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 9

Ajuste de Parámetros de un Algoritmo Genético

Los experimentos de David Schaffer

Schaffer et al. [1989] efectuaron una serie de experimentos que consumieron aproximadamente 1.5 años de tiempo de CPU (en una Sun 3 y una VAX), en los cuales intentaron encontrar los parámetros óptimos de un algoritmo genético con codificación de Gray y usando selección universal estocástica.

J. David Schaffer, Richard A. Caruana, Larry J. Eshelman and Raharshi Das, "**A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization**", in J. David Schaffer (Editor), *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 51–60, Morgan Kaufmann Publishers, San Mateo, California, USA, 1989.

Ajuste de Parámetros de un Algoritmo Genético

Los experimentos de David Schaffer

Los parámetros sugeridos por estos experimentos (para el desempeño "en línea") fueron:

Tamaño de población:	20-30 individuos
Porcentaje de cruza (2 puntos):	0.75-0.95
Porcentaje de mutación:	0.005-0.01

Los experimentos de David Schaffer

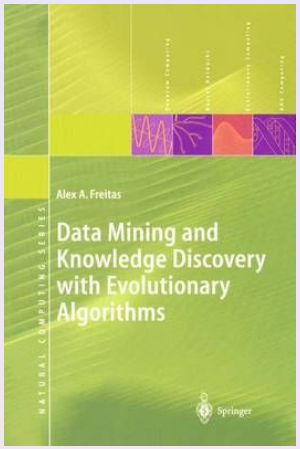
Algunas de las observaciones de Schaffer et al. [1989] fueron las siguientes:

1. El uso de tamaños grandes de población (> 200) con porcentajes altos de mutación (> 0.05) no mejora el desempeño de un algoritmo genético.
2. El uso de poblaciones pequeñas (< 20) con porcentajes bajos de mutación (< 0.002) no mejora el desempeño de un algoritmo genético.
3. La mutación parece tener mayor importancia de lo que se cree en el desempeño de un algoritmo genético.

Los experimentos de David Schaffer

4. El algoritmo genético resultó relativamente insensible al porcentaje de cruza. Un algoritmo de NE (*naive evolution*), o sea, un algoritmo genético sin cruza, funciona como un *hill climber*, el cual puede resultar más poderoso de lo que se cree.
5. Los operadores genéticos pueden muestrear eficientemente el espacio de búsqueda sin necesidad de usar tamaños de población excesivamente grandes.
6. La cruza de 2 puntos es mejor que la de un punto, pero sólo marginalmente.
7. Conforme se incrementa el tamaño de la población, el efecto de la cruza parece diluirse.

Ajuste de Parámetros de un Algoritmo Genético



Auto-Adaptación

En general, es poco probable poder determinar “a priori” un conjunto óptimo de parámetros para un algoritmo genético cualquiera aplicado a un problema en particular.

Algunos investigadores creen que la mejor opción es la **auto-adaptación**, o sea permitir que un algoritmo genético adapte por sí mismo sus parámetros.

Auto-Adaptación

Davis [1989] realizó un estudio muy interesante sobre un mecanismo de auto-adaptación aplicado a algoritmos genéticos.

En su estudio, Davis asignó a cada operador genético una "aptitud", la cual era función de cuántos individuos con aptitud elevada habían contribuido a crear dicho operador en un cierto número de generaciones. Un operador era recompensado por crear buenos individuos directamente, o por "dejar la mesa puesta" para ello (es decir, por crear ancestros para los buenos individuos).

Lawrence Davis, "**Adapting Operator Probabilities in Genetic Algorithms**", in J. David Schaffer (Editor), *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 61–69, Morgan Kaufmann Publishers, San Mateo, California, USA, 1989.

Ajuste de Parámetros de un Algoritmo Genético

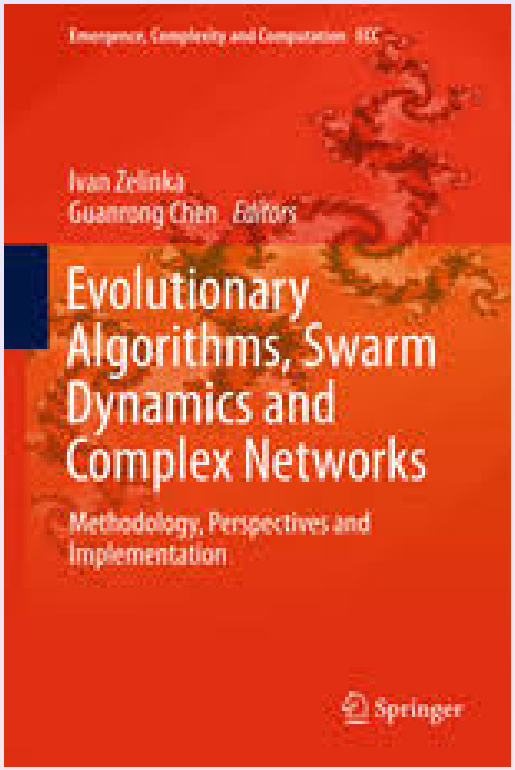
Auto-Adaptación

La técnica de Davis se usó en un algoritmo genético de estado uniforme.

Cada operador (cruza y mutación) empezaba con la misma aptitud y cada uno de estos operadores se seleccionaba con una probabilidad basada en su aptitud para crear un nuevo individuo, el cual reemplazaba al individuo menos apto de la población.

Cada individuo llevaba un registro de quién lo creó. Si un individuo tenía una aptitud mayor que la mejor aptitud actual, entonces el individuo recibía una recompensa para el operador que lo creó y ésta se propagaba a su padre, su abuelo, y tantos ancestros como se deseara.

Ajuste de Parámetros de un Algoritmo Genético



Auto-Adaptación

La aptitud de cada operador sobre un cierto intervalo de tiempo estaba en función de su aptitud previa y de la suma de recompensas recibidas por todos los individuos que ese operador hubiese ayudado a crear en ese tiempo.

Ajuste de Parámetros de un Algoritmo Genético

Auto-Adaptación

Pham [1995] propuso una estrategia competitiva para encontrar la mejor combinación de operadores y parámetros para los mismos.

Este método competitivo se aplicó entre varias sub-poblaciones que utilizaban diferentes conjuntos de parámetros.

Se evolucionaron varias poblaciones independientes en el mismo ambiente usando sus propios conjuntos de parámetros.

Estas poblaciones competían por un solo procesador, de tal forma que las poblaciones con los mejores valores de sus parámetros obtenían más tiempo del procesador que las otras para evolucionar.

Q.T. Pham, "**Competitive Evolution: A Natural Approach To Operator Selection**", in Xin Yao (Editor), *Progress in Evolutionary Computation*, pp. 49–60, Lecture Notes in Artificial Intelligence Vol. 956, Springer-Verlag, Heidelberg, Germany, 1995.

Auto-Adaptación del Porcentaje de Mutación

En este caso, el porcentaje de mutación se agrega como un parámetro más al genotipo, de tal forma que se vuelva una variable más tal que su valor oscile entre 0.0 y 1.0.

Bäck y Schütz [1996] proponen usar:

$$p'_m = \frac{1}{1 + \frac{1-p_m}{p_m} e^{-\gamma N(0,1)}}$$

donde:
p_m = porcentaje actual de mutación, p'_m = nuevo porcentaje de mutación.
γ = tasa de aprendizaje (se sugiere $\gamma = 0.2$).

Auto-Adaptación del Porcentaje de Mutación

$N(0,1)$ indica una variable aleatoria con una distribución normal tal que su esperanza es cero y su desviación estándar es uno.

Aplicando este operador, pasamos del genotipo:

$$c = (x, p_m)$$

al nuevo genotipo:

$$(x', p'_m)$$

Thomas Bäck and Martin Schütz, "**Intelligent mutation rate control in canonical genetic algorithms**", in Zbigniew W. Raś and Maciek Michalewicz (Editors), *Foundations of Intelligent Systems ISMIS 1996*, pp. 158–167, Springer-Verlag, Lecture Notes in Computer Science Vol. 1079, Zakopane, Poland, June 9-13, 1996.

Ajuste de Parámetros de un Algoritmo Genético

Auto-Adaptación del Porcentaje de Mutación

La mutación de la variable x está dada por:

$$x' = \begin{cases} x_j & \text{si } q \geq p'_m \\ 1 - x_j & \text{si } q < p'_m \end{cases}$$

donde: q es un valor aleatorio (distribución uniforme) muestreado de nuevo para cada posición j.

Auto-Adaptación del Porcentaje de Mutación

Este esquema puede generalizarse incluyendo un vector p de porcentajes de mutación asociados a cada variable:

$$p = (p_1, \cdots, p_L)$$

El genotipo c tiene la forma:

$$c = (x, p)$$

Los porcentajes de mutación se actualizan usando:

$$p'_j = \frac{1}{1 + \frac{1-p_j}{p_j} e^{-\gamma N(0,1)}}, j = 1, \cdots, L.$$

El algoritmo genético auto-adaptativo

Hinterding et al. [1996] propusieron un "**algoritmo genético auto-adaptativo**" que usa auto-adaptación para la tasa de mutación y un control adaptativo para el tamaño de la población.

El mecanismo para controlar la mutación es similar al propuesto por Bäck que vimos anteriormente, con la salvedad de que en este caso la mutación de los bits que codifican el porcentaje de mutación no se basa en esos bits, sino que se realiza con base en un mecanismo universal fijo para todos los individuos y todas las generaciones.

En otras palabras, el parámetro auto-adaptativo de la tasa de mutación se usa solo para los genes que codifican una solución.

Robert Hinterding, Zbigniew Michalewicz and T.C. Peachey, "**Self-Adaptive Genetic Algorithm for Numeric Functions**", in Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg and Hans-Paul Schwefel (Editors), *Parallel Problem Solving from Nature IV*, pp. 420–429, Springer-Verlag, Lecture Notes in Computer Science Vol. 1141, 1996.

Ajuste de Parámetros de un Algoritmo Genético

El algoritmo genético auto-adaptativo

Respecto al ajuste del tamaño de la población, Hinterding et al. [1996] usan 3 subpoblaciones: una pequeña (**P1**, de 50 individuos), una mediana (**P2** de 100 individuos) y una grande (**P3**, de 200 individuos).

Estas poblaciones se evolucionan en paralelo durante un cierto número fijo de evaluaciones (una época) y de manera independiente, para la misma configuración del algoritmo genético.

Después de cada época, las subpoblaciones reajustan su tamaño con base en unas reglas heurísticas, que mantienen un límite inferior y uno superior (10 y 1000) y que mantienen siempre a **P2** como la población de tamaño intermedio.

Ajuste de Parámetros de un Algoritmo Genético

El algoritmo genético auto-adaptativo

En este algoritmo, existen dos tipos de reglas.

Las reglas del primer tipo se activan cuando las aptitudes de una subpoblación han convergido e intenta separar las poblaciones. Por ejemplo, si **P2** y **P3** tienen la misma aptitud, se duplica el tamaño de **P3**.

Las reglas del segundo tipo se activan cuando los valores de aptitud son distintos al final de una época. Estas reglas buscan maximizar el desempeño de **P2**. Por ejemplo, si el desempeño de las subpoblaciones es: **P2** $<$ **P3** $<$ **P1**, entonces *size*(**P3**) = (*size*(**P2**)+*size*(**P3**))/2.

Este es realmente un mecanismo de adaptación en línea y no auto-adaptativo.

Ajuste de Parámetros de un Algoritmo Genético

Tongchim & Chulalongkorn [2002] propusieron un algoritmo adaptativo para ajustar dinámicamente los parámetros de un algoritmo genético durante su ejecución.

Se usaron dos niveles del algoritmo genético en este caso. Las subpoblaciones grandes evolucionan en paralelo usando diferentes conjuntos de parámetros en el nivel inferior del algoritmo genético.

El nivel superior del algoritmo genético es aplicado a la evolución de los conjuntos de parámetros. Dicha evolución ocurre si la aptitud de otro conjunto de parámetros es mejor que el conjunto actual.

Los autores encontraron que la confiabilidad y el número mínimo de generaciones seleccionados juegan un papel crucial en la obtención de la solución óptima.

S. Tongchim and P. Chongstitvatana, "**Parrallel Genetic Algorithm With Parameter Adaptation**" *Information Processing Letters*, Vol. 82, No. 1, pp. 47–54, 2002.

Críticas a la Auto-Adaptación

La auto-adaptación no ha sido tan exitosa en los algoritmos genéticos, como lo es en otras técnicas evolutivas (p.ej., las estrategias evolutivas) ¿Por qué?

El problema fundamental es que nadie ha podido contestar satisfactoriamente la siguiente pregunta [Mitchell, 1996]:

¿Qué tan bien corresponde la velocidad de adaptación de la población con la adaptación de sus parámetros?

Melanie Mitchell, "**An Introduction to Genetic Algorithms**", The MIT Press, Cambridge, Massachusetts, USA, 1996.

Ajuste de Parámetros de un Algoritmo Genético

Críticas a la Auto-Adaptación

Dado que la información necesaria para auto-adaptar los parámetros proviene de la población misma, esta información podría no poder viajar suficientemente rápido como para reflejar con fidelidad el estado actual de la población.

De ahí que el uso de auto-adaptación en un algoritmo genético siga siendo objeto de controversia.

Ajuste de Parámetros de un Algoritmo Genético

Auto-Adaptación vs. Adaptación en Línea

Cabe mencionar que la auto-adaptación no es la única opción para permitir que un algoritmo genético ajuste por sí mismo sus parámetros. Otra opción es la denominada **adaptación en línea**.

La diferencia entre estos dos mecanismos es que cuando se usa auto-adaptación, los parámetros del algoritmo genético que queremos ajustar se codifican como variables adicionales del problema, de tal forma que se evolucionen usando el mecanismo de selección y los operadores de cruza y mutación.

En contraste, la adaptación en línea consiste en utilizar un mecanismo de monitoreo del proceso evolutivo para decidir en qué momento ajustar alguno de los parámetros del algoritmo genético.

Ajuste de Parámetros de un Algoritmo Genético

Adaptación en Línea

Srinivas y Patnaik [1994] propusieron un esquema para adaptar las probabilidades de cruza y mutación de un algoritmo genético.

La propuesta se basa en la detección de que el algoritmo genético ha convergido. Para ello, verifican qué diferencia existe entre la aptitud máxima de la población y la aptitud promedio. De tal forma, se hacen variar los porcentajes de cruza y mutación en función de esta diferencia de valores (aptitud máxima y aptitud promedio de la población).

M. Srinivas and L.M. Patnaik, "**Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms**", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 24, No. 4, pp. 656–667, April 1994.

Ajuste de Parámetros de un Algoritmo Genético

Mutaciones Variables

Varios investigadores han abordado el problema del ajuste del porcentaje de mutación de un algoritmo genético.

La idea de usar porcentajes de mutación dependientes del tiempo fue sugerida por Holland [1975], aunque no proporcionó una expresión en particular que describiera la variabilidad requerida.

Fogarty [1989] usó varias expresiones para variar p_m en las que se incluye el tiempo, logrando mejoras notables de desempeño. En ambos casos, la propuesta fue decrementar de manera determinística los porcentajes de mutación, de manera que tiendan a cero.

Mecanismos de Adaptación

Otra propuesta es la de Hesser y Männer [1991], en la cual se usa:

$$p_m(t) = \sqrt{\frac{\alpha}{\beta}} \frac{e^{-\gamma t/2}}{\lambda \sqrt{l}}$$

donde: $\lambda =$ tamaño de la población, $l =$ longitud cromosómica, $t =$ generación actual, α, β, γ son constantes definidas por el usuario (dependientes del problema).

Nótese que en todas estas propuestas se usa el mismo porcentaje de mutación para todos los individuos de la población en la generación t.

Jürgen Hesser and Reinhard Männer, "**Towards an Optimal Mutation Probability for Genetic Algorithms**", in H.-P. Schwefel and R. Männer (Editors), *Parallel Problem Solving from Nature, PPSN I*, pp. 23–32, Springer-Verlag, Lecture Notes in Computer Science Vol. 496, Berlin, Germany, 1991.

Mecanismos de Adaptación

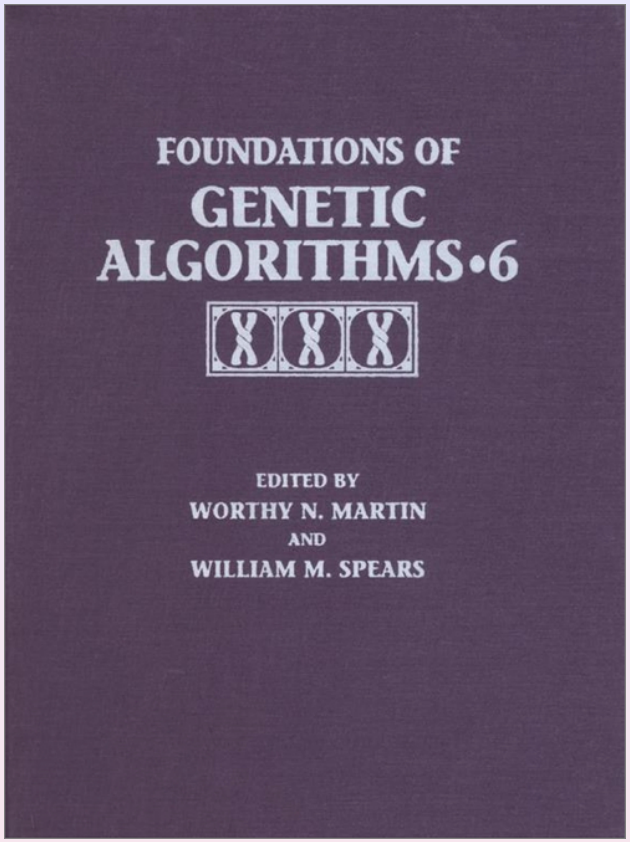
Bäck y Schütz [1996] propusieron un porcentaje de mutación que se decrementa usando:

$$p_m(t) = \frac{1}{2 + \frac{L-2}{T}t}$$

donde: $0 \leq t \leq T$, $L =$ longitud cromosómica, $t =$ generación actual y T es el número máximo de generaciones.

Thomas Bäck and Martin Schütz, "**Intelligent mutation rate control in canonical genetic algorithms**", in Zbigniew W. Raś and Maciek Michalewicz (Editors), *Foundations of Intelligent Systems ISMIS 1996*, pp. 158–167, Springer-Verlag, Lecture Notes in Computer Science Vol. 1079, Zakopane, Poland, June 9-13, 1996.

Ajuste de Parámetros de un Algoritmo Genético



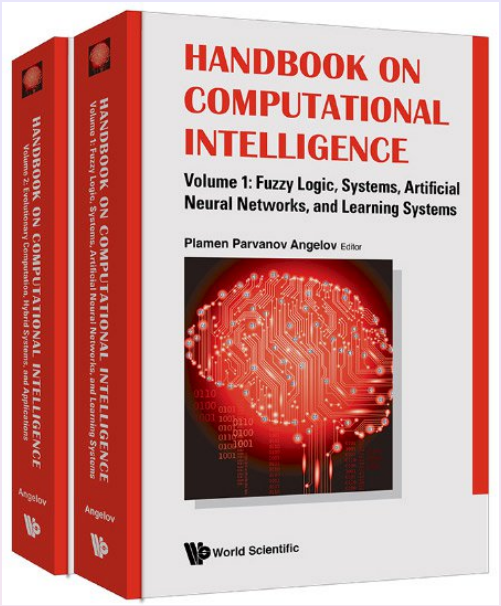
Mecanismos de Adaptación

La variabilidad obtenida con estra expresión es:

$p_m(0) = 0.5$

$p_m(T) = \frac{1}{L}$

Ajuste de Parámetros de un Algoritmo Genético



Mecanismos de Adaptación

Bäck [1992] sugirió el uso de un porcentaje de mutación que fuera función de la aptitud de cada individuo:

$$p_m(x) = \frac{1}{2(f(x)+1) - L}$$

Algoritmos Genéticos Adaptativos

Los objetivos principales de los algoritmos genéticos adaptativos son los siguientes (Herrera and Lozano, 1996]:

- Mantener diversidad en la población.
- Mejorar la convergencia del algoritmo genético, evitando a la vez la convergencia prematura.
- Evitar la disrupción de esquemas ocasionada por la cruza.

F. Herrera and M. Lozano, "**Adaptation of Genetic Algorithm Parameters Based on Fuzzy Logic Controllers**", in F. Herrera and J.L. Verdegay (Editors), *Genetic Algorithms and Soft Computing*, pp. 95–125, Physica-Verlag, 1996.

Ajuste de Parámetros de un Algoritmo Genético

Algoritmos Genéticos Adaptativos

Un algoritmo genético adaptativo incluye [Herrera and Lozano, 1996]:

- Ajuste adaptativo de parámetros (probabilidad de cruza y mutación, longitud del genotipo y tamaño de la población).
- Función de aptitud adaptativa.
- Operador de selección adaptativo.
- Operadores de búsqueda (variación) adaptativos.
- Representación adaptativa.

Ajuste de Parámetros de un Algoritmo Genético

Algoritmos Genéticos Adaptativos

El mecanismo de adaptación puede estar completamente separado del mecanismo de búsqueda del algoritmo genético.

Este tipo de esquema "**no acoplado**" no es muy atractivo, porque implica un control centralizado, superimpuesto al mecanismo de búsqueda del algoritmo genético.

Otra posibilidad es que el mecanismo de búsqueda del algoritmo genético sea usado parcialmente por el mecanismo adaptativo.

En este caso, se dice que el algoritmo genético y el mecanismo adaptativo están "**ligeramente acoplados**" (*loosely coupled*).

Ajuste de Parámetros de un Algoritmo Genético

Algoritmos Genéticos Adaptativos

Si la adaptación es conducida por las fuerzas internas de la búsqueda evolutiva, podemos hablar de un "**acoplamiento fuerte**".

En este caso, se origina un acoplamiento de los dos espacios de búsqueda sobre los que opera el algoritmo genético (el espacio de las soluciones y el de las variables de decisión).

Ajuste de Parámetros de un Algoritmo Genético

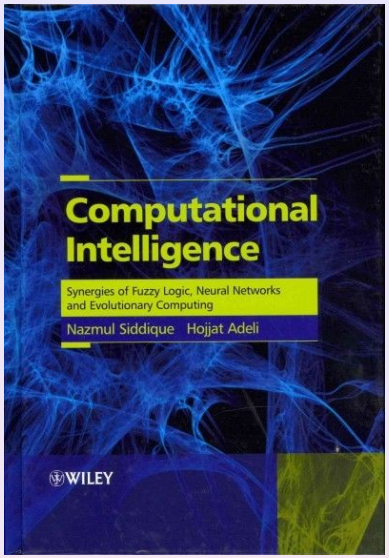
Técnicas Adaptativas Basadas en Lógica Difusa

Los controladores difusos se han usado con frecuencia como una técnica adaptativa para los algoritmos genéticos [Herrera y Lozano, 1996].

La integración de los algoritmos genéticos y los controladores difusos suele orientarse hacia los temas siguientes:

1. Elegir los parámetros del algoritmo genético antes de efectuar las ejecuciones correspondientes.
2. Ajustar los parámetros en línea, adaptándose a nuevas situaciones.
3. Asistir al usuario en detectar las soluciones emergentes útiles, en monitorear el proceso evolutivo con la finalidad de evitar convergencia prematura, y en diseñar algoritmos genéticos para una cierta tarea en particular.

Ajuste de Parámetros de un Algoritmo Genético



Representaciones Adaptativas

Se han propuesto varios esquemas en los que lo que se adapta es la representación usada con un algoritmo genético.

A continuación veremos dos propuestas muy interesantes de este tipo.

Ajuste de Parámetros de un Algoritmo Genético

Sistema ARGOT

Propuesto por Shaefer [1987], el método ARGOT es un algoritmo de búsqueda diseñado de tal manera que puede "aprender" la estrategia de búsqueda más adecuada.

ARGOT consiste de un algoritmo genético convencional al que se agregan varios operadores que modifican el mapeo intermedio que traduce los cromosomas en parámetros (o variables) de decisión.

Craig G. Shaefer, "**The ARGOT Strategy: Adaptive Representation Genetic Optimizer Technique**", in John J. Grefenstette (Editor), *Proceedings of the Second International Conference on Genetic Algorithms and their Application*, pp. 50–58, Lawrence Erlbaum Associates, Hilldale, New Jersey, USA, 1987.

Sistema ARGOT

Estos operadores se controlan por medio de tres tipos de medidas internas de la población:

(a) Medidas de convergencia (p.ej., la uniformidad de los cromosomas en un cierto lugar en particular).

(b) Medidas de posicionamiento (posición promedio relativa de las soluciones actuales con respecto a sus rangos extremos).

(c) Medidas de varianza (p.ej., el "ancho" de la distribución de las soluciones con respecto a los rangos permisibles).

Ajuste de Parámetros de un Algoritmo Genético

Sistema ARGOT

Estas medidas se aplican sobre cada gene del cromosoma y se usan para activar un cierto operador cuando resulta adecuado.

Los operadores incluyen uno que altera la resolución de un gene (número de bits empleados para representar una variable) y otros que mueven (*shift*), expanden y contraen el mapeo intermedio entre los cromosomas y las variables de decisión.

Estos cambios (expansión y contracción) hacen que los rangos de cada variable se modifiquen con la finalidad de focalizar la búsqueda y permiten también aplicar restricciones de desigualdad a las variables de decisión.

Sistema ARGOT

Además de los operadores primarios antes mencionados, ARGOT usa otros operadores secundarios.

Por ejemplo, se incluye un operador de mutación de Metropolis que acepta un cambio en un bit sólo si mejora la solución actual con la mutación. Si el cambio no mejora, se decide si se acepta o no el cambio usando una distribución de Boltzmann.

También se propuso un operador de homotopía (búsqueda local) para evitar convergencia a un óptimo local.

Codificación Delta

La idea de esta propuesta [Matthias & Whitley 1994] es cambiar dinámicamente la representación de un problema.

Nótese, sin embargo, que no intenta "aprender" cuál es la mejor representación del espacio de búsqueda, sino más bien se cambia la representación de manera periódica para evitar los sesgos asociados con una representación determinada del problema.

Keith E. Mathias and L. Darrell Whitley, "**Changing Representations During Search: A Comparative Study of Delta Coding** ", *Evolutionary Computation*, Vol. 2, No. 3, pp. 249–278, Fall 1994.

Codificación Delta

El algoritmo de la codificación delta empieza con la ejecución inicial de un algoritmo genético usando cadenas binarias.

Una vez que la diversidad de la población ha sido explotada adecuadamente, se almacena la mejor solución bajo el nombre de “solución temporal”.

Se reinicializa entonces el algoritmo genético con una nueva población aleatoria. En esta ocasión, sin embargo, las variables se decodifican de tal forma que representen una distancia o **valor delta** ($\pm\delta$) con respecto a la solución temporal.

Ajuste de Parámetros de un Algoritmo Genético

Codificación Delta (Ejemplo)

Parámetros numéricos	0	1	2	3	4	5	6	7
Codificación binaria	000	001	010	011	100	101	110	111
Cambios numéricos	0	1	2	3	-3	-2	-1	-0
Codificación delta	000	001	010	011	111	110	101	100

Ajuste de Parámetros de un Algoritmo Genético

Tamaño de Población Adaptativo

Smith [1993] propuso un algoritmo que ajusta su tamaño de población con respecto a la probabilidad del error de selección.

Hinterding et al. [1996] experimentaron con un algoritmo genético adaptativo, el cual consistía de 3 subpoblaciones. A intervalos regulares, los tamaños de estas subpoblaciones se ajustaban con base en el estado actual de la búsqueda.

Robert E. Smith and Ellen Smuda, "**Adaptively resizing populations: Algorithm, analysis, and first results**", *Complex Systems*, Vol. 9, pp. 47–72, 1995.

R. Hinterding, Z. Michalewicz and T. C. Peachey, "**Self-adaptive genetic algorithm for numeric functions**", in H.-M. Voigt, W. Ebeling, I. Rechenberg and H.-P. Schwefel (Editors), *Proceedings of the 4th Conference on Parallel Problem Solving from Nature*, pp. 420–429, Springer, Lecture Notes in Computer Science Vol. 1141, Berlin, Germany, 1996.

Tamaño de Población Adaptativo

Schlierkamp-Voosen and Mühlenbein [1994] usaron un esquema de competencia que cambia los tamaños de las subpoblaciones, manteniendo fijo el número total de individuos.

D. Schlierkamp-Voosen and H. Mühlenbein, "**Strategy adaptation by competing subpopulations**", in H.-P. Schwefel and R. Männer (Editors), *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, pp. 199–208, Springer-Verlag, Lecture Notes in Computer Science Vol. 866, New York, USA, 1994.

Tamaño de Población Adaptativo

Arabas et al. [1994] propusieron un algoritmo genético con tamaño de población variable (GAVaPS), que usa el concepto de "edad" de un cromosoma, el cual es equivalente al número de generaciones que un individuo ha permanecido "vivo". En otras palabras, la edad del cromosoma reemplaza el concepto de selección y, puesto que depende de la aptitud del individuo, ejerce influencia sobre el tamaño de la población en cada etapa del proceso.

J. Arabas, Z. Michalewicz, and J. Mulawka, "**GAVaPS–A genetic algorithm with varying population size**", in *Proceedings of the 2nd IEEE Confence on Evolutionary Computation*, pp. 73–78, IEEE Press, 1994.

Ajuste de Parámetros Fuera de Línea

El ajuste de parámetros fuera de línea es computacionalmente costoso porque debe evaluarse un gran número de configuraciones. Sin embargo, es útil para derivar conocimiento sobre las relaciones entre los parámetros de un algoritmo evolutivo.

El ajuste de parámetros fuera de línea puede abordarse de dos formas:

1. La especialización de algoritmos (dada la configuración de un algoritmo, el objetivo es encontrar un subconjunto de problemas de optimización) y
2. la generalización de algoritmos (el objetivo es encontrar una configuración algorítmica para resolver una mayor cantidad de problemas con características diferentes).

A.E. Eiben and S. K. Smit, "**Parameter tuning for configuring and analyzing evolutionary algorithms**", *Swarm and Evolutionary Computation*, Vol. 1, No. 1, pp. 19–31, March 2011.

Ajuste de Parámetros Fuera de Línea

A lo largo de los años, se han propuesto varios métodos de ajuste de parámetros fuera de línea, así como herramientas para la configuración automática de algoritmos que han sido diseñadas para buscar la configuración más adecuada de los valores de los parámetros de un algoritmo estocástico de búsqueda.

Estas técnicas pueden clasificarse en 4 grupos:

1. Diseño experimental
2. Basadas en modelos
3. Pruebas estadísticas secuenciales
4. Optimización heurística

A continuación veremos en qué consiste cada uno de estos grupos de técnicas, así como esquemas representativos de ellas.

Ajuste de Parámetros de un Algoritmo Genético

Diseño Experimental

A lo largo de los años se han propuesto diversas técnicas basadas en el Diseño de Experimentos, tales como: el análisis de varianza (ANOVA), el uso de intervalos de confianza, el diseño factorial, el diseño factorial fraccionario y los modelos de regresión lineal y no lineales.

Estas técnicas tienen dos objetivos:

1. Diseñar un experimento para recolectar datos adecuados y
2. analizar dichos datos usando métodos estadísticos para derivar conclusiones válidas y objetivas.

Estos dos temas están fuertemente ligados, porque el método de análisis depende directamente del diseño experimental adoptado. Además, las técnicas de diseño de experimentos se usan para construir un modelo predictivo del desempeño de un algoritmo sobre un rango de conjuntos de parámetros e instancias dadas.

D.C. Montgomery, "**Design and Analysis of Experiments**", John Wiley & Sons, USA, 2008.

Ajuste de Parámetros de un Algoritmo Genético

Diseño Experimental

Adenso-Diaz et al. [2006] propusieron un algoritmo denominado CALIBRA, que utiliza la metodología Taguchi en un diseño factorial nivel 2 acoplado con un procedimiento de búsqueda local.

En un diseño factorial, el primer paso es seleccionar los parámetros que influyen más en el desempeño del algoritmo evaluado. Posteriormente, se establecen dos o tres valores críticos para estos parámetros y se evalúan todas las combinaciones de ellos.

B. Adenso-Diaz and M. Laguna, "**Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search**", *Operations Research*, Vol. 54, No. 1, pp. 99–114, January 2006.

Basadas en Modelos

La técnica basada en modelos más representativa del área es la metodología denominada Optimización Secuencial de Parámetros desarrollada por Bartz-Beielstein [2006]. Esta metodología se divide en tres fases.

En el primer paso, el experimentador define exactamente lo que se estudiará y la forma en que los datos se colectarán. La segunda fase incluye el diseño de experimentos así como el modelado estadístico y el método de predicción a utilizarse. Los elementos que deben definirse son los siguientes:

- un método de inicialización y terminación,
- un algoritmo y sus factores importantes,
- una métrica para medir el desempeño.

Finalmente, el tercer paso se refiere a la aceptación o rechazo de la hipótesis estadística y a la interpretación de los resultados.

T. Bartz-Beielstein, "**Experimental Research in Evolutionary Computation: The New Experimentalism**", Springer-Verlag, New York, USA, 2006.

Pruebas Estadísticas Secuenciales

El concepto de "carrera" (*racing*) fue introducido en 1997 como una técnica de aprendizaje de máquina cuyo objetivo es decrementar el costo computacional de estimar la calidad de un conjunto de configuraciones de parámetros.

En este procedimiento, un conjunto de configuraciones de parámetros compiten en una carrera utilizando un conjunto de problemas de prueba. Se identifican las peores configuraciones y se les descarta usando evidencia estadística.

Este tipo de técnicas son útiles cuando intentamos ajustar un gran número de parámetros. La idea fundamental de este tipo de esquema es evaluar el desempeño de una configuración candidata incrementalmente, asignando recursos computacionales sólo a las configuraciones prometedoras, en vez de desperdiciar tiempo con las peores configuraciones.

O. Maron and A.W. Moore, "**The Racing Algorithm: Model Selection for Lazy Learners**", *Artificial Intelligence Review*, Vol. 11, No. 1, pp. 193–225, February 1997.

Pruebas Estadísticas Secuenciales

Birattari et al. [2002] diseñaron uno de los primeros esquemas de "carreras" reportados en la literatura de computación evolutiva, al que se le conoce como F-race. Este esquema utiliza el análisis de Friedman no paramétrico bi-direccional de varianza basado en jerarquías.

En F-race la validación estadística se basa en la jerarquía de las configuraciones de parámetros candidatas. F-race busca encontrar evidencia de que al menos una de las configuraciones es significativamente diferente de las demás. De no suceder esto, las pruebas de Friedman se aplican nuevamente para eliminar las configuraciones candidatas que son significativamente peores que la mejor.

M. Birattari, T. Stützle, L. Paquete and K.A. Varrentrapp, "**Racing Algorithm for Configuring Metaheuristics**", in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2007)*, pp. 11–18, Morgan Kaufmann Publishers, San Francisco, California, USA, 2002.

Ajuste de Parámetros de un Algoritmo Genético

Pruebas Estadísticas Secuenciales

Balaprakash et al. [2007] propusieron *Iterated F-race* (Irace). En esta técnica, a cada iteración, se construye un conjunto de nuevas configuraciones candidatas utilizando un modelo de probabilidad; posteriormente, se evalúan estas configuraciones y se selecciona la mejor mediante una carrera. Finalmente, la distribución de muestras se actualiza, tendiendo hacia las mejores configuraciones.

P. Balaprakash, M. Birattari and T. Stützle, "**Improvement Strategies for the F-Race Algorithm: Sampling Design and Iterative Refinement**", in C. Blum et al. (Editors), *Hybrid Metaheuristics*, pp. 108–122, Springer, Berlin, 2007.

Ajuste de Parámetros de un Algoritmo Genético

Optimización Heurística

El último grupo consiste en técnicas de optimización meta-nivel que trabajan en un espacio de búsqueda definido por los parámetros involucrados en el algoritmo estudiado.

El meta-algoritmo genético de Grefenstette que vimos anteriormente (el cual consiste en un algoritmo genético que optimiza los parámetros de otro algoritmo genético), pertenece a este grupo de técnicas.

John J. Grefenstette, "**Optimization of Control Parameters for Genetic Algorithms**", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-16, No. 1, pp. 122–128, January/February 1986.

Ajuste de Parámetros de un Algoritmo Genético

Optimización Heurística

Otro ejemplo de este tipo de técnica es el método para Estimación de Relevancia y Calibración de Valor de parámetros de algoritmos evolutivos [Nannen and Eiben, 2006].

Esta técnica determina los valores más apropiados de los parámetros vía una distribudad de densidad de probabilidad con entropía de Shannon maximizada, usando un algoritmo de Estimación de Distribución (EDA).

Esta técnica es capaz de determinar la sensibilidad de los valores de los parámetros a fin de establecer los rangos recomendados a utilizarse en el algoritmo evolutivo bajo estudio.

V. Nannen and A. Eiben, "**A Method for Parameter Calibration and Relevance Estimation in Evolutionary Algorithms**", in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'2006)*, pp. 183–190, ACM Press, New York, USA, 2006.

Ajuste de Parámetros de un Algoritmo Genético

Optimización Heurística

Hutter et al. [2009] propusieron ParamILS que se basa en un algoritmo de estado uniforme que usa un procedimiento de búsqueda local iterativa para mejorar el desempeño de configuraciones algorítmicas.

ParamILS requiere una configuración inicial y un conjunto de valores posibles para cada uno de los parámetros. Este algoritmo es muy sensible al valor inicial que adopta. Usar valores diferentes produce resultados diferentes. Por ello, se usa para mejorar configuraciones adoptadas previamente y no para configuraciones generadas aleatoriamente.

F. Hutter, H.H. Hoos, K. Leyton-Brown and T. Stützle, "**ParamILS: An Automatic Algorithm Configuration Framework**", *Journal of Artificial Intelligence Research*, Vol. 36, No. 1, pp. 267–306, September 2009.

Hacia una Taxonomía de Técnicas de Control de Parámetros

Eiben and Michalewicz [1999] propusieron la siguiente taxonomía de técnicas de control de parámetros, con base en los aspectos que se suelen tomar en cuenta:

- **¿Qué se cambia?** (p.ej., representación, función de evaluación, operadores, proceso de selección, porcentaje de mutación, etc.).
- **¿Cómo se realiza el cambio?** (p.ej., heurística determinista, heurística basada en retroalimentación o auto-adaptativa).
- **Alcance/Nivel del Cambio** (p.ej., poblacional, individual, etc.).
- **Evidencia en la que se basa el cambio** (p.ej., monitoreo del desempeño, diversidad de la población, etc.).

Ágoston Endre Eiben, Robert Hinterding and Zbigniew Michalewicz, "**Parameter Control in Evolutionary Algorithms**", *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 2, pp. 124–141, July 1999.

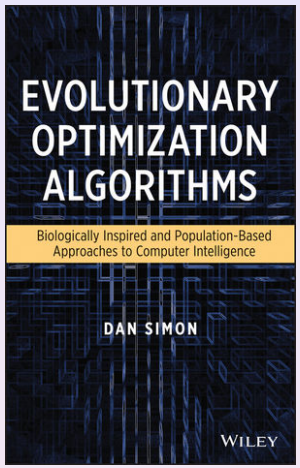
Hacia una Taxonomía de Técnicas de Control de Parámetros

El aspecto principal de esta taxonomía es el **¿cómo?**, de acuerdo al cual tenemos tres categorías:

1. **Control Determinista de Parámetros:** Toma lugar cuando el valor de un parámetro se altera usando una regla determinista. Esta regla modifica el valor sin usar ninguna retroalimentación del proceso de búsqueda. Usualmente, se usa una variabilidad dependiente del tiempo (o sea, se efectúa un cambio cada cierto número de generaciones).

Hacia una Taxonomía de Técnicas de Control de Parámetros

2. **Control Adaptativo de Parámetros:** Toma lugar cuando se usa algún tipo de retroalimentación de la búsqueda para determinar la dirección y/o magnitud del cambio de un parámetro. La asignación del valor del parámetro puede involucrar un sistema de asignación de crédito (o recompensas) y la acción del algoritmo evolutivo puede determinar si el nuevo valor persiste o no, o si éste se propagará a través de la población.



Hacia una Taxonomía de Técnicas de Control de Parámetros

3. **Control Auto-Adaptativo de Parámetros:** En este caso, los parámetros a adaptarse se codifican en el cromosoma y se someten a cruza y mutación. Los mejores valores de esos parámetros codificados nos conducen a mejores individuos, lo que, a su vez, hace más probable que éstos sobrevivan y generen hijos que propaguen dichos valores.

Comentarios Finales sobre Control de Parámetros

Un punto debatible respecto al control de parámetros es: ¿cuánto esfuerzo en este sentido vale la pena llevar a cabo? En otras palabras: ¿qué costos computacionales son aceptables?

Algunos investigadores argumentan que el control adaptativo, en general, complica sustancialmente la labor de un algoritmo genético y que las mejoras en la calidad de la solución no son significativas como para justificar el costo extra.

Fernando G. Lobo, Cláudio F. Lima and Zbigniew Michalewicz (Editors), **Parameter Setting in Evolutionary Algorithms**, Springer, Berlin, 2007, ISBN 978-3-540-69431-1.

Comentarios Finales sobre Control de Parámetros

Resulta claro que los mecanismos de control adaptativo y auto-adaptativo tienen un costo asociado.

Por ejemplo, requieren que se colecten estadísticas durante una ejecución del algoritmo o que se realicen operaciones adicionales sobre los individuos.

En consecuencia, realizar comparaciones entre un algoritmo que usa auto-adaptación (o adaptación en línea) contra otro que no lo haga, no es algo justo, porque se ignora el tiempo que se dedica al ajuste de los parámetros.

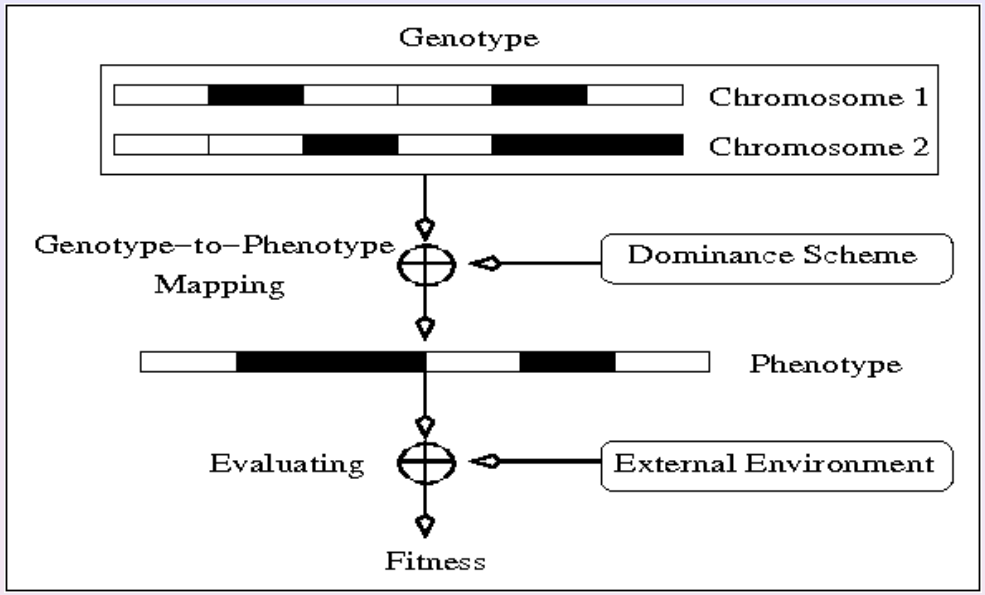
Comentarios Finales sobre Control de Parámetros

Los mecanismos de control de parámetros en línea pueden tener gran utilidad en ambientes no estacionarios.

En ese caso, frecuentemente es necesario modificar la solución actual debido a varios cambios del ambiente.

La capacidad de un algoritmo genético de considerar tales cambios y de rastrear eficientemente el óptimo, ha sido estudiada por varios autores (p.ej., [Angeline et al., 1996], [Vavak et al., 1997]).

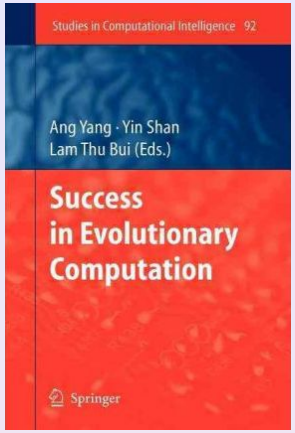
Diploides y Dominancia



En los algoritmos genéticos, se usan normalmente cromosomas haploides.

En la naturaleza, sin embargo, los genotipos de una gran cantidad de seres vivos son ser diploides y contienen uno o más pares de cromosomas (a los que se les llama **homólogos**), cada uno de los cuales contiene información (redundante) para las mismas funciones.

Diploides y Dominancia



Ejemplo de un cromosoma diploide:

AbCDefGhIj
aBCdeFgHij

Si suponemos que los genes representados por letras mayúsculas son los **dominantes** y los representados mediante letras minúsculas son los **recesivos**, entonces el fenotipo correspondiente al cromosoma anterior sería:

ABCDeFGHIj

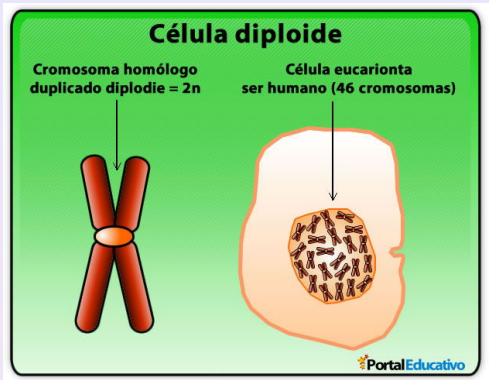
Diploides y Dominancia

El operador utilizado en el ejemplo anterior se denomina **dominancia**.

La idea es que un alelo (o un gen) dominante toma precedencia sobre uno recesivo, tal y como ocurre en la Naturaleza (por ejemplo, los ojos negros son un alelo dominante y los azules uno recesivo).

A un nivel más abstracto, podemos concebir a la dominancia como un mapeo reductor del genotipo hacia el fenotipo.

Diploides y Dominancia



¿Por qué usa esta redundancia la naturaleza?

Existen varias teorías al respecto de parte de los biólogos.

Las teorías biológicas más aceptadas, sugieren que los diploides son una especie de “registro histórico” que protegen ciertos alelos (y combinaciones de ellos) del daño que puede causar la selección en un ambiente hostil.

Diploides y Dominancia

En los algoritmos genéticos, los diploides suelen usarse para mantener soluciones múltiples (al mismo problema), las cuales pueden conservarse a pesar de que se exprese sólo una de ellas.

La idea es la misma que en Biología: preservar soluciones que fueron efectivas en el pasado, pero que eliminó el mecanismo de selección del algoritmo genético.

Los diploides parecen ser particularmente útiles en problemas en los que el ambiente cambia con el paso de las generaciones (por ejemplo, optimización de funciones dinámicas).

Diploides y Dominancia

El siguiente ejemplo se debe a Hillis [1990]:
Padre 1 (diploide):

A: 10110101 | 011110011110010010101001

B: 00000101 | 001001110011110010101001

Padre 2 (diploide): ↓

C: 0000011100000111110 | 000010101011

D: 11111111000010101101 | 010111011100

Hijo (diploide): ↓

10110101001001110011110010101001

00000111000000111110010111011100

W. Daniel Hillis, "**Co-evolving parasites improve simulated evolution as an optimization procedure**", *Physica D: Nonlinear Phenomena*, Vol. 42, Nos. 1-3, pp. 228–234, June 1990.

Diploides y Dominancia

En este ejemplo, el genotipo de un individuo consiste de 15 pares de cromosomas (por claridad, sólo se muestra un par de cromosomas por cada padre en la figura del acetato anterior).

Se elige aleatoriamente un punto de cruza para cada par, y se forma un gameto tomando los alelos antes del punto de cruza en el primer cromosoma, y los alelos después del punto de cruza en el segundo.

Los 15 gametos de un padre se unen con los 15 gametos del otro padre para formar un nuevo individuo diploide (nuevamente, por claridad, sólo un gameto se muestra en la figura del acetato anterior).

Inversión

Holland [1975] propuso formas de adaptar la codificación de su algoritmo genético original, pues advirtió que el uso de cruza de un punto no trabajaría correctamente en algunos casos.

John H. Holland, "**Adaptation in Natural and Artificial Systems**", University of Michigan Press, Ann Arbor, Michigan, USA, 1975.

Inversión

El operador de **inversión** es un operador de reordenamiento inspirado en una operación que existe en genética.

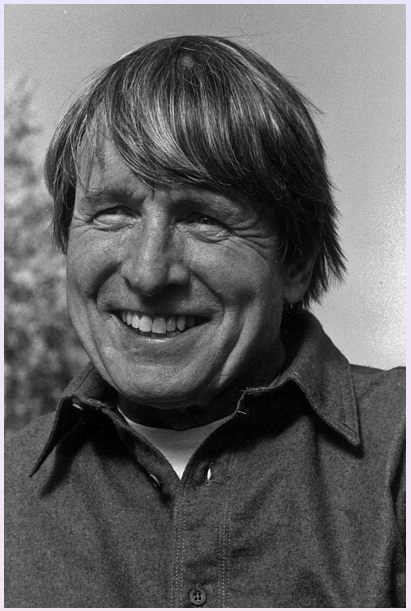
A diferencia de los algoritmos genéticos simples, en genética la función de un gene es frecuentemente independiente de su posición en el cromosoma (aunque frecuentemente los genes en un área local trabajan juntos en una red regulatoria), de manera que invertir parte del cromosoma retendrá mucha (o toda) la "semántica" del cromosoma original.

Inversión

Para usar inversión en los algoritmos genéticos, tenemos que encontrar la manera de hacer que la interpretación de un alelo sea la misma sin importar la posición que guarde en una cadena.

Con esa finalidad, Holland propuso que a cada alelo se le diera un índice que indicara su posición “real”, la cual se utilizaría para evaluar su aptitud.

Inversión



Por ejemplo, la cadena 00010101 se codificaría como:

$$\{ \ (1,0) \quad (2,0) \quad (3,0) \quad (4,1) \quad (5,0) \quad (6,1) \quad (7,0) \quad (8,1) \ \}$$

En donde el primer elemento de cada uno de estos pares proporciona la posición “real” del alelo dado.

Inversión

La inversión funciona tomando dos puntos (aleatoriamente) a lo largo de la cadena, e invirtiendo la posición de los bits entre ellos.

Por ejemplo, si usamos la cadena anterior, podríamos escoger los puntos **3** y **6** para realizar la inversión; el resultado sería:

$$\{ \ (1,0) \quad (2,0) \quad (6,1) \quad (5,0) \quad (4,1) \quad (3,0) \quad (7,0) \quad (8,1) \ \}$$

Inversión

Nótese que esta nueva cadena tiene la misma aptitud que la anterior porque los índices siguen siendo los mismos.

¿Cuál es entonces el propósito de la inversión?

Holland indicaba que este operador permitiría cambiar los enlaces alélicos.

La idea de este operador es producir ordenamientos en los cuales los esquemas benéficos puedan sobrevivir con mayor facilidad.

Inversión

Por ejemplo, supongamos que en el ordenamiento original el esquema **00**01**** es muy importante.

Tras usar el operador de inversión, el esquema nuevo será **0010******.

Si este nuevo esquema tiene una aptitud más alta, presumiblemente la cruza de un punto lo preservará y esta permutación tenderá a diseminarse con el paso de las generaciones.

Inversión

Debe advertirse que el uso del operador de inversión introduce nuevos problemas cuando se combina con la cruza de un punto.

Supongamos, por ejemplo, que se cruzan las cadenas:

$$\{ \ (1,0) \quad (2,0) \quad (6,1) \quad (5,0) \quad (4,1) \quad (3,0) \quad (7,0) \quad (8,1) \ \}$$

y

$$\{ \ (5,1) \quad (2,0) \quad (3,1) \quad (4,1) \quad (1,1) \quad (8,1) \quad (6,0) \quad (7,0) \ \}$$

Inversión

Si el punto de cruza es la tercera posición, los hijos producidos serán:

$$\{ \ (1,0) \quad (2,0) \quad (6,1) \quad (4,1) \quad (1,1) \quad (8,1) \quad (6,0) \quad (7,0) \ \}$$

y

$$\{ \ (5,1) \quad (2,0) \quad (3,1) \quad (5,0) \quad (4,1) \quad (3,0) \quad (7,0) \quad (8,1) \ \}$$

Inversión

Evidentemente, estas nuevas cadenas tienen algo mal.

La primera cadena tiene 2 copias de los bits **1** y **6** y ninguna copia de los bits **3** y **5**.

La segunda tiene 2 copias de los bits **3** y **5** y ninguna copia de los bits **1** y **6**.

¿Cómo podemos asegurarnos de que este problema no se presente?

Inversión

Holland propuso 2 soluciones posibles:

(1) Permitir que se realice la cruza sólo entre cromosomas que tengan los índices en el mismo orden. Esto funciona pero limitaría severamente la cruza.

(2) Emplear un enfoque "amo/esclavo": Escoger un padre como el amo, y reordenar temporalmente al otro padre para que tenga el mismo ordenamiento que su amo. Usando este tipo de ordenamiento se producirán cadenas que no tendrán redundancia ni posiciones faltantes.

Inversión

La inversión se usó en algunos trabajos iniciales con algoritmos genéticos, pero nunca mejoró dramáticamente el desempeño de un algoritmo genético.

Con el paso de los años comenzó a utilizarse una versión de este operador que sí altera los valores de aptitud, sobre todo en problemas de optimización combinatoria, teniendo resultados un poco mejores.

Sin embargo, en la actualidad, no existe consenso en torno a la utilidad real de este operador.

Adicionalmente, la versión de este operador que propuso Holland requiere de espacio extra para almacenar los índices de cada bit, así como de tiempo extra para reordenar un padre antes de efectuar la cruza.

Micro-Operadores

En la Naturaleza, muchos organismos tienen genotipos con múltiples cromosomas.

Por ejemplo, los seres humanos tenemos 23 pares de cromosomas diploides.

Para adoptar una estructura similar en los algoritmos genéticos necesitamos extender la representación a fin de permitir que un genotipo sea una lista de k pares de cadenas (suponiendo que son diploides).

Micro-Operadores

Pero, ¿para qué tomarnos estas molestias?

Holland [1975] sugirió que los genotipos con múltiples cromosomas podrían ser útiles para extender el poder de los algoritmos genéticos cuando se usan en combinación con dos operadores: **la segregación** y **la traslocación**.

John H. Holland, "**Adaptation in Natural and Artificial Systems**", University of Michigan Press, Ann Arbor, Michigan, USA, 1975.

Segregación

Para entender cómo funciona este operador, imaginemos el proceso de formación de gametos cuando tenemos más de un par cromosómico en el genotipo.

La cruza se efectúa igual que como vimos antes, pero cuando formamos un gameto, tenemos que seleccionar aleatoriamente uno de los cromosomas haploides.

A este proceso de selección aleatoria se le conoce como **segregación**. Este proceso rompe cualquier enlace que pueda existir entre los genes dentro de un cromosoma, y es útil cuando existen genes relativamente independientes en cromosomas diferentes.

R.B. Hollstein, "**Artificial genetic adaptation in computer control systems**", PhD thesis, University of Michigan, Ann Arbor, Michigan, USA, 1971.

Traslocación

Puede verse como un operador de cruza intercromosómico.

Para implementar este operador en un algoritmo genético necesitamos asociar los alelos con su "nombre genético" (su posición), de manera que podamos identificar su significado cuando se cambien de posición de un cromosoma a otro mediante la traslocación.

El uso de este operador permite mantener la organización de los cromosomas de manera que la segregación pueda explotar tal organización.

La segregación y la traslocación no se han usado mucho en la práctica, excepto por algunas aplicaciones de aprendizaje de máquina [p.ej., Schaffer, 1984; Smith, 1980].

R.E. Smith, "**An investigation of diploid genetic algorithms for adaptive search of nonstatonary functions**", Masters thesis, The University of Alabama, Tuscaloosa, Alabama, USA, 1988.

Micro-Operadores

Duplicación y Borrado

Estos son dos operadores de bajo nivel sugeridos para la búsqueda artificial efectuada por el algoritmo genético.

La **duplicación intracromosómica** produce duplicados de un gene en particular y lo coloca junto con su progenitor en el cromosoma.

El **borrado** actúa a la inversa, removiendo un gene duplicado del cromosoma.

Micro-Operadores

Duplicación y Borrado

Holland [1975] sugirió que estos operadores pueden ser métodos efectivos de controlar adaptativamente el porcentaje de mutación.

Si el porcentaje de mutación permanece constante y la duplicación ocasiona k copias de un gene en particular, la probabilidad de mutación efectiva para este gene se multiplica por k.

Por otra parte, cuando ocurre el borrado de un gene, el porcentaje efectivo de mutación se decrementa.

Duplicación y Borrado

Cabe mencionar que una vez que ha ocurrido una mutación en uno de los nuevos genes, debemos decidir cuál de las alternativas será la que se exprese, en un proceso similar al que enfrentamos con los diploides.

De hecho, podemos considerar la presencia de múltiples copias de un gen como una **dominancia intracromosómica**, en contraposición con la **dominancia intercromosómica** que resulta más tradicional en los diploides.

Duplicación y Borrado

Holland [1975] sugirió el uso de un esquema de arbitraje para hacer la elección necesaria entre las diferentes alternativas presentes, aunque no se han publicado estudios sobre este mecanismo hasta la fecha.

La duplicación puede permitir cosas más interesantes en un algoritmo genético, como por ejemplo, el uso de cadenas de longitud variable (como en los algoritmos genéticos desordenados).

Motivación

Las técnicas tradicionales de programación matemática que se usan para resolver problemas de optimización con restricciones tienen varias limitantes cuando tienen que lidiar con el problemas general de optimización no lineal:

$$\text{Min } f(\vec{x}) \tag{1}$$

sujeto a:

$$g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, m \tag{2}$$

$$h_j(\vec{x}) = 0, j = 1, \ldots, p \tag{3}$$

donde $\vec{x}$ es el vector de variables de decisión, $\vec{x} = [x_1, x_2, \ldots, x_n]^T$, m es el número de restricciones de desigualdad y p es el número de restricciones de igualdad (en ambos casos, las restricciones pueden ser lineales o no lineales).

Técnicas de Manejo de Restricciones

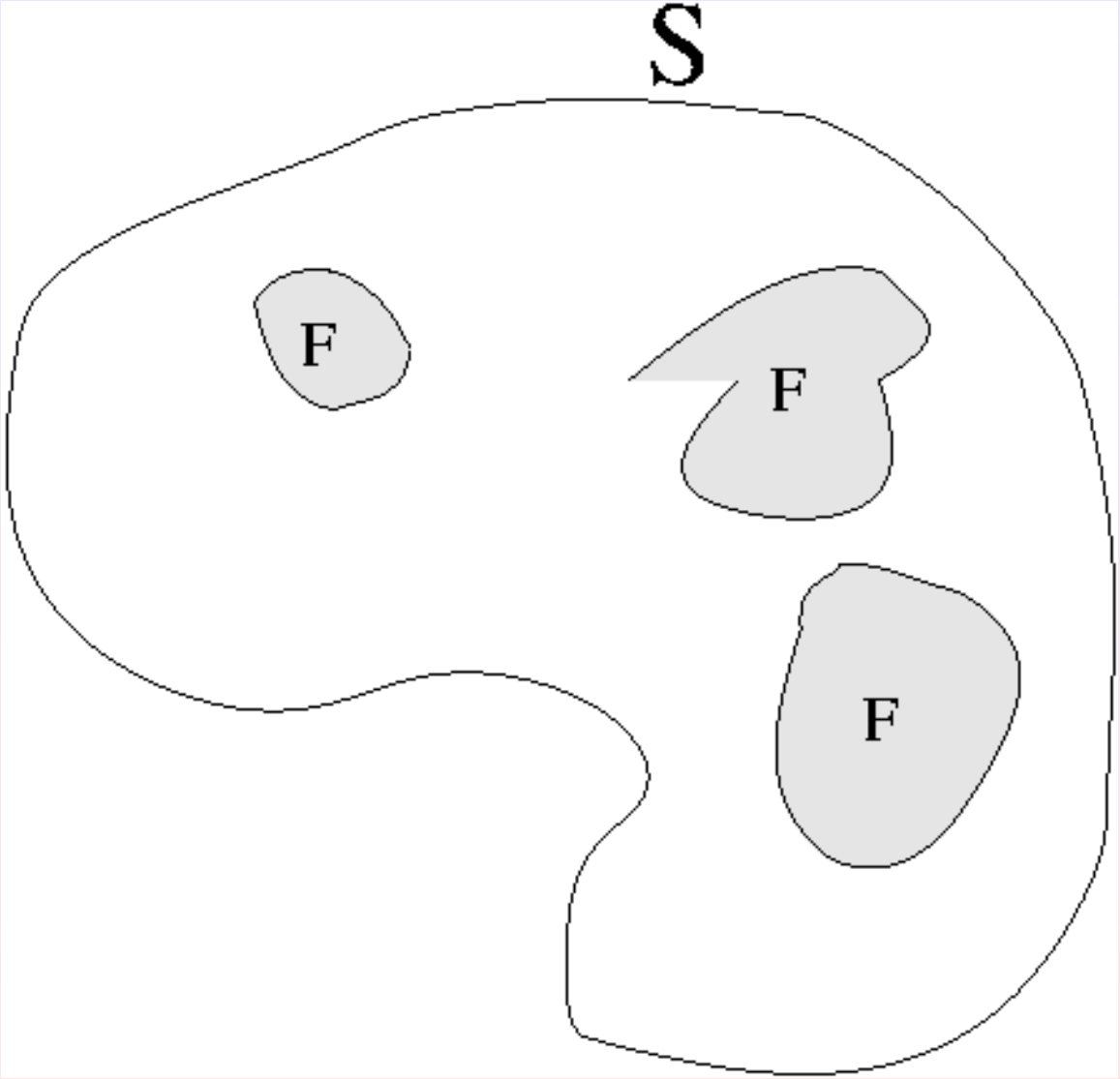
Motivación

Los algoritmos genéticos han resultado muy exitosos en la solución de una amplia variedad de problemas de optimización.

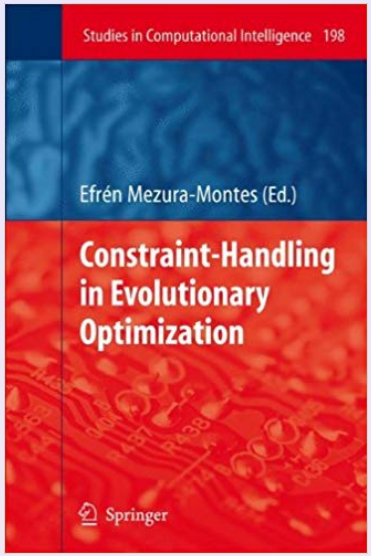
Sin embargo, como técnicas de optimización, los algoritmos genéticos son técnicas de búsqueda sin restricciones. Por ende, la incorporación de un mecanismo de manejo de restricciones a un algoritmo genético es un área abierta de investigación.

Actualmente, existe una cantidad considerable de trabajo en torno a este tema, el cual revisaremos brevemente a continuación

Técnicas de Manejo de Restricciones



Técnicas de Manejo de Restricciones



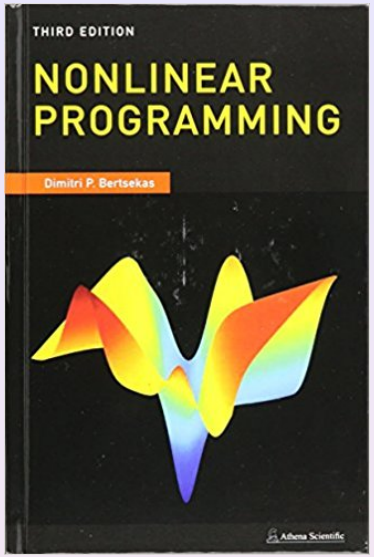
Una Taxonomía de Métodos para Manejo de Restricciones

- Funciones de penalización
- Representaciones y operadores especiales
- Algoritmos de reparación
- Separación de objetivos y restricciones
- Métodos híbridos

Funciones de Penalización

Una de las técnicas más populares de manejo de restricciones usadas con algoritmos genéticos es el uso de funciones de penalización. Esta técnica fue propuesta originalmente por Richard Courant en los 1940s y fue extendida posteriormente por Carroll y Fiacco & McCormick.

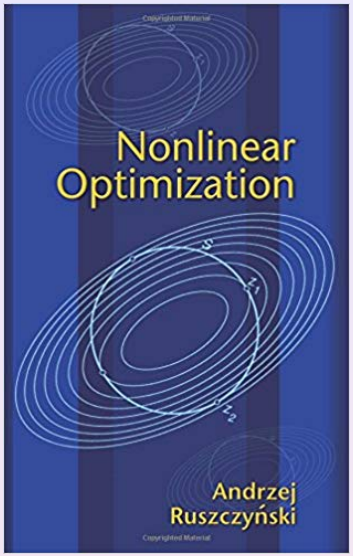
Funciones de Penalización



La idea fundamental de las funciones de penalización es transformar un problema de optimización con restricciones en uno sin restricciones, sumándole (o restándole) un cierto valor a/de la función objetivo.

Dicho valor se basa en la cantidad de violación de las restricciones presente en una solución.

Funciones de Penalización



En programación matemática, se consideran dos tipos de funciones de penalización: exteriores e interiores.

En el caso de los métodos de penalización **exteriores**, comenzamos con una solución infactible y nos movemos de ahí hacia la región factible.

Funciones de Penalización

En el caso de los métodos de penalización **interiores**, se elige el factor de penalización de tal forma que su valor sea pequeño en puntos lejos de la frontera entre la región factible e infactible y que tiendan a infinito conforme nos acercamos a dicha frontera.

De tal forma, si empezamos de un punto factible, los puntos subsecuentes que se generen, siempre permanecerán dentro de la región factible, puesto que la frontera con la región infactible actúa como una barrera durante el proceso de optimización

Funciones de Penalización

Penalizaciones Exteriores

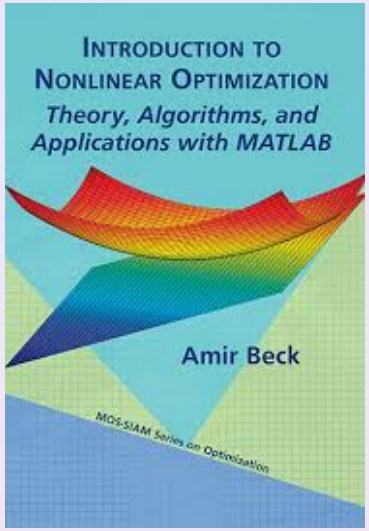
Dado que es muy complicado suponer que podemos contar con una solución factible para iniciar la búsqueda, los algoritmos genéticos suelen adoptar funciones de penalización externa.

Dichas funciones de penalización tienen la forma siguiente:

$$\phi(\vec{x}) = f(\vec{x}) \pm \left[\sum_{i=1}^{n} r_i \times G_i + \sum_{j=1}^{p} c_j \times L_j \right] \quad (4)$$

donde $\phi(\vec{x})$ es la función objetivo (expandida) a ser optimizada, G_i y L_j son funciones definidas en términos de las restricciones $g_i(\vec{x})$ y $h_j(\vec{x})$, respectivamente, r_i y c_j son constantes positivas normalmente llamada "factores de penalización".

Funciones de Penalización



Penalizaciones Exteriores

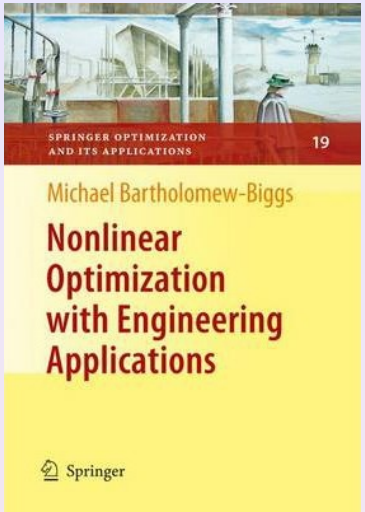
La forma más común de G_i y L_j es:

$$G_i = \max[0, g_i(\vec{x})]^{\beta} \tag{5}$$

$$L_j = |h_j(\vec{x})|^{\gamma} \tag{6}$$

donde: β y γ son normalmente 1 o 2.

Funciones de Penalización



Las funciones de penalización pueden manejar tanto restricciones de igualdad como de desigualdad, y el enfoque normalmente adoptado es transformar las restricciones de igualdad en restricciones de desigualdad, usando la siguiente expresión:

$$|h_j(\vec{x})| - \epsilon \leq 0 \tag{7}$$

donde ϵ es la tolerancia permitida (un valor muy pequeño).