14%)
14	-4.0 (49.87%)	-4.0 (33.27%)	-6.5 (20.88%)	-4.388013 (21.38%)	-3.0 (100.0%)	-2.241393 (44.19%)
15	-3.0 (50.0%)	-3.0 (33.47%)	-3.0 (20.81%)	-4.308013 (20.1%)	-3.0 (100.0%)	-1.912464 (44.26%)
16	-4.0 (49.97%)	-4.0 (33.34%)	-6.5 (20.84%)	-4.228013 (18.65%)	-3.0 (100.0%)	-1.795333 (44.21%)
17	-3.0 (49.97%)	-3.0 (33.35%)	-3.0 (20.83%)	-4.148013 (17.23%)	-2.40174 (100.0%)	-1.157734 (44.18%)
18	-4.0 (50.04%)	-4.0 (33.36%)	-6.5 (20.87%)	-4.068013 (15.93%)	-1.73313 (100.0%)	-0.996464 (44.22%)
19	-3.0 (50.0%)	-3.0 (33.38%)	-3.0 (20.77%)	-3.988013 (14.61%)	-2.40174 (100.0%)	-1.333497 (44.25%)

Tabla 5.8: Valores de óptimos globales factibles de los problemas de prueba de optimización dinámica con restricciones en diferentes tiempos con una severidad media del cambio en la función objetivo (k=0.5) y una severidad de baja cambio en las restricciones (S=10), el porcentaje estimado de la región factible se muestra entre paréntesis.

tiempo t	Funciones					
	G24_u	**G24_1**	**G24_f**	**G24_uf**	**G24_2**	**G24_2u**
0	-7.0 (100.0%)	-5.508013 (44.33%)	-5.508013 (44.21%)	-7.0 (100.0%)	-3.0 (44.21%)	-3.0 (100.0%)
1	-6.12132 (100.0%)	-4.825713 (44.23%)	-5.508013 (44.23%)	-7.0 (100.0%)	-5.330689 (44.22%)	-6.771639 (100.0%)
2	-4.0 (100.0%)	-3.442105 (44.15%)	-5.508013 (44.15%)	-7.0 (100.0%)	-4.825713 (44.26%)	-6.12132 (100.0%)
3	-4.0 (100.0%)	-3.009636 (44.22%)	-5.508013 (44.16%)	-7.0 (100.0%)	-3.139003 (44.26%)	-3.976477 (100.0%)
4	-4.0 (100.0%)	-2.830501 (44.24%)	-5.508013 (44.2%)	-7.0 (100.0%)	-2.433935 (44.2%)	-2.828427 (100.0%)
5	-4.0 (100.0%)	-3.009636 (44.18%)	-5.508013 (44.08%)	-7.0 (100.0%)	0.0 (44.24%)	0.0 (100.0%)
6	-4.0 (100.0%)	-3.442105 (44.24%)	-5.508013 (44.21%)	-7.0 (100.0%)	0.0 (44.26%)	0.0 (100.0%)
7	-6.12132 (100.0%)	-4.825713 (44.25%)	-5.508013 (44.23%)	-7.0 (100.0%)	0.0 (44.16%)	0.0 (100.0%)
8	-7.0 (100.0%)	-5.508013 (44.27%)	-5.508013 (44.23%)	-7.0 (100.0%)	0.0 (44.19%)	0.0 (100.0%)
9	-6.12132 (100.0%)	-4.825713 (44.24%)	-5.508013 (44.2%)	-7.0 (100.0%)	0.0 (44.18%)	0.0 (100.0%)
	G24_3	**G24_3b**	**G24_3f**	**G24_4**	**G24_5**	**G24_6a**
0	-3.508013 (7.15%)	-3.508013 (7.06%)	-3.508013 (7.09%)	-5.508013 (44.12%)	-3.0 (44.26%)	-4.0 (16.69%)
1	-3.588013 (13.31%)	-2.905713 (13.3%)	-3.508013 (7.11%)	-4.745713 (35.31%)	-5.250689 (35.39%)	-3.0 (16.65%)
2	-3.668013 (19.99%)	-1.602105 (20.01%)	-3.508013 (7.07%)	-3.282105 (27.38%)	-4.665713 (27.29%)	-4.0 (16.66%)
3	-3.748013 (27.41%)	-1.249636 (27.34%)	-3.508013 (7.14%)	-2.769636 (19.93%)	-2.969297 (20.04%)	-3.0 (16.63%)
4	-3.828013 (35.3%)	-1.150501 (35.31%)	-3.508013 (7.14%)	-2.510501 (13.33%)	-2.207661 (13.31%)	-4.0 (16.68%)
5	-3.908013 (44.14%)	-1.409636 (44.24%)	-3.508013 (7.11%)	-2.609636 (7.08%)	0.0 (7.09%)	-3.0 (16.63%)
6	-3.988013 (54.22%)	-1.922105 (54.2%)	-3.508013 (7.16%)	-2.962105 (3.0%)	0.0 (3.01%)	-4.0 (16.62%)
7	-4.068013 (64.02%)	-3.385713 (64.16%)	-3.508013 (7.09%)	-4.265713 (0.87%)	0.0 (0.86%)	-3.0 (16.72%)
8	-4.148013 (73.3%)	-4.148013 (73.3%)	-3.508013 (7.12%)	-4.868013 (0.09%)	0.0 (0.1%)	-4.0 (16.63%)
9	-4.228013 (81.28%)	-3.545713 (81.18%)	-3.508013 (7.15%)	-4.105713 (0.0%)	0.0 (0.0%)	-3.0 (16.73%)
	G24_6b	**G24_6c**	**G24_6d**	**G24_7**	**G24_8a**	**G24_8b**
0	-4.0 (50.02%)	-4.0 (33.38%)	-6.5 (20.92%)	-5.508013 (44.17%)	-3.0 (100.0%)	-1.795333 (44.2%)
1	-3.0 (50.01%)	-3.0 (33.36%)	-3.0 (20.88%)	-5.428013 (35.2%)	-2.40174 (100.0%)	-1.157734 (44.2%)
2	-4.0 (49.95%)	-4.0 (33.34%)	-6.5 (20.83%)	-5.348013 (27.32%)	-1.73313 (100.0%)	-0.996464 (44.17%)
3	-3.0 (50.09%)	-3.0 (33.31%)	-3.0 (20.94%)	-5.268013 (19.97%)	-2.40174 (100.0%)	-1.333497 (44.22%)
4	-4.0 (50.0%)	-4.0 (33.38%)	-6.5 (20.82%)	-5.188013 (13.28%)	-3.0 (100.0%)	-3.0 (44.18%)
5	-3.0 (50.04%)	-3.0 (33.36%)	-3.0 (20.83%)	-5.108013 (7.11%)	-3.0 (100.0%)	-1.87681 (44.26%)
6	-4.0 (49.98%)	-4.0 (33.33%)	-6.5 (20.76%)	-5.028013 (3.04%)	-3.0 (100.0%)	-2.241393 (44.23%)
7	-3.0 (50.03%)	-3.0 (33.37%)	-3.0 (20.89%)	-4.948013 (0.87%)	-3.0 (100.0%)	-1.912464 (44.29%)
8	-4.0 (50.04%)	-4.0 (33.33%)	-6.5 (20.86%)	-4.868013 (0.1%)	-3.0 (100.0%)	-1.795333 (44.18%)
9	-3.0 (50.02%)	-3.0 (33.35%)	-3.0 (20.85%)	-4.788013 (0.0%)	-2.40174 (100.0%)	-1.157734 (44.24%)

Tabla 5.9: Valores de óptimos globales factibles de los problemas de prueba de optimización dinámica con restricciones en diferentes tiempos con una severidad media del cambio en la función objetivo (k=0.5) y una severidad alta de cambio en las restricciones (S=50), el porcentaje estimado de la región factible se muestra entre paréntesis.

tiempo t	Funciones					
	G24_u	**G24_1**	**G24_f**	**G24_uf**	**G24_2**	**G24_2u**
0	-7.0 (100.0%)	-5.508013 (44.29%)	-5.508013 (44.22%)	-7.0 (100.0%)	-3.0 (44.24%)	-3.0 (100.0%)
1	-7.0 (100.0%)	-5.508013 (44.26%)	-5.508013 (44.2%)	-7.0 (100.0%)	-3.0 (44.19%)	-3.0 (100.0%)
2	-4.0 (100.0%)	-3.442105 (44.28%)	-5.508013 (44.28%)	-7.0 (100.0%)	-4.825713 (44.22%)	-6.12132 (100.0%)
3	-4.0 (100.0%)	-2.830501 (44.32%)	-5.508013 (44.32%)	-7.0 (100.0%)	-3.442105 (44.14%)	-4.0 (100.0%)
4	-4.0 (100.0%)	-3.442105 (44.17%)	-5.508013 (44.27%)	-7.0 (100.0%)	0.0 (44.27%)	0.0 (100.0%)
5	-7.0 (100.0%)	-5.508013 (44.26%)	-5.508013 (44.15%)	-7.0 (100.0%)	0.0 (44.17%)	0.0 (100.0%)
6	-4.0 (100.0%)	-3.442105 (44.18%)	-5.508013 (44.28%)	-7.0 (100.0%)	0.0 (44.11%)	0.0 (100.0%)
7	-4.0 (100.0%)	-2.830501 (44.22%)	-5.508013 (44.23%)	-7.0 (100.0%)	0.0 (44.13%)	0.0 (100.0%)
8	-4.0 (100.0%)	-3.442105 (44.29%)	-5.508013 (44.22%)	-7.0 (100.0%)	-2.12132 (44.22%)	-2.12132 (100.0%)
9	-7.0 (100.0%)	-5.508013 (44.2%)	-5.508013 (44.21%)	-7.0 (100.0%)	-3.0 (44.18%)	-3.0 (100.0%)
	G24_3	**G24_3b**	**G24_3f**	**G24_4**	**G24_5**	**G24_6a**
0	-3.508013 (7.14%)	-3.508013 (7.13%)	-3.508013 (7.05%)	-5.508013 (44.18%)	-3.0 (44.2%)	-4.0 (16.64%)
1	-3.508013 (8.3%)	-3.508013 (8.32%)	-3.508013 (7.14%)	-5.508013 (42.31%)	-3.0 (42.29%)	-4.0 (16.6%)
2	-3.588013 (9.48%)	-1.522105 (9.61%)	-3.508013 (7.07%)	-3.362105 (40.51%)	-4.745713 (40.55%)	-3.0 (16.61%)
3	-3.668013 (10.76%)	-0.990501 (10.76%)	-3.508013 (7.1%)	-2.670501 (38.66%)	-3.282105 (38.68%)	-4.0 (16.61%)
4	-3.748013 (12.03%)	-1.682105 (12.01%)	-3.508013 (7.11%)	-3.202105 (36.98%)	0.0 (36.96%)	-3.0 (16.69%)
5	-3.828013 (13.24%)	-3.828013 (13.33%)	-3.508013 (7.12%)	-5.188013 (35.26%)	0.0 (35.28%)	-4.0 (16.69%)
6	-3.908013 (14.68%)	-1.842105 (14.64%)	-3.508013 (7.11%)	-3.042105 (33.71%)	0.0 (33.73%)	-3.0 (16.67%)
7	-3.988013 (15.96%)	-1.310501 (15.92%)	-3.508013 (7.09%)	-2.350501 (32.02%)	0.0 (32.01%)	-4.0 (16.7%)
8	-4.068013 (17.26%)	-2.002105 (17.24%)	-3.508013 (7.08%)	-2.882105 (30.42%)	-1.973604 (30.43%)	-3.0 (16.62%)
9	-4.148013 (18.63%)	-4.148013 (18.63%)	-3.508013 (7.06%)	-4.868013 (28.9%)	-2.774597 (28.87%)	-4.0 (16.65%)
	G24_6b	**G24_6c**	**G24_6d**	**G24_7**	**G24_8a**	**G24_8b**
0	-4.0 (49.98%)	-4.0 (33.42%)	-6.5 (20.78%)	-5.508013 (44.15%)	-3.0 (100.0%)	-1.795333 (44.2%)
1	-4.0 (49.98%)	-4.0 (33.29%)	-6.5 (20.81%)	-5.508013 (42.25%)	-3.0 (100.0%)	-1.795333 (44.26%)
2	-3.0 (50.05%)	-3.0 (33.3%)	-3.0 (20.8%)	-5.428013 (40.47%)	-1.73313 (100.0%)	-0.996464 (44.26%)

Continúa en la página siguiente.

Tabla 5.9 – continúa de la página anterior

tiempo t	Funciones					
	G24_6b	**G24_6c**	**G24_6d**	**G24_7**	**G24_8a**	**G24_8b**
3	-4.0 (49.94%)	-4.0 (33.37%)	-6.5 (20.88%)	-5.348013 (38.69%)	-3.0 (100.0%)	-3.0 (44.19%)
4	-3.0 (50.02%)	-3.0 (33.33%)	-3.0 (20.9%)	-5.268013 (37.04%)	-3.0 (100.0%)	-2.241393 (44.2%)
5	-4.0 (50.0%)	-4.0 (33.37%)	-6.5 (20.89%)	-5.188013 (35.31%)	-3.0 (100.0%)	-1.795333 (44.2%)
6	-3.0 (50.02%)	-3.0 (33.38%)	-3.0 (20.79%)	-5.108013 (33.58%)	-1.73313 (100.0%)	-0.996464 (44.24%)
7	-4.0 (49.99%)	-4.0 (33.34%)	-6.5 (20.79%)	-5.028013 (31.98%)	-3.0 (100.0%)	-3.0 (44.16%)
8	-3.0 (49.99%)	-3.0 (33.38%)	-3.0 (20.75%)	-4.948013 (30.4%)	-3.0 (100.0%)	-2.241393 (44.22%)
9	-4.0 (49.97%)	-4.0 (33.36%)	-6.5 (20.85%)	-4.868013 (28.8%)	-3.0 (100.0%)	-1.795333 (44.27%)

Tabla 5.10: Valores de óptimos globales factibles de los problemas de prueba de optimización dinámica con restricciones en diferentes tiempos con una severidad alta del cambio en la función objetivo (k=1.0) y una severidad baja de cambio en las restricciones (S=10), el porcentaje estimado de la región factible se muestra entre paréntesis.

tiempo t	Funciones					
	G24_u	**G24_1**	**G24_f**	**G24_uf**	**G24_2**	**G24_2u**
0	-7.0 (100.0%)	-5.508013 (44.18%)	-5.508013 (44.2%)	-7.0 (100.0%)	-3.0 (44.23%)	-3.0 (100.0%)
1	-4.0 (100.0%)	-2.830501 (44.24%)	-5.508013 (44.16%)	-7.0 (100.0%)	-3.442105 (44.14%)	-4.0 (100.0%)
2	-7.0 (100.0%)	-5.508013 (44.2%)	-5.508013 (44.3%)	-7.0 (100.0%)	-2.830501 (44.22%)	-4.0 (100.0%)
3	-4.0 (100.0%)	-2.830501 (44.13%)	-5.508013 (44.12%)	-7.0 (100.0%)	0.0 (44.22%)	0.0 (100.0%)
4	-7.0 (100.0%)	-5.508013 (44.13%)	-5.508013 (44.25%)	-7.0 (100.0%)	-3.0 (44.21%)	-3.0 (100.0%)
	G24_3	**G24_3b**	**G24_3f**	**G24_4**	**G24_5**	**G24_6a**
0	-3.508013 (7.13%)	-3.508013 (7.1%)	-3.508013 (7.16%)	-5.508013 (44.19%)	-3.0 (44.21%)	-4.0 (16.65%)
1	-3.908013 (13.27%)	-1.230501 (13.27%)	-3.508013 (7.08%)	-2.430501 (35.35%)	-3.042105 (35.3%)	-3.0 (16.65%)
2	-4.308013 (20.02%)	-4.308013 (20.03%)	-3.508013 (7.06%)	-4.708013 (27.33%)	-2.030501 (27.29%)	-4.0 (16.71%)
3	-4.708013 (27.29%)	-2.030501 (27.32%)	-3.508013 (7.13%)	-1.630501 (20.05%)	0.0 (20.08%)	-3.0 (16.76%)
4	-5.108013 (35.28%)	-5.108013 (35.3%)	-3.508013 (7.14%)	-3.908013 (13.2%)	-2.606254 (13.28%)	-4.0 (16.64%)
	G24_6b	**G24_6c**	**G24_6d**	**G24_7**	**G24_8a**	**G24_8b**
0	-4.0 (50.04%)	-4.0 (33.31%)	-6.5 (20.86%)	-5.508013 (44.24%)	-3.0 (100.0%)	-1.795333 (44.21%)
1	-3.0 (50.04%)	-3.0 (33.32%)	-3.0 (20.9%)	-5.108013 (35.23%)	-3.0 (100.0%)	-3.0 (44.21%)
2	-4.0 (50.06%)	-4.0 (33.28%)	-6.5 (20.8%)	-4.708013 (27.28%)	-3.0 (100.0%)	-1.795333 (44.12%)
3	-3.0 (49.99%)	-3.0 (33.32%)	-3.0 (20.87%)	-4.308013 (20.01%)	-3.0 (100.0%)	-3.0 (44.22%)
4	-4.0 (49.98%)	-4.0 (33.37%)	-6.5 (20.83%)	-3.908013 (13.28%)	-3.0 (100.0%)	-1.795333 (44.22%)

Tabla 5.11: Valores de óptimos globales factibles de los problemas de prueba de optimización dinámica con restricciones en diferentes tiempos con una severidad alta del cambio en la función objetivo (k=1.0) y una severidad media de cambio en las restricciones (S=20), el porcentaje estimado de la región factible se muestra entre paréntesis.

tiempo t	Funciones					
	G24_u	**G24_1**	**G24_f**	**G24_uf**	**G24_2**	**G24_2u**
0	-7.0 (100.0%)	-5.508013 (44.24%)	-5.508013 (44.17%)	-7.0 (100.0%)	-3.0 (44.28%)	-3.0 (100.0%)
1	-4.0 (100.0%)	-2.830501 (44.14%)	-5.508013 (44.25%)	-7.0 (100.0%)	-3.442105 (44.18%)	-4.0 (100.0%)
2	-7.0 (100.0%)	-5.508013 (44.18%)	-5.508013 (44.26%)	-7.0 (100.0%)	-2.830501 (44.26%)	-4.0 (100.0%)
3	-4.0 (100.0%)	-2.830501 (44.15%)	-5.508013 (44.19%)	-7.0 (100.0%)	0.0 (44.18%)	0.0 (100.0%)
4	-7.0 (100.0%)	-5.508013 (44.27%)	-5.508013 (44.22%)	-7.0 (100.0%)	-3.0 (44.2%)	-3.0 (100.0%)
	G24_3	**G24_3b**	**G24_3f**	**G24_4**	**G24_5**	**G24_6a**
0	-3.508013 (7.11%)	-3.508013 (7.12%)	-3.508013 (7.16%)	-5.508013 (44.26%)	-3.0 (44.19%)	-4.0 (16.67%)
1	-3.708013 (10.1%)	-1.030501 (10.15%)	-3.508013 (7.06%)	-2.630501 (39.61%)	-3.242105 (39.62%)	-3.0 (16.63%)
2	-3.908013 (13.29%)	-3.908013 (13.34%)	-3.508013 (7.09%)	-5.108013 (35.27%)	-2.430501 (35.31%)	-4.0 (16.66%)
3	-4.108013 (16.57%)	-1.430501 (16.59%)	-3.508013 (7.09%)	-2.230501 (31.23%)	0.0 (31.18%)	-3.0 (16.7%)
4	-4.308013 (20.04%)	-4.308013 (20.09%)	-3.508013 (7.14%)	-4.708013 (27.33%)	-2.743496 (27.31%)	-4.0 (16.65%)
	G24_6b	**G24_6c**	**G24_6d**	**G24_7**	**G24_8a**	**G24_8b**
0	-4.0 (49.86%)	-4.0 (33.27%)	-6.5 (20.85%)	-5.508013 (44.26%)	-3.0 (100.0%)	-1.795333 (44.26%)
1	-3.0 (50.01%)	-3.0 (33.35%)	-3.0 (20.76%)	-5.308013 (39.53%)	-3.0 (100.0%)	-3.0 (44.25%)
2	-4.0 (49.99%)	-4.0 (33.33%)	-6.5 (20.86%)	-5.108013 (35.33%)	-3.0 (100.0%)	-1.795333 (44.19%)
3	-3.0 (50.03%)	-3.0 (33.35%)	-3.0 (20.8%)	-4.908013 (31.19%)	-3.0 (100.0%)	-3.0 (44.21%)
4	-4.0 (49.87%)	-4.0 (33.31%)	-6.5 (20.87%)	-4.708013 (27.3%)	-3.0 (100.0%)	-1.795333 (44.22%)

Tabla 5.12: Valores de óptimos globales factibles de los problemas de prueba de optimización dinámica con restricciones en diferentes tiempos con una severidad alta del cambio (k=1.0 and S=50) y el porcentaje estimado de la región factible se muestra entre paréntesis.

tiempo t	**Funciones**					
	G24_u	**G24_1**	**G24_f**	**G24_uf**	**G24_2**	**G24_2u**
0	-7.0 (100.0%)	-5.508013 (44.14%)	-5.508013 (44.26%)	-7.0 (100.0%)	-3.0 (44.23%)	-3.0 (100.0%)
1	-4.0 (100.0%)	-2.830501 (44.15%)	-5.508013 (44.16%)	-7.0 (100.0%)	-3.442105 (44.2%)	-4.0 (100.0%)
2	-7.0 (100.0%)	-5.508013 (44.32%)	-5.508013 (44.17%)	-7.0 (100.0%)	-2.830501 (44.24%)	-4.0 (100.0%)
3	-4.0 (100.0%)	-2.830501 (44.22%)	-5.508013 (44.26%)	-7.0 (100.0%)	0.0 (44.18%)	0.0 (100.0%)
4	-7.0 (100.0%)	-5.508013 (44.29%)	-5.508013 (44.23%)	-7.0 (100.0%)	-3.0 (44.28%)	-3.0 (100.0%)
	G24_3	**G24_3b**	**G24_3f**	**G24_4**	**G24_5**	**G24_6a**
0	-3.508013 (7.09%)	-3.508013 (7.12%)	-3.508013 (7.13%)	-5.508013 (44.21%)	-3.0 (44.24%)	-4.0 (16.61%)
1	-3.588013 (8.32%)	-0.910501 (8.34%)	-3.508013 (7.1%)	-2.750501 (42.43%)	-3.362105 (42.26%)	-3.0 (16.66%)
2	-3.668013 (9.56%)	-3.668013 (9.53%)	-3.508013 (7.09%)	-5.348013 (40.51%)	-2.670501 (40.48%)	-4.0 (16.67%)
3	-3.748013 (10.75%)	-1.070501 (10.73%)	-3.508013 (7.12%)	-2.590501 (38.68%)	0.0 (38.73%)	-3.0 (16.68%)
4	-3.828013 (11.99%)	-3.828013 (12.01%)	-3.508013 (7.06%)	-5.188013 (36.93%)	-2.846851 (37.0%)	-4.0 (16.69%)
	G24_6b	**G24_6c**	**G24_6d**	**G24_7**	**G24_8a**	**G24_8b**
0	-4.0 (50.05%)	-4.0 (33.37%)	-6.5 (20.78%)	-5.508013 (44.18%)	-3.0 (100.0%)	-1.795333 (44.15%)
1	-3.0 (50.02%)	-3.0 (33.32%)	-3.0 (20.85%)	-5.428013 (42.36%)	-3.0 (100.0%)	-3.0 (44.13%)
2	-4.0 (49.99%)	-4.0 (33.28%)	-6.5 (20.89%)	-5.348013 (40.39%)	-3.0 (100.0%)	-1.795333 (44.28%)
3	-3.0 (49.97%)	-3.0 (33.28%)	-3.0 (20.86%)	-5.268013 (38.68%)	-3.0 (100.0%)	-3.0 (44.27%)
4	-4.0 (49.97%)	-4.0 (33.35%)	-6.5 (20.84%)	-5.188013 (36.95%)	-3.0 (100.0%)	-1.795333 (44.18%)

Referencias

[1] V.-J. Aguilera-Rueda, M.-Y. Ameca-Alducin, E. Mezura-Montes, and N. Cruz-Ramírez. Particle swarm optimization with feasibility rules in constrained numerical optimization. a brief review. In *Power, Electronics and Computing (ROPEC), 2016 IEEE International Autumn Meeting on*, pages 1–6, Noviembre 2016.

[2] M.-Y. Ameca-Alducin, N. Cruz-Ramírez, E. Mezura-Montes, E. Martín-Del-Campo-Mena, N. Pérez-Castro, and H. G. Acosta-Mesa. *Assessment of Bayesian Network Classifiers as Tools for Discriminating Breast Cancer Pre-diagnosis Based on Three Diagnostic Methods*, pages 419–431. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[3] M.-Y. Ameca-Alducin, E. Mezura-Montes, and N. Cruz-Ramirez. Differential evolution with combined variants for dynamic constrained optimization. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 975–982, July 2014.

[4] M.-Y. Ameca-Alducin, E. Mezura-Montes, and N. Cruz-Ramírez. Differential evolution with a repair method to solve dynamic constrained optimization problems. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference*, GECCO Companion '15, pages 1169–1172, New York, NY, USA, 2015. ACM.

[5] M.-Y. Ameca-Alducin, E. Mezura-Montes, and N. Cruz-Ramírez. A repair method for differential evolution with combined variants to solve dynamic constrained optimization problems. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, GECCO '15, pages 241–248, New York, NY, USA, 2015. ACM.

[6] M.-Y. Ameca-Alducin, E. Mezura-Montes, and N. Cruz-Ramírez. Differential evolution with combined variants plus a repair method to solve dynamic constrained optimization problems: A comparative study. *Soft Computing*, pages 1–30, 2016.

[7] V. Aragón, S. Esquivel, and C. Coello. Artificial immune system for solving dynamic constrained optimization problems. In E. Alba, A. Nakib, and P. Siarry, editors, *Metaheuristics for Dynamic Optimization*, volume 433 of *Studies in Computational Intelligence*, pages 225–263. Springer Berlin Heidelberg, 2013.

[8] R. Azzouz, S. Bechikh, and L. B. Said. A dynamic multi-objective evolutionary algorithm using a change severity-based adaptive population management strategy. *Soft Computing*, pages 1–22, 2015.

[9] T. Blackwell. *Particle Swarm Optimization in Dynamic Environments*, pages 29–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[10] J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, page 1882 Vol. 3, 1999.

[11] J. Branke and H. Schmeck. Designing evolutionary algorithms for dynamic optimization problems. In A. Ghosh and S. Tsutsui, editors, *Advances in Evolutionary Computing*, Natural Computing Series, pages 239–262. Springer Berlin Heidelberg, 2003.

[12] C. Bu, W. Luo, and L. Yue. Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies. *IEEE Transactions on Evolutionary Computation*, PP(99):1–1, 2016.

[13] L. T. Bui, H. A. Abbass, and J. Branke. Multiobjective optimization for dynamic environments. In *2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2349–2356 Vol. 3, Sept 2005.

[14] T. Bäck. On the behavior of evolutionary algorithms in dynamic environments. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 446–451, May 1998.

[15] A. Carlisle and G. Dozler. Tracking changing extrema with adaptive particle swarm optimizer. In *Automation Congress, 2002 Proceedings of the 5th Biannual World*, volume 13, pages 265–270, 2002.

[16] H. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical report, Naval Research Lab Washington DC, 1990.

[17] H. Cobb and J. Grefenstette. Genetic algorithms for tracking changing environments. In S. Forrest, editor, *ICGA*, pages 523–530. Morgan Kaufmann, 1993.

[18] C. A. Coello Coello. Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, January 2002.

[19] E. Collingwood, D. Corne, and P. Ross. Useful diversity via multiploidy. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 810–813, May 1996.

[20] N. Cruz-Ramírez, H. G. Acosta-Mesa, E. Mezura-Montes, A. Guerra-Hernández, G. Hoyos-Rivera, R.-E. Barrientos-Martínez, K. Gutiérrez-Fragoso, L. A. Nava-Fernández, P. González-Gaspar, Novoa-Del-Toro, V.-J. Aguilera-Rueda, and M.-Y. Ameca-Alducin. How good is crude mdl for solving the bias-variance dilemma? an empirical investigation based on bayesian networks. *PLoS ONE*, 9(3), 2014.

[21] N. Cruz-Ramírez, E. Mezura-Montes, M.-Y. Ameca-Alducin, E. Martín-Del-Campo-Mena, H. G. Acosta-Mesa, N. Pérez-Castro, A. Guerra-Hernández, G. Hoyos-Rivera, and R.-E. Barrientos-Martínez. Evaluation of the diagnostic power of thermography in breast cancer using bayesian network classifiers. *Computational and Mathematical Methods in Medicine*, 2013:1–10, 2013.

[22] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(24):311–338, 2000.

[23] K. Deb. *Optimization for Engineering Design - Algorithms and Examples, Second Edition*. PHI Learning Private Limited, 2012.

[24] J. Derrac, S. García, D. Molina, and F. Herrera. 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, 2011.

[25] M. du Plessis. *Adaptive Multi-Population Differential Evolution for Dynamic Environments*. PhD thesis, Faculty of Engineering, Built Environment and Information Technology, University of Pretoria, April 2012.

[26] P. Filipiak and P. Lipinski. *Univariate Marginal Distribution Algorithm with Markov Chain Predictor in Continuous Dynamic Environments*, pages 404–411. Springer International Publishing, Cham, 2014.

[27] C. A. Floudas, C. S. Adjiman, and P. M. Pardalos. Handbook of test problems in local and global optimization. 1999.

[28] C. A. Floudas, C. S. Adjiman, and P. M. Pardalos. *Handbook of test problems in local and global optimization*. Nonconvex optimization and its applications. Kluwer Academic Publ., Dordrecht, Boston, London, 1999.

[29] C. K. Goh and K. C. Tan. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 13(1):103–127, Feb 2009.

[30] J. Grefenstette. Genetic algorithms for changing environments. In *Parallel Problem Solving from Nature 2*, pages 137–144. Elsevier, 1992.

[31] S. Hernandez, G. Leguizamon, and E. Mezura-Montes. A hybrid version of differential evolution with two differential mutation operators applied by stages. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 2895–2901, 2013.

[32] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975.

[33] W. Hordijk. A measure of landscapes. *Evol. Comput.*, 4(4):335–360, Dec. 1996.

[34] M. I. Review all currently known publications on approaches which solve the moving peaks problem. Technical report, Swinburne University of Technology, Melbourne, Australia, 2007.

[35] P. Ioannou, A. Chassiakos, H. Jula, and U. R. Dynamic optimization of cargo movement by trucks in metropolitan area with adjacent ports. Technical report, METRANS Transportation Center, University of Southern California, Los Angeles, CA 90089, USA, 2002.

[36] B. J. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[37] S. Jiang and S. Yang. Evolutionary dynamic multiobjective optimization: Benchmarks and algorithm comparisons. *IEEE Transactions on Cybernetics*, PP(99):1–14, 2016.

[38] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, June 2005.

[39] S. Kauffman and S. Levin. Towards a general theory of adaptive walks on rugged landscapes. *Journal of Theoretical Biology*, 128(1):11 – 45, 1987.

[40] S. A. Kauffman. *The origins of order : self organization and selection in evolution*. Oxford university press, New York, 1993.

[41] O. Kramer. *Biased Mutation for Evolution Strategies*, pages 51–80. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[42] J. Lewis, E. Hart, and G. Ritchie. *A comparison of dominance mechanisms and simple mutation on non-stationary problems*, pages 139–148. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[43] C. Li, T. T. Nguyen, M. Yang, S. Yang, and S. Zeng. Multi-population methods in unconstrained continuous dynamic environments: The challenges. *Information Sciences*, 296:95 – 118, 2015.

[44] C. Li, S. Yang, and M. Yang. An adaptive multi-swarm optimizer for dynamic optimization problems. *Evol. Comput.*, 22(4):559–594, Dec. 2014.

[45] X. Li, J. Branke, and T. Blackwell. Particle swarm with speciation and adaptation in a dynamic environment. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 51–58, New York, NY, USA, 2006. ACM.

[46] J. J. Liang, T. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C. A. Coello Coello, and K. Deb. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. Technical report, Nanyang Technological University, Singapore, Singapure, December 2005.

[47] R. Liu, Y. Chen, W. Ma, C. Mu, and L. Jiao. A novel cooperative coevolutionary dynamic multiobjective optimization algorithm using a new predictive model. *Soft Computing*, 18(10):1913–1929, 2014.

[48] M. López-Ibáñez and T. Stützle. Automatically improving the anytime behaviour of optimisation algorithms. Technical Report TR/IRIDIA/2012-012, IRIDIA, Université Libre de Bruxelles, Belgium, May 2012. Published in European Journal of Operations Research [73].

[49] M. G. Martínez-Peñaloza and E. Mezura-Montes. Immune generalized differential evolution for dynamic multiobjective optimization problems. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 1918–1925, May 2015.

[50] K. Mertens, T. Holvoet, and Y. Berbers. The dyncoaa algorithm for dynamic constraint optimization problems. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, pages 1421–1423, New York, NY, USA, 2006. ACM.

[51] P. Merz. Advanced fitness landscape analysis and the performance of memetic algorithms. *Evol. Comput.*, 12(3):303–325, Sept. 2004.

[52] E. Mezura-Montes, editor. *Constraint-Handling in Evolutionary Optimization*, volume 198 of *Studies in Computational Intelligence*. Springer-Verlag, 2009.

[53] E. Mezura-Montes and C. A. C. Coello. Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011.

[54] E. Mezura-Montes, M. E. Miranda-Varela, and R. del Carmen Gómez-Ramón. Differential evolution in constrained numerical optimization. an empirical study. *Information Sciences*, 180(22):4223–4262, 2010.

[55] Z. Michalewicz and G. Nazhiyath. Genocop iii: a co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In *Evolutionary Computation, 1995., IEEE International Conference on*, volume 2, pages 647–651 vol.2, Nov 1995.

[56] Z. Michalewicz and M. Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[57] R. W. Morrison. *Designing Evolutionary Algorithms for Dynamic Environments*. SpringerVerlag, 2004.

[58] R. Mukherjee, S. Debchoudhury, and D. Swagatam. Modified differential evolution with locality induced genetic operators for dynamic optimization. *European Journal of Operational Research*, 253(2):337 – 355, 2016.

[59] K. P. Ng and K. C. Wong. A new diploid scheme and dominance change mechanism for non-stationary function optimization. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 159–166, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[60] T. Nguyen. A proposed real-valued dynamic constrained benchmark set. Technical report, School Comput. Sci., Univ. Birmingham, Birmingham, U.K., 2008.

[61] T. Nguyen. *Continuous Dynamic Optimisation Using Evolutionary Algorithms*. PhD thesis, School of Computer Science The University of Birmingham, October 2010.

[62] T. Nguyen, S. Yang, and J. Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, 6(0):1 – 24, 2012.

[63] T. Nguyen and X. Yao. Detailed experimental results of ga, riga, hyperm and ga+repair on the g24 set of benchmark problems. Technical report, School Comput. Sci., Univ. Birmingham, Birmingham, U.K., 2010. available at: http://www.staff.livjm.ac.uk /enrtngu1/Papers/DCOP fulldata.pdf.

[64] T. Nguyen and X. Yao. Continuous dynamic constrained optimization: The challenges. *IEEE Transactions on Evolutionary Computation*, 16(6):769–786, 2012.

[65] T. Nguyen and X. Yao. Dynamic time-linkage evolutionary optimization: Definitions and potential solutions. In E. Alba, A. Nakib, and P. Siarry, editors, *Metaheuristics for Dynamic Optimization*, volume 433 of *Studies in Computational Intelligence*, pages 371–395. Springer Berlin Heidelberg, 2013.

[66] T. Nguyen and X. Yao. Evolutionary optimization on continuous dynamic constrained problems - an analysis. In S. Yang and X. Yao, editors, *Evolutionary Computation for Dynamic Optimization Problems*, volume 490 of *Studies in Computational Intelligence*, pages 193–217. Springer Berlin Heidelberg, 2013.

[67] T. T. Nguyen, S. Yang, J. Branke, and X. Yao. *Evolutionary Computation for Dynamic Optimization Problems*, chapter Evolutionary Dynamic Optimization: Methodologies, pages 39–64. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[68] T. T. Nguyen and X. Yao. Benchmarking and solving dynamic constrained problems. In *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pages 690–697, 2009.

[69] K. Pal, C. Saha, and S. Das. Differential evolution and offspring repair method based dynamic constrained optimization. In B. Panigrahi, P. Suganthan, S. Das, and S. Dash, editors, *Swarm, Evolutionary, and Memetic Computing*, volume 8297 of *Lecture Notes in Computer Science*, pages 298–309. Springer International Publishing, 2013.

[70] K. Pal, C. Saha, S. Das, and C. Coello-Coello. Dynamic constrained optimization with offspring repair based gravitational search algorithm. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 2414–2421, 2013.

[71] H. Pekdemir and H. R. Topcuoglu. Enhancing fireworks algorithms for dynamic optimization problems. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 4045–4052, July 2016.

[72] K. Price, R. Storn, and J. Lampinen. *Differential Evolution A Practical Approach to Global Optimization*. Natural Computing Series. Springer-Verlag, 2005.

[73] A. Radulescu, M. López-Ibáñez, and T. Stützle. Automatically improving the anytime behaviour of multiobjective evolutionary algorithms. In R. Purshouse, P. Fleming, C. M. Fonseca, S. Greco, and J. Shaw, editors, *Evolutionary Multi-Criterion Optimization*, volume 7811 of *Lecture Notes in Computer Science*, pages 825–840. Springer Berlin Heidelberg, 2013.

[74] S. S. Rao. *Optimization: Theory and Applications*. Wiley, New York, 1984.

[75] E. Rashedi, H. Nezamabadi, and S. Saryazdi. Gsa: A gravitational search algorithm. *Information Sciences*, 179(13):2232 – 2248, 2009.

[76] H. Richter. Change detection in dynamic fitness landscapes: An immunological approach. In *Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 719–724, Dec 2009.

[77] H. Richter. Detecting change in dynamic fitness landscapes. In *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pages 1613–1620, 2009.

[78] H. Richter. *Evolutionary Optimization and Dynamic Fitness Landscapes*, pages 409–446. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[79] H. Richter. *Memory Design for Constrained Dynamic Optimization Problems*, pages 552–561. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[80] H. Richter. *Evolutionary Computation for Dynamic Optimization Problems*, chapter Dynamic Fitness Landscape Analysis, pages 269–297. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[81] H. Richter and S. Yang. *Memory Based on Abstraction for Dynamic Fitness Functions*, pages 596–605. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[82] H. Richter and S. Yang. Learning behavior in abstract memory schemes for dynamic optimization problems. *Soft Computing*, 13(12):1163–1173, 2009.

[83] M. Rocha, J. Neves, A. C. Veloso, E. C. Ferreira, and I. Rocha. *Adaptive and Natural Computing Algorithms: Proceedings of the International Conference in Coimbra, Portugal, 2005*, chapter Evolutionary Algorithms for Static and Dynamic Optimization of Fed-batch Fermentation Processes, pages 288–291. Springer Vienna, Vienna, 2005.

[84] P. Rohlfshagen and X. Yao. Attributes of dynamic combinatorial optimisation. In X. Li, M. Kirley, M. Zhang, D. Green, V. Ciesielski, H. Abbass, Z. Michalewicz, T. Hendtlass, K. Deb, K. Tan, J. Branke, and Y. Shi, editors, *Simulated Evolution and Learning*, volume 5361 of *Lecture Notes in Computer Science*, pages 442–451. Springer Berlin Heidelberg, 2008.

[85] P. Rohlfshagen and X. Yao. *Evolutionary Computation for Dynamic Optimization Problems*, chapter Evolutionary Dynamic Optimization: Challenges and Perspectives, pages 65–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[86] M. Schlegel, , and W. Marquardt. Adaptive switching structure detection for the solution of dynamic optimization problems. *Industrial & Engineering Chemistry Research*, 45(24):8083–8094, 2006.

[87] A. Sharma and D. Sharma. *Neural Information Processing: 19th International Conference, ICONIP 2012, Doha, Qatar, November 12-15, 2012, Proceedings, Part I*, chapter ICHEA - A Constraint Guided Search for Improving Evolutionary Algorithms, pages 269–279. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[88] A. Sharma and D. Sharma. *Neural Information Processing: 19th International Conference, ICONIP 2012, Doha, Qatar, November 12-15, 2012, Proceedings, Part III*, chapter Solving Dynamic Constraint Optimization Problems Using ICHEA, pages 434–444. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[89] H. Singh, A. Isaacs, T. Nguyen, T. Ray, and X. Yao. Performance of infeasibility driven evolutionary algorithm (idea) on constrained dynamic single objective optimization problems. In *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pages 3127–3134, 2009.

[90] J. Tavares, F. B. Pereira, and E. Costa. Multidimensional knapsack problem: A fitness landscape analysis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(3):604–616, June 2008.

[91] K. Trojanowski and Z. Michalewicz. Searching for optima in non-stationary environments. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, page 1850 Vol. 3, 1999.

[92] Y. Umenai, F. Uwano, Y. Tajima, M. Nakata, H. Sato, and K. Takadama. A modified cuckoo search algorithm for dynamic optimization problems. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 1757–1764, July 2016.

[93] F. Vavak, T. C. Fogarty, and K. Jukes. *A genetic algorithm with variable range of local search for tracking changing environments*, pages 376–385. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

[94] F. Vavak, K. Jukes, and T. C. Fogarty. Learning the local search range for genetic optimisation in nonstationary environments. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 355–360, Apr 1997.

[95] F. Vavak, K. Jukes, and T. C. Fogarty. Learning the local search range for genetic optimisation in nonstationary environments. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 355–360, Apr 1997.

[96] Y. Wang and M. Wineberg. Estimation of evolvability genetic algorithm and dynamic environments. *Genetic Programming and Evolvable Machines*, 7(4):355–382, 2006.

[97] Y. G. Woldesenbet and G. G. Yen. Dynamic evolutionary algorithm with variable relocation. *IEEE Transactions on Evolutionary Computation*, 13(3):500–513, June 2009.

[98] S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of the Sixth International Congress of Genetics*, 1:356–66, 1932.

[99] H. X. and E. R. C. Adaptive particle swarm optimization: detection and response to dynamic systems. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 2, pages 1666–1670, 2002.

[100] S. Yang. *Associative Memory Scheme for Genetic Algorithms in Dynamic Environments*, pages 788–799. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[101] S. Yang. A comparative study of immune system based genetic algorithms in dynamic environments. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 1377–1384, New York, NY, USA, 2006. ACM.

[102] S. Yang, T. Nguyen, and C. Li. Evolutionary dynamic optimization: Test and evaluation environments. In S. Yang and X. Yao, editors, *Evolutionary Computation for Dynamic Optimization*

Problems, volume 490 of *Studies in Computational Intelligence*, pages 3–37. Springer Berlin Heidelberg, 2013.

[103] S. Yang and X. Yao. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Computing*, 9(11):815–834, 2005.

[104] S. Yang and X. Yao. Population-based incremental learning with associative memory for dynamic environments. *IEEE Transactions on Evolutionary Computation*, 12(5):542–561, Oct 2008.

[105] X. Yu and X. Wu. A multi-point local search algorithm for continuous dynamic optimization. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 2736–2743, July 2016.

[106] W. Zhang, G. G. Yen, and X. Wang. An immune inspired framework for optimization in dynamic environment. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 1800–1807, July 2016.