Introducción a la Computación Evolutiva

Carlos A. Coello Coello

carlos.coellocoello@ccinvestav.mx
CINVESTAV-IPN
Evolutionary Computation Group (EVOCINV)
Departamento de Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D.F. 07360, MEXICO

Clase 10

Tipos de Funciones de Penalización

Los tipos más comunes de funciones de penalización adoptadas con los algoritmos genéticos son las siguientes:

- Pena de muerte
- Penalización estática
- Penalización dinámica
- Penalización adaptativa
- Otros enfoques:
  - La formulación de aptitud auto-adaptativa (*Self-Adaptive Fitness Formulation*)
  - ASCHEA
  - Jerarquización estocástica (*Stochastic Ranking*)

Tipos de Funciones de Penalización

Pena de Muerte

El rechazar soluciones infactibles (una técnica denominada "pena de muerte") es probablemente la técnica más simple de manejo de restricciones.

Evidentemente, también es una técnica muy eficiente, porque cuando una solución es infactible, simplemente generamos otra de forma aleatoria para reemplazarla.

Evidentemente, no se requieren cálculos de ningún tipo para estimar el grado de infactibilidad de una solución.

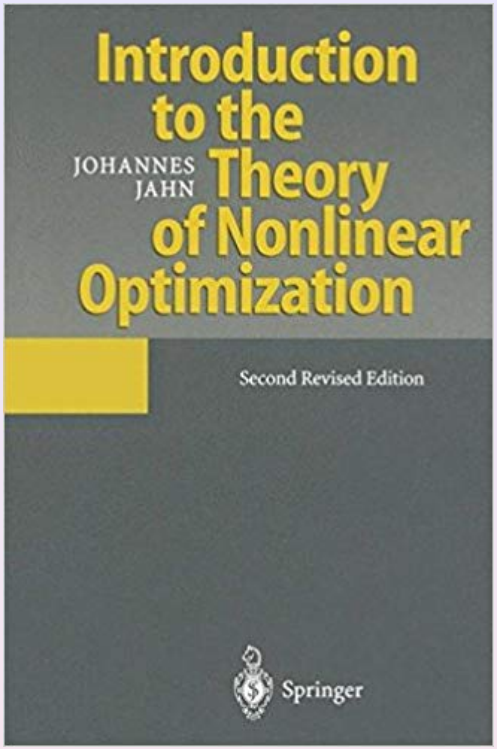
Tipos de Funciones de Penalización

Pena de Muerte

Esta técnica presenta varias limitantes:

- En general, su uso no se recomienda, excepto en problemas en los que la región factible sea bastante grande con respecto al tamaño total del espacio de búsqueda.
- No utiliza información de las soluciones infactibles.
- La búsqueda se puede "estancar" cuando la región factible es muy pequeña con respecto al tamaño de todo el espacio de búsqueda.
- Algunos investigadores han utilizado una variante de esta técnica que asigna una aptitud de cero a las soluciones infactibles. Esta versión parece funcionar bien en problemas con regiones factibles grandes.

Tipos de Funciones de Penalización



Penalización Estática

Se denominan así las funciones de penalización en las cuales los factores de penalización no dependen de la generación actual y que, por tanto, permanenecen constantes durante todo el proceso evolutivo.

Tipos de Funciones de Penalización

Penalización Estática

Un ejemplo de este tipo de técnica es el esquema propuesto por Homaifar, Lai y Qi [1994], en el cual se definen niveles de violación de las restricciones (y factores de penalización asociados a dichos niveles):

$$\text{fitness}(\vec{x}) = f(\vec{x}) + \sum_{i=1}^{m} \left(R_{k,i} \times \max [0, g_i(\vec{x})]^2 \right) \qquad (1)$$

donde $R_{k,i}$ son los factores de penalización utilizados, m es el número total de restricciones, $f(\vec{x})$ es la función objetivo sin penalizar, y $k = 1, 2, \ldots, l$, donde l es el número de niveles de violación de las restricciones, el cual es definido por el usuario.

A. Homaifar, S.H.Y. Lai and X. Qi, "**Constrained Optimization via Genetic Algorithms**", *Simulation*, Vol. 62, No. 4, pp. 242–254, 1994.

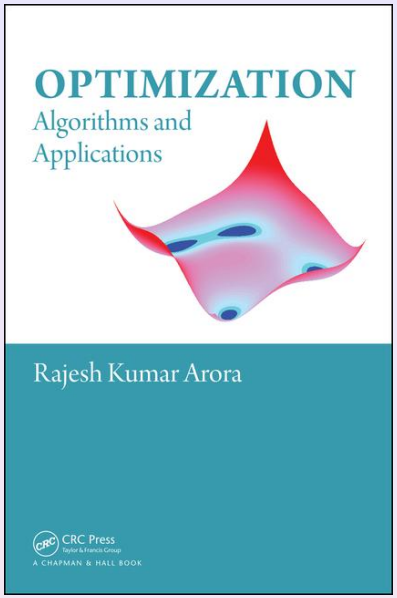
Tipos de Funciones de Penalización

Penalización Estática

Sus críticas principales son las siguientes:

- Normalmente, no es una buena idea mantener constantes los factores de penalización durante todo el proceso evolutivo.
- Los factores de penalización, en general, dependen del problema.
- Aunque esta técnica es simple, en algunos casos (como por ejemplo, en la propuesta de Homaifar, Lai and Qi [1994]), el usuario podría tener que definir un gran número de factores de penalización.

Tipos de Funciones de Penalización



Penalización Dinámica

En esta categoría se consideran las funciones de penalización en las cuales se utiliza el número de generación actual para calcular los factores de penalización correspondientes (normalmente, los factores de penalización se calculan de tal forma que involucren el tiempo—o sea, la generación actual).

Penalización Dinámica

Un ejemplo de este tipo de técnica es la propuesta de Joines y Houck [1994] en la que los individuos se evalúan (en la generación t) usando:

$$\text{fitness}(\vec{x}) = f(\vec{x}) + (C \times t)^{\alpha} \times SVC(\beta, \vec{x}) \quad (2)$$

donde C, α y β son constantes definidas por el usuario (los autores usaron $C = 0.5$, $\alpha = 1$ or 2, y $\beta = 1$ or 2).

J. Joines and C. Houck, "**On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs**", in *Proceedings of the first IEEE Conference on Evolutionary Computation*, pp. 579–584, IEEE Press, Orlando, Florida, USA, 1994.

Penalización Dinámica

$SVC(\beta, \vec{x})$ se define de la forma siguiente:

$$SVC(\beta, \vec{x}) = \sum_{i=1}^{n} D_i^{\beta}(\vec{x}) + \sum_{j=1}^{p} D_j(\vec{x}) \tag{3}$$

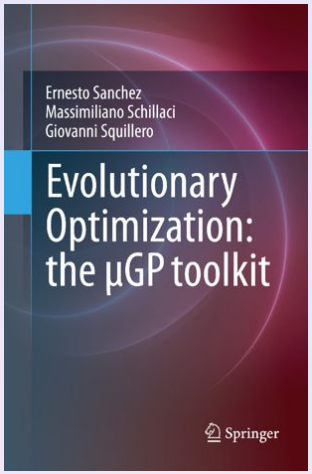
y

$$D_i(\vec{x}) = \begin{cases} 0 & g_i(\vec{x}) \leq 0 \\ |g_i(\vec{x})| & \text{de lo contrario} \end{cases} \qquad 1 \leq i \leq n \tag{4}$$

$$D_j(\vec{x}) = \begin{cases} 0 & -\epsilon \leq h_j(\vec{x}) \leq \epsilon \\ |h_j(\vec{x})| & \text{de lo contrario} \end{cases} \qquad 1 \leq j \leq p \tag{5}$$

Esta función dinámica incrementa la penalización conforme se incrementa el número de generación.

Tipos de Funciones de Penalización



Penalización Dinámica

Sus críticas principales son las siguientes:

- Definir buenas funciones de penalización dinámicas es tan difícil como definir buenas funciones de penalización estáticas.
- Sin embargo, en la práctica, las funciones de penalización dinámicas suelen funcionar mejor que las estáticas.
- Las funciones de penalización dinámica suelen ser una buena opción sobre todo en problemas con un número arbitrario de restricciones.

Tipos de Funciones de Penalización

Penalización Adaptativa

Bean y Hadj-Alouane [1992] desarrollaron un método que usa una función de penalización que recibe retroalimentación del proceso de búsqueda.

Cada individuo es evaluado usando la fórmula siguiente:

$$\text{fitness}(\vec{x}) = f(\vec{x}) + \lambda(t) \left[\sum_{i=1}^{n} g_i^2(\vec{x}) + \sum_{j=1}^{p} |h_j(\vec{x})| \right] \quad (6)$$

donde $\lambda(t)$ se actualiza a cada generación t.

James C. Bean and Atidel Ben Hadj-Alouane, "**A Dual Genetic Algorithm for Bounded Integer Programs**", Technical Report No. TR 92-53, Department of Industrial and Operations Engineering, The University of Michigan, USA, 1992.

Penalización Adaptativa

$\lambda(t)$ se actualiza de acuerdo al esquema siguiente:

$$\lambda(t+1) = \begin{cases} (1/\beta_1) \cdot \lambda(t), & \text{si se cumple el caso \#1} \\ \beta_2 \cdot \lambda(t), & \text{si se cumple el caso \#2} \\ \lambda(t), & \text{de lo contrario,} \end{cases} \quad (7)$$

en donde los casos #1 y #2 denotan situaciones en las que el mejor individuo en las útimas k generaciones fue siempre (caso #1) o nunca (caso #2) factible. Adicionalmente, $\beta_1, \beta_2 > 1$, $\beta_1 > \beta_2$, y $\beta_1 \neq \beta_2$ (para evitar ciclarse).

Tipos de Funciones de Penalización

Penalización Adaptativa

En otras palabras, el factor de penalización, $\lambda(t+1)$ para la generación $t+1$ se decrementa si todos los mejores individuos de las últimas k generaciones fueron factibles. Esto se debe a que en este caso, queremos regresar a la frontera entre la región factible y la infactible.

Por el contrario, si todos los mejores individuos de las últimas k generaciones fueron infactibles, $\lambda(t+1)$ se incrementa, porque no hemos llegado todavía a la región factible.

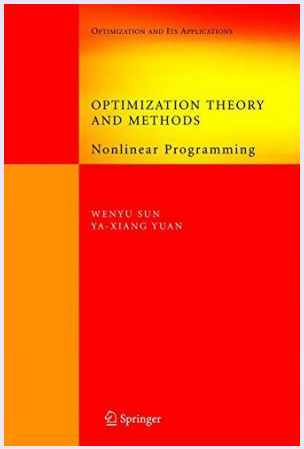
El caso deseable es, por tanto, aquel en el que algunas veces, en las últimas k generaciones, el mejor individuo es factible y en otros es infactible.

Penalización Adaptativa

Algunas críticas en torno a esta técnica son las siguientes:

- Establecer el valor de la brecha generacional (k) que requiere este método no luce completamente trivial.
- Este mecanismo parece ser una opción más "inteligente" para cambiar los factores de penalización que el esquema dinámico, pues se usa directamente información obtenida del proceso de búsqueda.
- Un aspecto interesante de esta técnica es que se considera igualmente indeseable el tener una población en la que todos los individuos son infactibles, como una en donde todos los individuos son factibles. Como veremos más adelante, varias técnicas modernas de manejo de restricciones le ponen mucha atención a este aspecto.

Funciones de Penalización

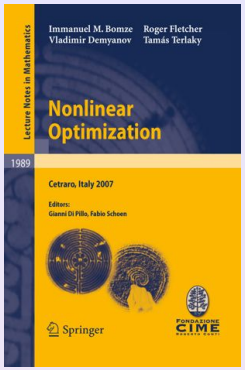


Temas Centrales

El problema principal con las funciones de penalización es que el factor de penalización "ideal" no puede conocerse *a priori* para un problema arbitrario.

Adicionalmente, si el factor de penalización adoptado es demasiado alto o demasiado bajo, la función de penalización no permitirá realizar adecuadamente el proceso de optimización.

Funciones de Penalización



Otros Enfoques

Existen cuatro técnicas de manejo de restricciones más modernas que también se basan en el uso de una función de penalización, pero que son altamente competitivas:

- La formulación de aptitud auto-adaptativa (*Self-Adaptive Fitness Formulation*)
- ASCHEA
- Jerarquización estocásticas (*Stochastic Ranking*)
- El Método α Restringido (α *Constrained Method*)

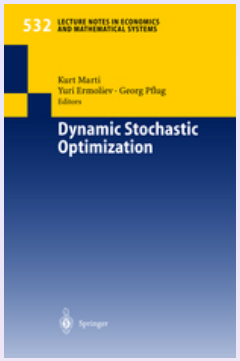
Self-Adaptive Fitness Formulation

Fue propuesta por Farmani y Wright [2003] y se basa en una función de penalización que se aplica en 3 pasos:

1. Se calcula la suma de violación de restricciones para cada individuo.
2. Se identifica a la peor y a la mejor solución en la población actual.
3. Se aplica una penalización en dos partes:
   - Se aplica sólo si una o más soluciones factibles tienen un mejor valor de la función objetivo que la mejor solución obtenida hasta el momento. La idea en este caso es incrementar la aptitud de las soluciones infactibles.
   - Incrementar la aptitud de las soluciones infactibles de tal forma que se favorezcan las soluciones que son casi factibles y tienen también un buen valor de la función objetivo.

Raziyeh Farmani and Jonathan A. Wright, "**Self-Adaptive Fitness Formulation for Constrained Optimization**", *IEEE Transactions on Evolutionary Computation*, Vol. 7, No. 5, pp. 445–455, October 2003.

Funciones de Penalización



Self-Adaptive Fitness Formulation

El factor de penalización se define en términos tanto de la mejor como de la peor solución.

Los autores adoptaron un algoritmo genético con codificación binaria (usando códigos de Gray) y selección de ruleta.

Se obtuvieron buenos resultados, pero no mejores que los de los algoritmos del estado del arte (p.ej., jerarquización estocástica).

Funciones de Penalización

Self-Adaptive Fitness Formulation

Esta técnica requirió un número elevado de evaluaciones de la función objetivo (1,400,000) en los problemas de prueba típicamente utilizados en esta área.

El principal punto de venta de esta técnica es que no requiere ningún parámetro adicional definido por el usuario. Adicionalmente, su implementación es relativamente simple.

Se hace notar que existen otras funciones de penalización auto-adaptativas en la literatura especializada (ver por ejemplo [Tessema & Yen, 2006]).

Biruk Tessema and Gary G. Yen, "**A Self Adaptive Penalty Function Based Algorithm for Constrained Optimization**", in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pp. 950–957, IEEE Press, Vancouver, Canada, 2006.

ASCHEA

El *Adaptive Segregational Constraint Handling Evolutionary Algorithm* (ASCHEA) fue propuesto por Hamida y Schoenauer [2000].

Esta propuesta utiliza una estrategia evolutiva a la que incorpora tres componentes principales:

- Una función de penalización adaptativa.
- Un operador de recombinación guiado por las restricciones.
- Un operador de selección "segregacional".

S. Ben Hamida and Marc Schoenauer, "**An Adaptive Algorithm for Constrained Optimization Problems**", in M. Schoenauer, K. Deb, G. Rudolph, X. Yao and E. Lutton (Editors), *Parallel Problem Solving from Nature 2000 (PPSN VI)*, pp. 529–533, Springer-Verlag, Lecture Notes in Computer Science Vol. 1917, Paris, France, September 2000.

ASCHEA

La **penalización adaptativa** que usaron es la siguiente:

$$fitness(\vec{x}) = \begin{cases} f(\vec{x}) & \text{si la solución es factible} \\ f(\vec{x}) - penal(\vec{x}) & \text{de lo contrario} \end{cases} \quad (8)$$

donde

$$\text{penal}(\vec{x}) = \alpha \sum_{j=1}^{q} g_j^+(\vec{x}) + \alpha \sum_{j=q+1}^{m} |h_j(\vec{x})| \quad (9)$$

donde $g_j^+(\vec{x})$ es la parte positiva de $g_j(\vec{x})$ y α es el factor de penalización adoptado para todas las restricciones.

ASCHEA

El factor de penalización se adapta con base en la tasa deseable de soluciones factibles (con respecto al total de la población) a la que se denomina τ_{target}. Este valor se compara con la tasa real en la generación t, a la que denominan τ_t:

$$\begin{aligned} \text{if}(\tau_t > \tau_{target}) \quad & \alpha(t+1) = \alpha(t)/fact \\ \text{else} \quad & \alpha(t+1) = \alpha(t) * fact \end{aligned}$$

donde $fact > 1$ y τ_{target} son parámetros definidos por el usuario y

$$\alpha(0) = \left| \frac{\sum_{i=1}^{n} f_i(\vec{x})}{\sum_{i=1}^{n} V_i(\vec{x})} \right| * 1000 \qquad (10)$$

donde $V_i(\vec{x})$ es la suma de violación de restricciones del individuo i.

Funciones de Penalización

ASCHEA

El operador de **Recombinación guiada por las restricciones** combina una solución infactible con una factible cuando hay pocas soluciones factibles en la población, de acuerdo a τ_{target}.

Si $\tau_t > \tau_{target}$, entonces la recombinación se realiza de la forma tradicional (es decir, sin considerar la factibilidad de las soluciones a recombinarse).

El operador de **Selección Segregacional** busca definir una tasa de soluciones factibles que serán parte de la siguiente generación, a la que se denomina τ_{select}. El resto de los individuos son seleccionados de la forma tradicional, con base en su aptitud penalizada. Evidentemente, τ_{select} es otro parámetro definido por el usuario.

Funciones de Penalización

ASCHEA

En su versión más reciente [2002], ASCHEA usa un factor de penalización para cada restricción, a fin de permitir la definición de penalizaciones más precisas.

Esta versión incorpora también nichos para mantener diversidad (aunque esto incrementa el número de parámetros definidos por el usuario).

Esta técnica realiza un número elevado de evaluaciones de la función de aptitud (1,500,000).

Sana Ben Hamida and Marc Schoenauer, "**ASCHEA: New Results Using Adaptive Segregational Constraint Handling**", in *2002 IEEE Congress on Evolutionary Computation (CEC'2002)*, Vol. 1, pp. 884–889, IEEE Press, May 2002.

Funciones de Penalización

Jerarquización Estocástica

Esta técnica fue propuesta por Runarsson y Yao [2000] y sigue siendo una de las más competitivas en el área.

La jerarquización estocástica (*stochastic ranking*) se basa en una estrategia evolutiva multi-miembro que utiliza una función de penalización y un mecanismo de selección basado en un proceso de jerarquización.

Thomas P. Runarsson and Xin Yao, "**Stochastic Ranking for Constrained Evolutionary Optimization**", *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 3, pp. 284–294, September 2000.

Funciones de Penalización

Jerarquización Estocástica

La idea fundamental de esta técnica es intentar balancear la influencia de la función objetivo con la de la función de penalización cuando asignamos aptitud a un individuo.

Un aspecto interesante de esta técnica es que no requiere la definición de un factor de penalización.

Sin embargo, requiere un parámetro definido por el usuario, llamado P_f, el cual determina el balance entre la función objetivo y la función de penalización.

Funciones de Penalización

Begin
  For i=1 **to** N
    For j=1 **to** P-1
      u=random(0,1)
      If $\left(\phi(I_j) = \phi(I_{j+1}) = 0\right)$ or $(u < P_f)$
        If $\left(f(I_j) > f(I_{j+1})\right)$
          swap(I_j,I_{j+1})
      Else
        If $\left(\phi(I_j) > \phi(I_{j+1})\right)$
          swap(I_j,I_{j+1})
    End For
    If no swap is performed
      break
  End For
End

Funciones de Penalización

Jerarquización Estocástica

La población se ordena usando un algoritmo similar al de la burbuja (que ordena una lista con base en comparaciones por parejas).

Con base en el valor de P_f, la comparación de dos individuos adyacentes se realizará con base en la función objetivo.

El resto de las comparaciones se realizan con base en la suma de la violación de las restricciones.

Por ende, P_f introduce el componente "estocástico" al proceso de jerarquización, de tal forma que las soluciones puedan obtener una buena jerarquía aun si son infactibles.

Jerarquización Estocástica

El valor de P_f tiene un impacto importante en el desempeño de esta técnica. Sus autores encontraron empíticamente que $0.4 < P_f < 0.5$ produce los mejores resultados.

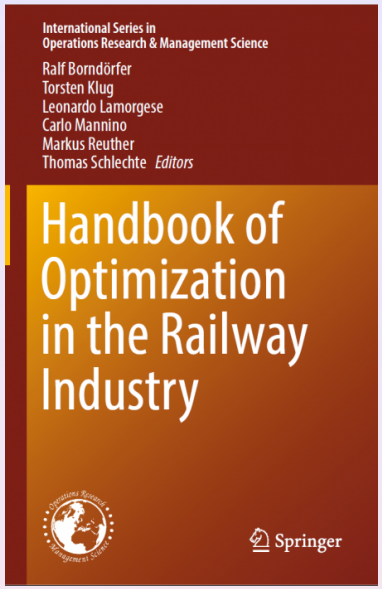
Los autores reportaron que los mejores resultados los obtuvieron realizando sólo 350,000 evaluaciones de las funciones objetivo, usando los problemas de prueba estándar del área.

La técnica es fácil de implementar.

En 2006, se propuso una versión mejorada de este algoritmo, en la cual se usan estrategias evolutivas y variación diferencial

Thomas Philip Runarsson, "**Approximate Evolution Strategy using Stochastic Ranking**", in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pp. 2760–2767, IEEE Press, Vancouver, Canada, July 2006.

Otras Técnicas



Hay dos técnicas más que discutiremos por su relevancia en el área:

- Una Estrategia Evolutiva Multi-Miembro Simple (*Simple Multimembered Evolution Strategy* (SMES))
- El Método α Restringido

Una Estrategia Evolutiva Multi-Miembro Simple

Mezura-Montes y Coello [2005] propusieron una técnica basada en una estrategia evolutiva $(\mu + \lambda)$.

En este caso, los individuos de la población se comparan usando los siguientes criterios (propuestos originalmente por [Deb, 2000]):

1. Entre las dos soluciones factibles, la que tenga la mayor aptitud es la ganadora.
2. Si una solución es factible y la otra es infactible, la solución factible gana.
3. Si las dos soluciones son infactibles, se prefiere la que tenga la menor suma de violación de restricciones.

Efrén Mezura-Montes and Carlos A. Coello Coello, "**A Simple Multimembered Evolution Strategy to Solve Constrained Optimization Problems**", *IEEE Transactions on Evolutionary Computation*, Vol. 9, No. 1, pp. 1–17, February 2005.

Una Estrategia Evolutiva Multi-Miembro Simple

Adicionalmente, esta técnica utiliza tres mecanismos principales:

1. **Mecanismo de Diversidad**: La solución infactible que está más cerca de volverse factible es retenida en la población, de tal forma que se recombine con las soluciones factibles.
2. **Recombinación Combinada**: Se usa recombinación panmítica, pero con una combinación de los operadores de recombinación discreta e intermedia.
3. **Longitud de Paso**: La longitud de paso inicial de la estrategia evolutiva se reduce de tal forma que se favorezcan los movimientos más finos en el espacio de búsqueda.

Otras Técnicas

Una Estrategia Evolutiva Multi-Miembro Simple

Esta técnica obtuvo resultados muy competitivos con respecto al método de jerarquización estocástica (*stochastic ranking*), los mapas homomorfos y ASCHEA.

Esta técnica sólo realiza 240,000 evaluaciones de la función objetivo.

En un artículo posterior, se incorporaron mecanismos similares al algoritmo de evolución diferencial, obteniéndose resultados todavía mejores que los de la estrategia evolutiva.

Esta técnica es fácil de implementar y bastante robusta.

El Método α Restringido

Este método fue propuesto por Takahama y Sasai [2004].

Su idea fundamental es definir un nivel de satisfacción para cada una de las restricciones de un problema.

La técnica utiliza ordenamiento lexicográfico combinado con una relajación de las restricciones.

La técnica de relajación de las restricciones facilita el manejo de restricciones de igualdad.

T. Takahama and S. Sakai, "**Constrained Optimization by α Constrained Genetic Algorithm (αGA)**", *Systems and Computers in Japan*, Vol. 35, No. 5, pp. 11–22, May 2004.

El Método α Restringido

En una versión posterior [Takahama and Sakai, 2005], esta técnica es acoplada con una versión modificada del método de búsqueda directa conocido como Nelder-Mead (también conocido como Simplex No Lineal).

Sus autores argumentan que el método de Nelder-Mead puede verse como un algoritmo evolutivo en el cual, por ejemplo, se usan los siguientes operadores de variación: reflexión, contracción y expansión.

Los autores también extendieron este método con un operador de mutación de límite, el uso de varias mallas (simplexes), y una modificación de los operadores tradicionales de este método, a fin de evitar que el algoritmo quede fácilmente atrapado en un óptimo local.

Tetsuyuki Takahama and Setsuko Sakai, "**Constrained Optimization by Applying the α Constrained Method to the Nonlinear Simplex with Mutations**", *IEEE Transactions on Evolutionary Computation*, Vol. 9, No. 5, pp. 437–451, October 2005.

El Método α Restringido

Esta técnica se validó usando un conjunto de 13 problemas de prueba.

Los resultados obtenidos por esta técnica se compararon con respecto a los de la jerarquización estocástica.

El número de evaluaciones de la función de aptitud realizados por el método varió de 290,000 a 330,000 en la mayoría de los casos.

Los resultados obtenidos fueron muy competitivos, aunque la técnica tuvo cierta sensibilidad a la variabilidad de sus parámetros.

El Método α Restringido

En un artículo posterior, Takahama y Sasai [2006] propusieron otra técnica para relajar las restricciones, pero usando un parámetro llamado ϵ.

En este caso, se usa evolución diferencial como el motor de búsqueda, así como un operador de mutación basado en gradientes.

Tetsuyuki Takahama and Setsuko Sakai, "**Constrained Optimization by the ϵ Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites**", in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pp. 308–315, IEEE Press, Vancouver, Canada, July 2006.

Representaciones y Operadores Especiales

Algunos investigadores han desarrollado esquemas de representación especiales para resolver ciertos problemas (particularmente difíciles) para los cuales un esquema de representación genérico (por ejemplo, la codificación binaria de un algoritmo genético tradicional) puede no resultar apropiado.

Debido al cambio de representación, es necesario diseñar operadores genéticos especiales que funcionen de manera análoga a los operadores que se usan tradicionalmente con una codificación binaria.

Un ejemplo de este tipo de técnica son las **llaves aleatorias** (*Random Keys*) [Bean, 1994].

James C. Bean, "**Genetics and random keys for sequencing and optimization**", *ORSA Journal on Computing*, Vol. 6, No. 2, pp. 154–160, 1994.

Otras Técnicas

Representaciones y Operadores Especiales

Un tipo de enfoque más interesante dentro de esta categoría es el de los denominados **Decodificadores**.

El énfasis en este tipo de técnicas es mapear cromosomas de la región infactible a la región factible del problema a resolverse.

En algunos casos, se han diseñado también operadores especiales para producir hijos que se encuentran en la frontera entre la región factible y la infactible.

Representaciones y Operadores Especiales

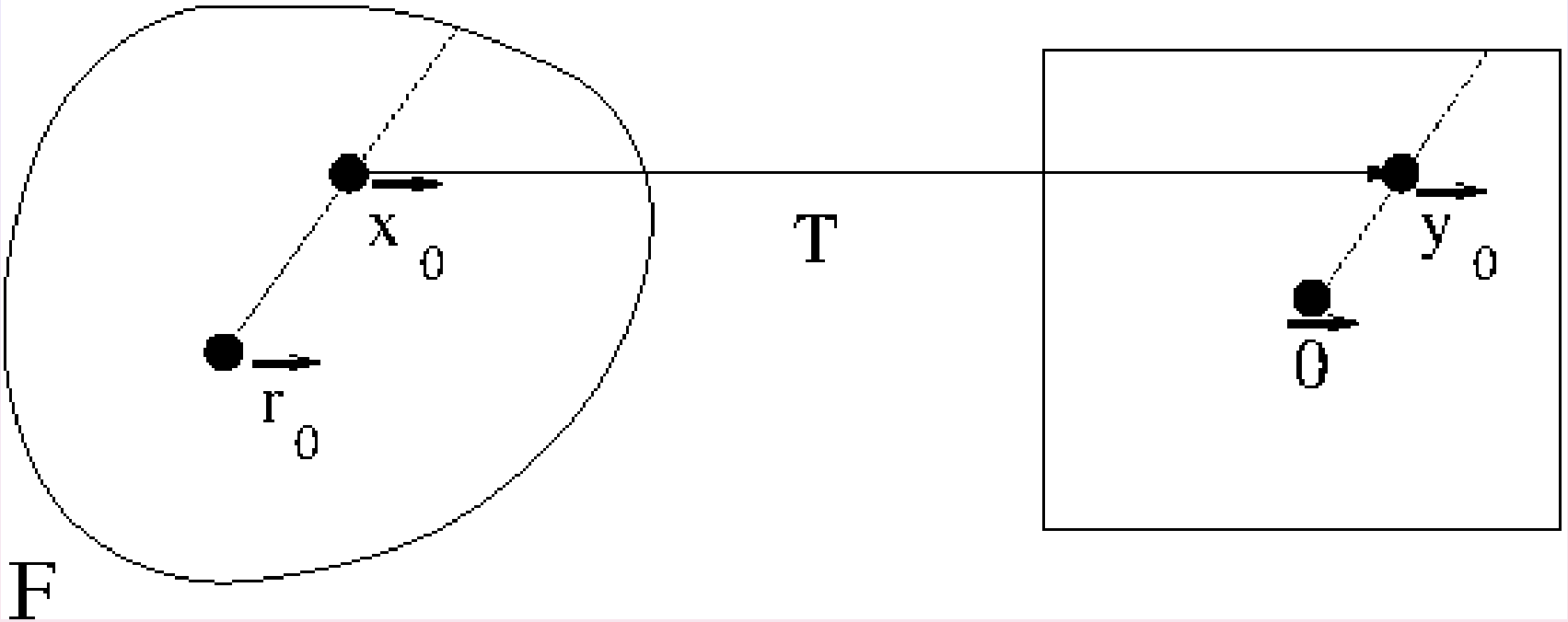
Una idea intrigante es la de transformar toda la región factible en una forma diferente que sea más fácil de explorar.

La técnica más importante dentro de este grupo es la denominada **mapas homomorfos** [Koziel & Michalewicz, 1999].

Esta técnica realiza un mapeo homomorfo entre un cubo *n* dimensional y un espacio de búsqueda factible (ya sea convexo o no convexo).

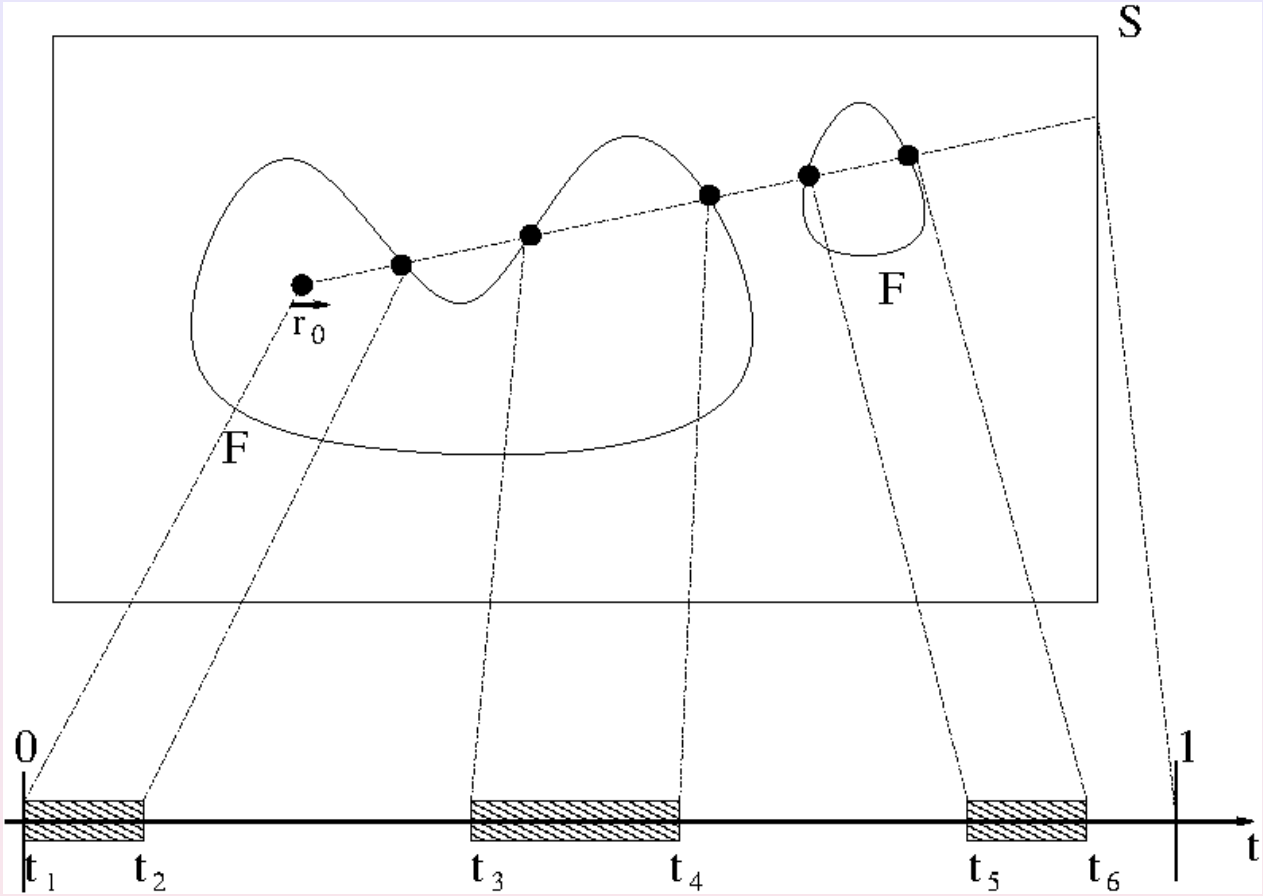
La idea principal de esta técnica es transformar el problema original en otra función (topológicamente equivalente) que sea más fácil de optimizar usando un algoritmo genético.

Slawomir Koziel and Zbigniew Michalewicz, "**Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization**", *Evolutionary Computation*, Vol. 7, No. 1, pp. 19–44, 1999.



Mapas Homomorfos

Este es el caso convexo.



Mapas Homomorfos

Este es el caso no convexo.

Mapas Homomorfos

En su época, el método de los Mapas Homomorfos fue la técnica de manejo de restricciones más competitiva que existía (hasta antes de la publicación de las Jerarquías Estocásticas).

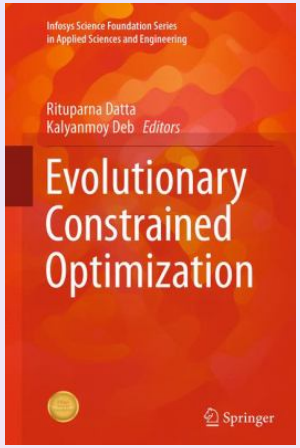
Sin embargo, la implementación de este algoritmo es compleja, y los experimentos reportados en el artículo requirieron un número elevado de evaluaciones de la función de aptitud (1,400,000).

Mapas Homomorfos

La versión de los mapas homomorfos para regiones factibles convexas es muy eficiente.

Sin embargo, la versión para regiones factibles no convexas requiere de un parámetro v y de un procedimiento de búsqueda binaria para encontrar la intersección de una línea con el borde de la región factible.

Algoritmos de Reparación



Este tipo de técnica consiste en proponer un procedimiento (o mecanismo) que permita transformar una solución infactible en una factible (es decir, "reparamos" al individuo infactible).

La versión reparada de un individuo puede ser usada sólo para fines de evaluación, o puede también reemplazar (con alguna probabilidad) al individuo original de la población (o sea, a la versión original del individuo reparado).

Algoritmos de Reparación

Liepins et al. [1990] mostraron, mediante una validación empírica de desempeño de un algoritmo genético en un conjunto diverso de problemas de optimización combinatoria con restricciones, que un algoritmo de reparación era capaz de superar a otros esquemas de manejo de restricciones tanto en velocidad de convergencia como en desempeño.

G.E. Liepins and Michael D. Vose, "**Representational Issues in Genetic Optimization**", *Journal of Experimental and Theoretical Computer Science*, Vol. 2, No. 2, pp. 4–30, 1990.

Algoritmos de Reparación

GENOCOP III [Michalewicz & Nazhiyath, 1995] utiliza también algoritmos de reparación.

La idea principal es adoptar el GENOCOP original [Michalewicz & Janikow, 1991] (el cual maneja sólo restricciones lineales) y extenderlo con dos poblaciones separadas, de tal forma que los resultados de una población influencian las evaluaciones de los individuos en la otra población.

Zbigniew Michalewicz and G. Nazhiyath, "**Genocop III: A co-evolutionary algorithm for numerical optimization with nonlinear constraints**", in David B. Fogel (Editor) *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, pp. 647–651, IEEE Press, Piscataway, New Jersey, USA, 1995.

Algoritmos de Reparación

La primera población consiste de los denominados puntos de búsqueda que satisfacen las restricciones lineales del problema. La factibilidad (en lo que se refiere a las restricciones lineales) de estos puntos se mantiene mediante operadores especializados.

La segunda población consiste de puntos de referencia factibles. Puesto que estos puntos de referencia ya son factibles, se evalúan directamente mediante la función objetivo, de tal forma que los puntos de búsqueda son "reparados" para su evaluación.

Algoritmos de Reparación

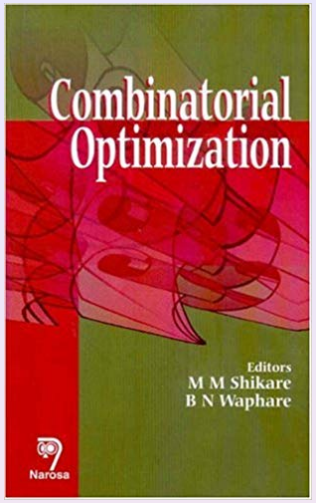
Xiao et al. [1997] usaron un algoritmo de reparación para transformar una ruta infactible de un robot que intenta moverse entre dos puntos en la presencia de obstáculos, de tal forma que la ruta se vuelva factible (es decir, libre de colisiones).

El algoritmo de reparación fue implementado a través de un conjunto de operadores genéticos cuidadosamente diseñados, que usa conocimiento del dominio para mover las soluciones infactibles a la región factible de manera eficiente.

Otros de los muchos autores que han usado algoritmos de reparación son Orvosh y Davis [1994], Mühlenbein [1992], Le Riche y Haftka [1994], y Tate & Smith [1995].

Jing Xiao, Zbigniew Michalewicz and Krzysztof Trojanowski, "**Adaptive Evolutionary Planner/Navigator for Mobile Robots**", *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 18–28, 1997.

Algoritmos de Reparación



No existen heurísticas estándar para el diseño de algoritmos de reparación: normalmente, es posible usar un algoritmo avaricioso (*greedy*) (o sea, un algoritmo de optimización que proceda a través de una serie de alternativas tomando la mejor decisión, determinada localmente, en cada punto de la serie), un algoritmo aleatorio o cualquier otra heurística que guíe el proceso de reparación.

Sin embargo, el éxito de esta técnica depende principalmente en la capacidad del usuario para proponer dicha heurística.

Algoritmos de Reparación

Otro aspecto interesante de esta técnica es que normalmente una solución infactible que es reparada es utilizada únicamente para calcular su aptitud.

Sin embargo, la versión reparada es regresada a la población sólo en ciertos casos (usando una cierta probabilidad).

La pregunta de si se reemplaza o no a la versión original del individuo por su versión reparada, se relaciona con la denominada evolución Lamarckiana, la cual supone que un individuo mejora durante su tiempo de vida y que las mejoras resultantes se codifican de vuelta en el cromosoma.

Algoritmos de Reparación

Algunos investigadores como Liepins et al. [1991] han adoptado el enfoque de **nunca** reemplazar (es decir, la versión reparada nunca se regresa a la población).

En tanto, otros autores, tales como Nakano [1991] han adoptado el enfoque de **siempre** reemplazar.

Orvosh y Davis [1994] reportaron que un reemplazo del 5% proporcionaba los mejores resultados para un algoritmo genético diseñado para resolver problemas de optimización combinatoria.

Michalewicz [1996], sin embargo, reportó que un reemplazo del 15% le resultó como la mejor opción para problemas de optimización numérica con restricciones no lineales.

Ryohei Nakano and Takeshi Yamada, "**Conventional Genetic Algorithm for Job Shop Scheduling**", in R.K. Belew and L.B. Booker (Editors), *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 474–479, Morgan Kaufmann Publishers, San Mateo, California, USA, 1991.

Algoritmos de Reparación

Cuando una solución infactible puede ser fácilmente (o al menos, a un bajo costo computacional) transformada en una solución factible, los algoritmos de reparación son una buena opción.

Sin embargo, esto no siempre es posible y, en algunos casos, los operadores de reparación pueden introducir un fuerte sesgo en la búsqueda, lo que daña el proceso evolutivos mismo [Smith & Tate, 1993].

Alice E. Smith and David M. Tate, "**Genetic Optimization using a Penalty Function**", in Stephanie Forrest (Editor), *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 499–503, Morgan Kaufmann Publishers, San Mateo, California, USA, July 1993.

Algoritmos de Reparación

Además, esta técnica depende del problema, puesto que debe diseñarse un algoritmo de reparación específico para cada problema en particular.

Sin embargo, en años recientes, ésta se ha vuelto un área relativamente activa de investigación (ver por ejemplo, [Salcedo-Sanz, 2009]).

Sancho Salcedo-Sanz, "**A Survey of Repair Methods Used as Constraint Handling Techniques in Evolutionary Algorithms**", *Computer Science Review*, Vol. 3, No. 3, pp. 175–192, 2009.

Separación de Restricciones y Objetivos

A diferencia de las funciones de penalización, con las que se combina el valor de la función objetivo con el de la violación de las restricciones para asignar aptitud, en este tipo de técnicas se manejan por separado los valores de las funciones objetivo y de las restricciones. Veremos a continuación ejemplos de este tipo de enfoque.

Coevolución

La coevolución es un esquema que usa dos poblaciones que interactúan entre sí y tienen encuentros periódicos a lo largo del proceso evolutivo [Paredis, 1994].

J. Paredis, "**Co-evolutionary Constraint Satisfaction**", in *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, pp. 46–55, Springer-Verlag, New York, USA, 1994.

Separación de Restricciones y Objetivos

Superioridad de Puntos Factibles

La idea de esta técnica es asignar siempre una aptitud más elevada a las soluciones factibles [Powell & Skolnick, 1993; Deb, 2000].

David Powell and Michael M. Skolnick, "**Using genetic algorithms in engineering design optimization with non-linear constraints**", in Stephanie Forrest (Editor) *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 424–431, Morgan Kaufmann Publishers, San Mateo, California, USA, July 1993.

Memoria Conductista

Schoenauer y Xanthakis [1993] propusieron satisfacer secuencialmente las restricciones de un problems.

Marc Schoenauer and Spyros Xanthakis, "**Constrained GA Optimization**", in Stephanie Forrest (Editor), *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 573–580, Morgan Kaufmann Publishers, San Mateo, California, USA, July 1993.

Separación de Restricciones y Objetivos

Uso de conceptos de optimización multi-objetivo

La idea principal en este caso es re-definir el problema original mono-objetivo con restricciones, en uno con $m + 1$ objetivos y sin restricciones, donde m es el número total de restricciones. El objetivo adicional es la función objetivo original $f(\vec{x})$.

El problema resultante puede ser resuelto por cualquier algoritmo evolutivo multi-objetivo [Coello et al., 2007].

Se hace notar, sin embargo, que el problema multi-objetivo que se plantea en este tipo de enfoque no es convencional y requiere de algunas consideraciones especiales para resolverlo.

Carlos A. Coello Coello, Gary B. Lamont and David A. Van Veldhuizen, **Evolutionary Algorithms for Solving Multi-Objective Problems**, Second Edition, Springer, New York, ISBN 978-0-387-33254-3, September 2007.

Separación de Restricciones y Objetivos

Uso de conceptos de optimización multi-objetivo

Existen diversas propuestas de este tipo en la literatura especializada.

Surry & Radcliffe [1997] propusieron COMOGA (*Constrained Optimization by Multiobjective Optimization Genetic Algorithms*) en el cual los individuos se jerarquizan de acuerdo a la optimalidad de Pareto, usando su suma de violación de restricciones.

Posteriormente, las soluciones pueden seleccionarse usando un torneo binario basado ya sea en la jerarquía de Pareto o en el valor de la función objetivo.

Patrick D. Surry and Nicholas J. Radcliffe, "**The COMOGA Method: Constrained Optimisation by Multiobjective Genetic Algorithms**", *Control and Cybernetics*, Vol. 26, No. 3, pp. 391–412, 1997.

Separación de Restricciones y Objetivos

Uso de conceptos de optimización multi-objetivo

Zhou et al. [2003] propusieron un procedimiento de jerarquización basado en el concepto de "fortaleza" de Pareto (utilizado en el *Strength Pareto Evolutionary Algorithm* [Zitzler, 1999]) para un problema bi-objetivo.

Este concepto consiste en contar el número de individuos que son dominados por una cierta solución.

Los empates se resuelven mediante la suma de la violación de restricciones, que se considera en este caso como el segundo objetivo del problema.

Se adopta el operador de cruza Simplex (SPX) para generar un conjunto de hijos, en los cuales el individuo con la fortaleza más alta así como la solución con la suma de restricciones más baja son seleccionados para formar parte de la siguiente generación.

Yuren Zhou, Yuanxiang Li, Jun He and Lishan Kang, "**Multi-objective and MGG Evolutionary Algorithm for Constrained Optimization**", in *Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC'2003)*, pp. 1–5, Vol. 1, IEEE Press, Canberra, Australia, December 2003.

Separación de Restricciones y Objetivos

Uso de conceptos de optimización multi-objetivo

Venkatraman y Yen [2005] propusieron un marco genérico para optimización con restricciones, dividido en dos fases: la primera fase trata el problema de optimización no lineal con restricciones como uno de satisfacción de restricciones (es decir, el objetivo es encontrar al menos una solución factible). Para conseguir esa meta, se jerarquiza la población con base sólo en la suma de violación de las restricciones.

La segunda fase comienza una vez que se obtiene al menos una solución factible.

En este punto, se consideran dos objetivos (la función objetivo original y la suma de violación de las restricciones) y se aplica el ordenamiento no dominado [Deb, 2002] para jerarquizar la población.

Sangameswar Venkatraman and Gary G. Yen, "**A Generic Framework for Constrained Optimization Using Genetic Algorithms**", *IEEE Transactions on Evolutionary Computation*, Vol. 9, No. 4, pp. 424–435, August 2005.

Separación de Restricciones y Objetivos

Uso de conceptos de optimización multi-objetivo

Hernández et al. [2004] propusieron una técnica denominada IS-PAES, la cual se basa en un algoritmo evolutivo multi-objetivo llamado *Pareto Archived Evolution Strategy* (PAES), que fue propuesto por Knowles y Corne [2000].

IS-PAES usa un archivo externo para almacenar las mejores soluciones encontradas. En este caso, se usa la dominancia de Pareto como un criterio secundario (sólo para decidir si se inserta o no una nueva solución en el archivo externo).

Arturo Hernández-Aguirre, Salvador Botello-Rionda, Carlos A. Coello Coello, Giovanni Lizárraga-Lizárraga and Efrén Mezura-Montes, "**Handling Constraints using Multiobjective Optimization Concepts**", *International Journal for Numerical Methods in Engineering*, Vol. 59, No. 15, pp. 1989–2017, April 2004.

Separación de Restricciones y Objetivos

Uso de conceptos de optimización multi-objetivo

Runarsson y Yao [2005] presentaron una comparación de dos esquemas posibles para incorporar la jerarquización de Pareto en el espacio de las restricciones:

1. Considerando el valor de la función objetivo en el proceso de jerarquización.
2. Sin considerar el valor de la función objetivo.

Estas dos versiones fueron comparados con respecto al uso de una función de penalización sobre-penalizada. Los autores concluyeron que el uso de jerarquización de Pareto produce una búsqueda sin sesgo, lo que hace que la búsqueda se realice la mayor parte del tiempo en la región infactible. Esto hace que se produzcan únicamente soluciones infactibles, y si se llegan a producir soluciones factibles, éstas tienen un valor pobre en la función objetivo.

homas Philip Runarsson and Xin Yao, "**Search biases in constrained evolutionary optimization**", *IEEE Transactions on Systems, Man, and Cybernetics Part C—Applications and Reviews*, Vol. 35, No. 2, pp. 233–243, May 2005.

Separación de Restricciones y Objetivos

Uso de conceptos de optimización multi-objetivo

Es importante destacar, sin embargo, que la jerarquización de Pareto "pura" prácticamente nunca se usa para el manejo de restricciones.

Esto se debe a que se necesita un sesgo para guiar el mecanismo de selección.

En problemas con restricciones, una vez que una restricción se satisface, no tiene sentido manejarla más como un objetivo, por lo cual ya no debiera ser incluída en el chequeo de dominancia de Pareto.

Métodos Híbridos

En esta categoría consideramos a métodos que se acoplan a otra técnica (ya sea otra metaheurística o una técnica de programación matemática).

Adeli y Cheng [1994] propusieron un algoritmo genético híbrido que integra una función de penalización con el método primal-dual. Esta técnica se basa en la minimización secuencial del método Lagrangiano.

Hojjat Adeli and Nai-Tsang Cheng, "**Augmented Lagrangian Genetic Algorithm for Structural Optimization**", *Journal of Aerospace Engineering*, Vol. 7, No. 1, pp. 104–118, January 1994.

Métodos Híbridos

Kim and Myung [1997] propusieron el uso de un método de optimización evolutiva combinado con una función Lagrangiana aumentada que garantiza la generación de soluciones factibles durante el proceso de búsqueda.

J.-H. Kim and H. Myung, "**Evolutionary programming techniques for constrained optimization problems**", *IEEE Transactions on Evolutionary Computation*, Vol. 1, pp. 129–140, July 1997.

Constrained optimization by random evolution (CORE)

Esta técnica fue propuesta por Belur [1997] y combina una búsqueda usando *random evolution* con el método de Nelder-Mead [1965].

Sheela V. Belur, "**CORE: Constrained Optimization by Random Evolution**", John R. Koza (editor), *Late Breaking Papers at the Genetic Programming 1997 Conference*, pp. 280–286, Stanford bookstore, Stanford University, California, USA, July 1997.

Colonia de Hormigas

La metaheurística denominada *colonia de hormigas* es un sistema multi-agente en el cual las interacciones de bajo nivel entre los agentes individuales (o sea, las hormigas artificiales) resulta en un comportamiento complejo de toda la colonia.

Aunque se le usa principalmente para optimización combinatoria, la colonia de hormigas ha sido extendida también para espacios de búsqueda continuos [Bilchev & Parmee, 1995; Leguizamon, 2004].

Aquí vale la pena mencionar el uso de la colonia de hormigas para manejar restricciones de igualdad, mediante la exploración de la frontera entre la región factible y la infactible [Leguizamon & Coello, 2009].

Guillermo Leguizamón and Carlos A. Coello Coello, "**Boundary Search for Constrained Numerical Optimization Problems with an Algorithm Inspired on the Ant Colony Metaphor**", *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 2, pp. 350–368, April 2009.

Métodos Híbridos

Recocido Simulado

Wah & Chen [2001] propusieron un híbrido entre el recocido simulado y un algoritmo genético.

Usando este híbrido, la primera parte de la búsqueda es guiada por el recocido simulado. Posteriormente, la mejor solución obtenida es refinada por el algoritmo genético.

Para el manejo de restricciones, los autores utilizan multiplicadores de Lagrange.

Benjamin W. Wah and Yixin Chen, "**Hybrid Constrained Simulared Annealing and Genetic Algorithms for Nonlinear Constrained Optimization**", in *Proceedings of the 2001 IEEE Congress on Evolutionary Computation (CEC'2001)*, Vol. 2, pp. 925–932, IEEE Press, May 2001.

Sistema Inmune Artificial

Hajela y Lee [1996] propusieron un híbrido entre un algoritmo genético y un sistema inmune artificial basado en selección negativa, para resolver problemas con restricciones.

La idea principal consistía en hacer que las soluciones factibles fueran los antígenos y evolucionar a las soluciones infactibles (que hacen las veces de los anticuerpos) de manera que se volvieran "similares" (usando distancia de Hamming) a los antígenos, sin llegar a ser idénticas.

P. Hajela and J. Lee, "**Constrained Genetic Search via Schema Adaptation. An Immune Network Solution**", *Structural Optimization*, Vol. 12, pp. 11–15, 1996.

Algoritmos Culturales

Esta técnica fue propuesta por Robert Reynolds en 1994, y su idea fundamental es preservar creencias que son socialmente aceptables y desechar (o podar) aquellas que son inaceptables.

Las creencias aceptables pueden verse como restricciones que dirigen a la población a un nivel micro-evolutivo.

Por tanto, las restricciones pueden influenciar directamente el proceso de búsqueda, lo cual conduciría a un proceso de optimización muy eficiente.

Algoritmos Culturales

En otras palabras, cuando se usan algoritmos culturales, se extrae algún tipo de conocimiento durante el proceso de búsqueda y éste se utiliza para influenciar a los operadores evolutivos, de tal forma que pueda realizarse una búsqueda más eficiente.

Las primeras versiones de los algoritmos culturales para optimización con restricciones tenían algunos problemas de manejo de memoria [Chung & Reynolds, 1996], pero posteriormente, fueron mejoradas usando estructuras de datos espaciales que permiten manejar un número relativamente elevado de variables [Coello & Landa, 2002].

Chan-Jin Chung and Robert G. Reynolds, "**A Testbed for Solving Optimization Problems Using Cultural Algorithms**", in Lawrence J. Fogel, Peter J. Angeline and Thomas Bäck (Editors), *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pp. 225–236, MIT Press, March 1996.

Carlos A. Coello Coello and Ricardo Landa Becerra, "**Adding Knowledge and Efficient Data Structures to Evolutionary Programming: A Cultural Algorithm for Constrained Optimization**", in W.B. Langdon et al. (editors), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pp. 201–209, Morgan Kaufmann Publishers, San Mateo, California, USA, July 2002.

An Ensemble of Constraint-Handling Techniques

Mallipeddi y Suganthan [2010] propusieron el uso de una combinación (*ensemble*) de técnicas de manejo de restricciones.

La idea es tener varias ténicas de manejo de restricciones, cada una de las cuales cuenta con su propia población y parámetros.

Cada población produce sus hijos y los evalúa. Sin embargo, estos hijos compiten no sólo contra su propia población, sino también contra las demás. Por tanto, un cierto hijo puede ser rechazado por su población, pero puede ser aceptado en otra.

Esto busca automatizar la selección de la mejor técnica de manejo de restricciones para un problema en particular, con base en el hecho de que ninguna de ellas será la mejor en todos los casos.

Rammohan Mallipeddi and Ponnuthurai N. Suganthan, "**Ensemble of Constraint Handling Techniques**", *IEEE Transactions On Evolutionary Computation*, Vol. 14, No. 4, pp. 561–579, August 2010.

Uso de Información Basada en el Gradiente

Han habido algunas propuestas para incorporar información de gradiente ya sea de los valores de aptitud o de las restricciones.

Por ejemplo, Sun y Garibaldi [2010] propusieron un Algoritmo de Estimación de Distribución combinado con un optimizador local basado en gradiente.

Jianyong Sun and Jonathan M. Garibaldi, "**A Novel Memetic Algorithm for Constrained Optimization**", in *2010 IEEE Congress on Evolutionary Computation (CEC'2010)*, pp. 549–556, IEEE Press, Barcelona, Spain, July 18-23, 2010.

Uso de Información Basada en el Gradiente

Handoko et al. [2010] propusieron un algoritmo memético que combina un algoritmo genético, programación secuencial cuadrática, aproximaciones funcionales de segundo orden y una técnica para modelar la región factible a través del uso de máquinas de soporte vectorial.

Hamza et al. [2014] propusieron una variante de un algoritmo genético basada en consenso, la cual se combina con programación secuencial cuadrática, pero usando información del gradiente de las restricciones.

Stephanus Daniel Handoko and Chee Keong Kwoh and Yew-Soon Ong, "**Feasibility Structure Modeling: An Effective Chaperone for Constrained Memetic Algorithms**", *IEEE Transactions on Evolutionary Computation*, Vol. 14, No. 5, pp. 740–758, October 2010.

Noha M. Hamza, Ruhul A. Sarker, Daryl L. Essam, Kalyanmoy Deb and Saber M. Elsayed, "**A Constraint Consensus Memetic Algorithm for Solving Constrained Optimization Problems**", *Engineering Optimization*, Vol. 46, No. 11, pp. 1447–1464, November 2014.

Viabilidad Memética

Maesani et al. [2016] propusieron el uso de una noción llamada *evolución de la viabilidad*, la cual enfatiza la eliminación de soluciones que no satisfagan los criterios de viabilidad (o sea, los límites de los objetivos y las restricciones).

Tales límites se adaptan durante la búsqueda de manera análoga a como se hace en ASCHEA. Sin embargo, en este caso, la técnica (que se basa en el uso de la *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES)) produce una población de motores de búsqueda local que pueden ser recombinados usando evolución diferencial.

Un aspecto interesante de esta técnica es que usa un programador de tareas adaptativo que cambia entre exploración y explotación, eligiendo si avanzar con uno de los buscadores locales (o individuos) o recombinarlos.

Andrea Maesani and Giovanni Iacca and Dario Floreano, "**Memetic Viability Evolution for Constrained Optimization**", *IEEE Transactions on Evolutionary Computation*, Vol. 20, No. 1, pp. 125–144, February 2016.

Problemas de Prueba

Michalewicz y Schoenauer [1996] propusieron un conjunto de problemas de prueba, que fueron expandidos posteriormente por Runarsson y Yao [2000].

El conjunto de prueba que se usó por muchos años en el área consiste de 13 problemas de prueba.

Estos problemas contienen características que son representativas de lo que se considera como "difícil" en optimización con restricciones usando algoritmos evolutivos.

Zbigniew Michalewicz and Marc Schoenauer, "**Evolutionary Algorithms for Constrained Parameter Optimization Problems**", *Evolutionary Computation*, Vol. 4, No. 1, pp. 1–32, 1996.

Thomas P. Runarsson and Xin Yao, "**Stochastic Ranking for Constrained Evolutionary Optimization**", *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 3, pp. 284–294, September 2000.

Problemas de Prueba

Existen, sin embargo, otros conjuntos de problemas de prueba. Por ejemplo:

- Efrén Mezura Montes and Carlos A. Coello Coello, **What Makes a Constrained Problem Difficult to Solve by an Evolutionary Algorithm**, Technical Report EVOCINV-01-2004, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México, February 2004.
- C. A. Floudas and P. M. Pardalos, **A Collection of Test Problems for Constrained Global Optimization Algorithms**, Number 455 in Lecture Notes in Computer Science. Springer-Verlag, 1990.
- Christodoulos A. Floudas et al. (editors), **Handbook of Test Problems in Local and Global Optimization**, Kluwer Academic Publishers, Dordrecht, 1999.

Problemas de Prueba

Problema	n	Tipo de función	ρ	LI	NI	LE	NE
g01	13	cuadrática	0.0003%	9	0	0	0
g02	20	no lineal	99.9973%	1	1	0	0
g03	10	no lineal	0.0026%	0	0	0	1
g04	5	cuadrática	27.0079%	0	6	0	0
g05	4	no lineal	0.0000%	2	0	0	3
g06	2	no lineal	0.0057%	0	2	0	0
g07	10	cuadrática	0.0000%	3	5	0	0
g08	2	no lineal	0.8581%	0	2	0	0
g09	7	no lineal	0.5199%	0	4	0	0
g10	8	lineal	0.0020%	3	3	0	0
g11	2	cuadrática	0.0973%	0	0	0	1
g12	3	cuadrática	4.7697%	0	9^3	0	0
g13	5	no lineal	0.0000%	0	0	1	2

Problemas de Prueba

Se han propuesto conjuntos de problemas de prueba adicionales, así como un nuevo conjunto de 24 problemas de prueba (incluyendo los 13 problemas propuestos en 1996). Ver:

- J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. Coello Coello, K. Deb, **Problem Definitions and Evaluation Criteria for the CEC 2006, Special Session on Constrained Real-Parameter Optimization**, Technical Report, Nanyang Technological University, Singapore, 2006.

Existe también un conjunto de problemas de prueba que fueron introducidos en el *2010 World Congress on Computational Intelligence* (WCCI'2010). Este conjunto consiste de 18 problemas de prueba escalables.

R. Mallipeddi and P. N. Suganthan, **Problem Definitions and Evaluation Criteria for the CEC 2010 Competition on Constrained Real-Parameter Optimization**, Technical Report, Nanyang Technological University, Singapore, 2010. Disponible en:
`http://www.ntu.edu.sg/home/EPNSugan/`.

Problemas de Prueba

Hay sitios web con colecciones de problemas de prueba (incluyendo problemas de optimización con restricciones) que vale la pena visitar. Ver por ejemplo:

`https://www.mat.univie.ac.at/~neum/glopt.html`

También existe trabajo en progreso encaminado a desarrollar una rama de COCO para problemas de optimización de caja negra (*black-box*) con restricciones (BBOB-constrained). Este nuevo conjunto de problemas de prueba toma en consideración una sola medida de desempeño: el tiempo de ejecución del algoritmo. En este caso, el tiempo de ejecución se define en términos del número de evaluaciones realizadas por un algoritmo en un problema en particular, hasta que se alcance un valor meta previamente definido.

Problemas de Prueba

Hellwig y Beyer [2019] proporcionan una revisión de los dominios de problemas con restricciones disponibles en la actualidad para fines de validación. La discusión de estos autores incluye principios comunes que deben tomarse en cuenta cuando se consideran problemas de prueba para optimización con restricciones.

Michael Hellwig and Hans-Georg Beyer, "**Benchmarking evolutionary algorithms for single objective real-valued constrained optimization – A Critical Review**", *Swarm and Evolutionary Computation*, Vol. 55, pp. 927–944, 2019.

Generadores de Problemas de Prueba

Michalewicz et al. [2000] propusieron un generador de problemas de prueba para técnicas de optimización diseñados para problemas con restricciones.

Este generador permite construir problemas de prueba a través de la modificación de varias características tales como: dimensionalidad, multimodalidad, número de restricciones, conectividad de la región factible, tamaño de la región factible con respecto al de todo el espacio del búsqueda, así como la rugosidad de la función objetivo.

Zbigniew Michalewicz, Kalyanmoy Deb, Martin Schmidt and Thomas Stidsen, "**Test-Case Generator for Nonlinear Continuous Parameter Optimization Techniques**", *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 3, pp. 197–215, September 2000.

Generadores de Problemas de Prueba

La primera versión de este generador de problemas de prueba presentó dificultades porque producía funciones simétricas.

Esto motivó el desarrollo de una nueva versión de este generador de problemas de prueba, llamado TCG-2 [Schmidt y Michalewicz, 2000].

Martin Schmidt and Zbigniew Michalewicz, "**Test-Case Generator TCG-2 for Nonlinear Parameter Optimisation**", in Marc Schoenauer et al. (Editors), *Proceedings of the 6th Parallel Problem Solving From Nature (PPSN VI)*, pp. 539–548, Springer-Verlag, Lecture Notes in Computer Science Vol. 1917, Paris, France, September 2000.

Algunas Recomendaciones

- Estudiar y probar primero las técnicas de programación matemática (p.ej., los métodos de gradiente, el método de Nelder-Mead, el método de Hooke-Jeeves, etc.).
- Si están interesados en optimización numérica, es aconsejable utilizar primero las estrategias evolutivos o la evolución diferencial, en vez de usar algoritmos genéticos. Sin embargo, si se llegan a usar algoritmos genéticos, se recomienda utilizar selección más.
- Es importante poner atención a la diversidad. Mantener poblaciones en las que no todos los individuos son factibles es siempre una buena idea.
- Normalizar las restricciones del problema suele ser una buena idea.

Temas Actuales de Investigación

- Nuevas técnicas de manejo de restricciones (p.ej., basadas en conceptos de optimización multi-objetivo).
- Optimización con restricciones a gran escala [Aguilar-Justo & Mezura-Montes, 2016].
- Técnicas de manejo de restricciones para algoritmos evolutivos multi-objetivo [Yen, 2009; Fukumoto and Oyama, 2018].

Adan E. Aguilar-Justo and Efrén Mezura-Montes, "**Towards an Improvement of Variable Interaction Identification for Large-Scale Constrained Problems**", in *2016 IEEE Congress on Evolutionary Computation (CEC'2016)*, pp. 4167–4174, IEEE Press, Vancouver, Canada, July 24-29, 2016.

Gary G. Yen, "**An Adaptive Penalty Function for Handling Constraints in Multi-objective Evolutionary Optimization**", in Efrén Mezura-Montes (Editor), *Constraint-Handling in Evolutionary Computation*, pp. 121–143, Chapter 6, Springer. Studies in Computational Intelligence, Volume 198, Berlin, 2009.

Hiroaki Fukumoto and Akira Oyama, "**A Generic Framework for Incorporating Constraint Handling Techniques into Multi-Objective Evolutionary Algorithms**", in Kevin Sim and Paul Kaufmann (editors), *Applications of Evolutionary Computation 21st International Conference, EvoApplications 2018*, pp. 634–649, Springer-Verlag, Lecture Notes in Computer Science Vol. 10784, Parma, Italy, April 4-6, 2018.

Temas Actuales de Investigación

- Técnicas existentes de manejo de restricciones con nuevos motores de búsqueda (p.ej., colonia de hormigas, búsqueda dispersa, etc.).
- Mecanismos auto-adaptativos para optimización con restricciones [Brest, 2009].
- Métodos para manejo de restricciones de igualdad en optimización multi-objetivo [Saha & Ray, 2012].
- Análisis de complejidad temporal de los algoritmos evolutivos usados para resolver problemas con restricciones (particularmente, el papel de los factores de penalización en la complejidad temporal de un algoritmo evolutivo) [Zhou & He, 2007].

Janez Brest, "**Constrained Real-Parameter Optimization with ϵ-Self-Adaptive Differential Evolution**", in Efrén Mezura-Montes (Editor), *Constraint-Handling in Evolutionary Computation*, pp. 73–93, Chapter 4, Springer. Studies in Computational Intelligence, Volume 198, Berlin, 2009.

Amit Saha and Tapabrata Ray, "**Equality Constrained Multi-objective Optimization**", in *2012 IEEE Congress on Evolutionary Computation (CEC'2012)*, pp. 47–53, IEEE Press, Brisbane, Australia, June 10-15, 2012.

Yuren Zhou and Jun He, "**A Runtime Analysis of Evolutionary Algorithms for Constrained Optimization Problems**", *IEEE Transactions on Evolutionary Computation*, Vol. 11, No. 5, pp. 608–619, October 2007

Temas Actuales de Investigación

- Uso de esquemas de votación como una manera de incorporar preferencias a fin de permitir la exploración de una porción de la región infactible [Yu, 2014].
- Híbridos de algoritmos evolutivos con técnicas de programación matemática (p.ej., estrategias evolutivas + método de Nelder-Mead, uso de multiplicadores de Lagrange, etc.). Ver por ejemplo [Arnold and Porter, 2015].
- Restricciones dinámicas [Nguyen & Yao, 2012].

Erdong Yu, Qing Fei, Hongbin Ma and Qingbo Geng, "**Improving Constraint Handling for Multiobjective Particle Swarm Optimization**", in *Proceedings of the 33rd Chinese Control Conference*, pp. 8622–8627, IEEE Press, Nanjing, China, July 28-30, 2014.

Dirk V. Arnold and Jeremy Porter, "**Towards an Augmented Lagrangian Constraint-Handling Approach for the (1+1)-ES**", in *2015 Genetic and Evolutionary Computation Conference (GECCO 2015)*, pp. 249–256, ACM Press, Madrid, Spain, July 11-15, 2015, ISBN 978-1-4503-3472-3.

Trung Thanh Nguyen and Xin Yao, "**Continuous Dynamic Constrained Optimization—The Challenges**", *IEEE Transactions on Evolutionary Computation*, Vol. 16, No. 6, pp. 769–786, December 2012.

Temas Actuales de Investigación

- Técnicas que reducen el número de evaluaciones de las funciones objetivo que se realizan en problemas con restricciones (p.ej., modelos subrogados, herencia de aptitud, aproximación de aptitud). Ver por ejemplo [Bagheri et al., 2016].
- Optimización restringida robusta [Tagawa y Miyanaga, 2017].
- Generadores de problemas de prueba (cómo construirlos y cómo hacerlos confiables). Problemas de prueba disponibles en línea).
- Criterios de paro para problemas de optimización con restricciones resueltos con algoritmos evolutivos [Zielinski y Laur, 2008].

Samineh Bagheri, Wolfgang Konen and Thomas Bäck, "**Equality Constraint Handling for Surrogate-Assisted Constrained Optimization**", in *2016 IEEE Congress on Evolutionary Computation (CEC'2016)*, pp. 1924–1931, IEEE Press, Vancouver, Canada, July 24-29, 2016, ISBN 978-1-5090-0623-9.

Kiyoharu Tagawa and Shun Miyanaga, "**Weighted empirical distribution based approach to Chance Constrained Optimization Problems using Differential Evolution**", in *2017 IEEE Congress on Evolutionary Computation (CEC'2017)*, pp. 97–104, IEEE Press, San Sebastián, Spain, June 5-8, 2017.

Karin Zielinski and Rainer Laur, "**Stopping Criteria for Differential Evolution in Constrained Single-Objective Optimization**", in Uday K. Chakraborty (Editor), *Advances in Differential Evolution*, pp. 111–138, Springer, Berlin, 2008.

Temas Actuales de Investigación

- Nuevas métricas que nos permitan evaluar el desempeño en línea de una técnica de manejo de restricciones y la caracterización de espacios de búsqueda restringidos (ver por ejemplo la noción de *violation landscape* propuestos en [Malan, 2015]).
- Operadores especiales para explorar la frontera entre las regiones factible e infactible [Leguizamón & Coello, 2009].

Katherine M. Malan, Johannes F. Oberholzer and Andries P. Engelbrecht, "**Characterising Constrained Continuous Optimisation Problems**", in *2015 IEEE Congress on Evolutionary Computation (CEC'2015)*, pp. 1351–1358, IEEE Press, Sendai, Japan, 25-28 May, 2015.

Guillermo Leguizamón and Carlos A. Coello Coello, "**Boundary Search for Constrained Numerical Optimization Problems with an Algorithm Inspired on the Ant Colony Metaphor**", *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 2, pp. 350-368, April 2009.

Temas Actuales de Investigación

En un artículo reciente, Picard y Schiffmann [2021] re-visitan un tema que ha sido un tanto controversial en el área: el hecho de que hay evidencia que indica que las técnicas de manejo de restricciones del estado del arte (en este caso para algoritmos evolutivos multi-objetivo) no tienen un buen desempeño cuando se usan para resolver problemas de optimización en ingeniería (problemas de diseño mecánico en este caso). Esto a pesar de que estos algoritmos del estado del arte tienen un muy buen desempeño en problemas de prueba [Picard, 2018; Fukumoto, 2018].

Cyril Picard and Jürg Schiffmann, "**Realistic Constrained Multiobjective Optimization Benchmark Problems From Design**", *IEEE Transactions on Evolutionary Computation*, Vol. 25, No. 2, pp. 234–246, April 2021.

Cyril Picard and Jürg Schiffmann, "**Impacts of Constraints and Constraint Handling Strategies for Multi-Objective Mechanical Design Problems**", in *2018 Genetic and Evolutionary Computation Conference (GECCO'2018)*, pp. 1341–1347, ACM Press, July 2018.

H. Fukumoto and A. Oyama, "**Benchmarking multiobjective evolutionary algorithms and constraint handling techniques on a real-world car structure design optimization benchmark problem**", in *Proceedings of the 2018 Genetic and Evolutionary Computaton Conference (GECCO'2018) Companion*, pp. 177–178, ACM Press, July 2018.

Temas Actuales de Investigación

Picard y Schiffmann [2021] indican que los problemas de optimización multi-objetivo con restricciones incluidos en los benchmarks actuales, tienen las siguientes limitantes:

- Son demasiado simples.
- Ofrecen una escalabilidad limitada del espacio de búsqueda y en el número de funciones objetivo.
- Incorporan solo una o dos restricciones de desigualdad.
- No son ajustables en términos de la complejidad de las restricciones.

Los autores también indican que los problemas de prueba con restricciones no han sido caracterizados adecuadamente. Normalmente, la única información que se proporciona de ellos es: el número de variables de decisión, el número de restricciones, el verdadero frente de Pareto y la tasa de factibilidad. Sin embargo, hay estudios que indican que esta información es insuficiente [Tanabe, 2017].

R. Tanabe and A. Oyama, "**A note on constrained multi-objective optimization benchmark problems**,, in *Proceedings of the 2017 IEEE Congress on Evolutionary Computation*, pp. 1127–1134, IEEE Press, June 2017.

Temas Actuales de Investigación

Con base en este análisis, Picard y Schiffmann [2021] proponen un entorno genérico para diseñar actuadores electro-mecánicos. Este entorno es denominado *Multiobjective Design of Actuators* (MODAct), y se adopta para instanciar un conjunto de problemas de prueba multi-objetivo con restricciones que son más realistas que los disponibles en los benchmarks actuales. Los nuevos problemas son escalables en el número de variables de decisión, en el número de funciones objetivo y en el número y tipo de restricciones.

Este artículo proporciona también un análisis del paisaje de aptitud de los nuevos problemas de prueba, y propone métricas para obtener información detallada sobre las características de dichos problemas. Esta es una contribución muy interesante que intenta ligar la optimización en ingeniería (en un ambiente del mundo real) con el desarrollo de nuevas técnicas de manejo de restricciones.

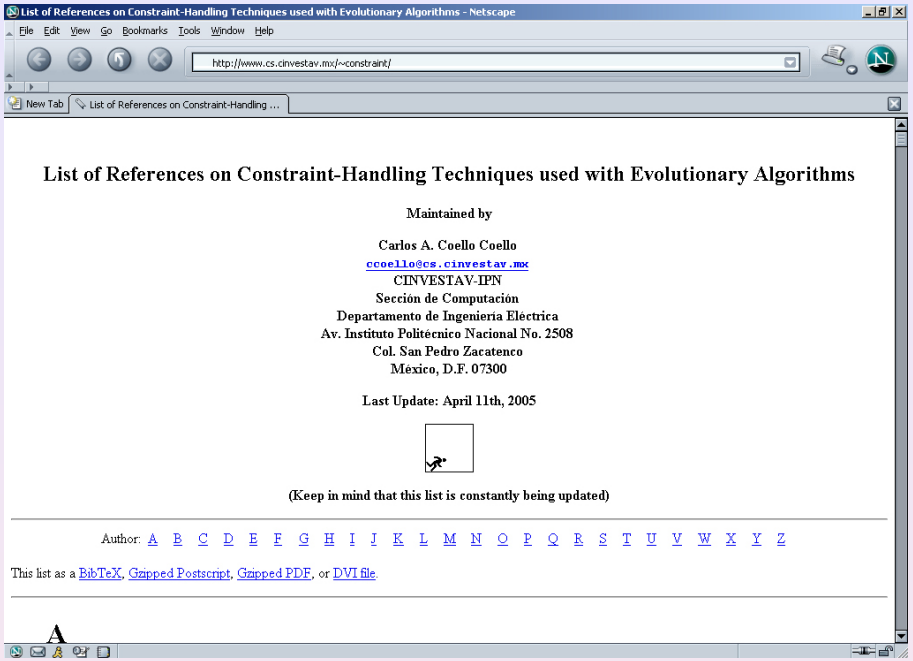
Algunas Ideas

- Se necesitan más problemas de prueba y más generadores de problemas de prueba. Por ejemplo, es deseable tener más problemas con restricciones de igualdad y más problemas multi-objetivo con restricciones.
- Es importante tener nuevas métricas que nos permitan medir el desempeño en línea de una técnica de manejo de restricciones, así como caracterizar los espacios de búsqueda con restricciones (ver por ejemplo, la noción de *paisaje de violación* propuesta en [Malan et al., 2015]).
- Es todavía posible proponer nuevas técnicas de manejo de restricciones, pero éstas son raras en las actualidad. Pero, ¿por qué no enfocarnos en entender las limitantes de las técnicas existentes de manejo de restricciones? Ver por ejemplo: [Arnold, 2011].

Katherine M. Malan, Johannes F. Oberholzer and Andries P. Engelbrecht, "**Characterising Constrained Continuous Optimisation Problems**", in *2015 IEEE Congress on Evolutionary Computation (CEC'2015)*, pp. 1351–1358, IEEE Press, Sendai, Japan, 25-28 May 2015, ISBN 978-1-4799-7492-4.

Dirk V. Arnold, "**Analysis of a Repair Mechanism for the $(1, \lambda)$-ES Applied to a Simple Constrained Problem**", in *2011 Genetic and Evolutionary Computation Conference (GECCO'2011)*, pp. 853–860, ACM Press, Dublin, Ireland, July 12-16, 2011.

Para saber más...

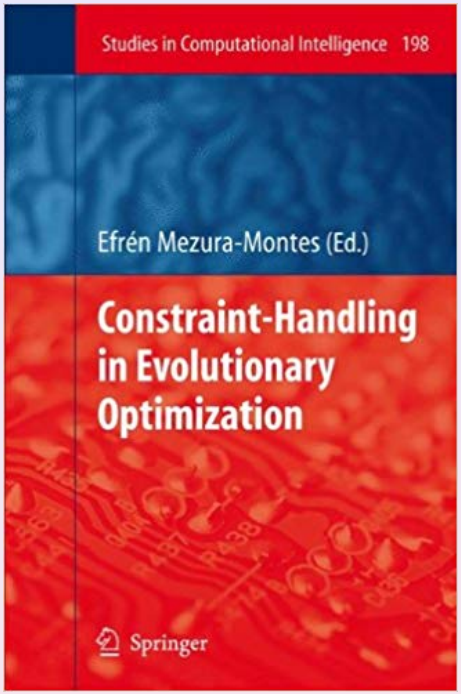


El repositorio de técnicas de manejo de restricciones usadas con algoritmos evolutivos está localizado en:

http://www.cs.cinvestav.mx/~constraint

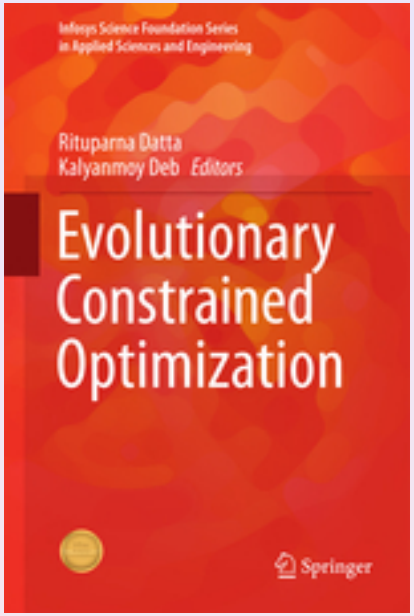
Al **2 de mayo de 2023**, este repositorio contiene **1450** referencias bibliográficas.

Lecturas Sugeridas



Efrén Mezura-Montes (editor), **Constraint-Handling in Evolutionary Optimization**, Springer, 2009, ISBN: 978-3-642-00618-0.

Lecturas Sugeridas



Rituparna Datta and Kalyanmoy Deb (editors), **Evolutionary Constrained Optimization**, Springer, 2015, ISBN: 978-81-322-2183-8.