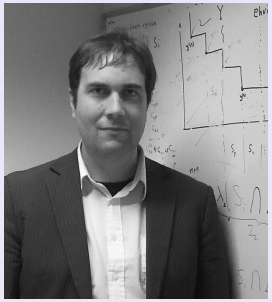
# Introducción a la Computación Evolutiva

Carlos A. Coello Coello

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

**Clase 15**



## El Hipervolumen

Cabe destacar que el uso del hipervolumen para seleccionar soluciones no es del todo trivial. Este indicador opera sobre un conjunto de soluciones, y el operador de selección sólo considera una solución a la vez. Por lo tanto, cuando se utiliza el hipervolumen para seleccionar soluciones, se requiere una estrategia de asignación de aptitud.

La estrategia que se ha adoptado más comúnmente en la literatura especializada consiste en realizar primero un ordenamiento no dominado y posteriormente jerarquizar las soluciones dentro de cada frente con base en la pérdida de hipervolumen que resulte de remover una solución en particular [Knowles and Corne, 2003; Emmerich et al., 2005; Igel et al., 2007; Bader et al., 2010].

## El Hipervolumen

La principal motivación para usar indicadores en el mecanismo de selección de un AEMO es la escalabilidad (en el espacio de las funciones objetivo). Sin embargo, el alto costo computacional del hipervolumen ha motivado el estudio de otros indicadores de desempeño tales como $\Delta_p$.

Oliver Schütze, Xavier Esquivel, Adriana Lara and Carlos A. Coello Coello, **Using the Averaged Hausdorff Distance as a Performance Measure in Evolutionary Multi-Objective Optimization**, *IEEE Transactions on Evolutionary Computation*, Vol. 16, No. 4, pp. 504–522, August 2012.

# Algoritmos Recientes

## Indicador $\Delta_p$

El indicador $\Delta_p$ puede verse como una "distancia promediada de Hausdorff" entre nuestra aproximación y el verdadero frente de Pareto. $\Delta_p$ combina algunas ligeras variantes de dos indicadores de desempeño bien conocidos: la distancia generacional [Van Veldhuizen, 1999] y la distancia generacional invertida [Coello & Cruz, 2005].

$\Delta_p$ es una pseudo-métrica que evalúa simultáneamente proximidad al verdadero frente de Pareto y la distribución de las soluciones a lo largo del mismo. Aunque no es un indicador Pareto compatible, en la práctica, parece trabajar razonablemente bien, y es capaz de lidiar adecuadamente con los *outliers*. Esto lo hace un indicador de desempeño atractivo. Adicionalmente, su costo computacional es muy bajo.

## Indicador $\Delta_p$

Sin embargo, cabe mencionar que para incorporar $\Delta_p$ en el mecanismo de selección de un AEMO, se debe contar con una aproximación al verdadero frente de Pareto a todo momento. Esto ha motivado el desarrollo de técnicas que producen dicha aproximación de manera eficiente y efectiva.

Por ejemplo, Gerst et al. [2011] linearizaron el frente no dominado producido por la población actual y usaron esa información para diseñar el denominado $\Delta_p$-EMOA, el cual se utilizó para resolver problemas bi-objetivo. Este algoritmo está inspirado en el SMS-EMOA y adopta un archivo externo.

K. Gerstl, G. Rudolph, O. Schütze and H. Trautmann, "**Finding Evenly Spaced Fronts for Multiobjective Control via Averaging Hausdorff-Measure**", in *The 2011 8th International Conference on Electrical Engineering, Computer Science and Automatic Control (CCE'2011)*, pp. 975–980, IEEE Press, Mérida, Yucatán, México, October 2011.

## Indicador $\Delta_p$

Existe otra extensión del AEMO del acetato anterior que fue diseñada para resolver problemas con tres funciones objetivo [Trautmann, 2012]. En este caso, el algoritmo requiere algunos pasos previos, que incluyen reducir la dimensionalidad de las soluciones no dominadas y calcular su cubierta convexa (*convex hull*).

Esta versión del $\Delta_p$-EMOA genera soluciones con una mejor distribución pero requiere más parámetros y tiene un costo computacional elevado cuando se utiliza para resolver problemas de optimización con muchos objetivos.

Heike Trautmann, Günter Rudolph, Christian Dominguez-Medina and Oliver Schütze, "**Finding Evenly Spaced Pareto Fronts for Three-Objective Optimization Problems**, in Oliver Schütze et al. (editors), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation II*, pp. 89–105, Springer, Advances in Intelligent Systems and Computing Vol. 175, Berlin, Germany, 2012, ISBN 978-3-642-31519-0.

## Indicador $\Delta_p$

Otra forma posible de incorporar $\Delta_p$ en un AEMO es mediante el uso de las soluciones no dominadas disponibles hasta el momento, de forma escalonada, a fin de construir una aproximación del verdadero frente de Pareto.

Éste es precisamente el enfoque adoptado por el $\Delta_p$-DDE [Rodríguez & Coello, 2012], que utiliza evolución diferencial como su motor de búsqueda. Este AEMO proporciona resultados de calidad similar a los generados por el SMS-EMOA, pero a un costo computacional significativamente inferior (en alta dimensionalidad). Su principal limitante es que las soluciones que produce normalmente no están bien distribuidas en problemas con muchas funciones objetivo. Adicionalmente, este algoritmo tiene dificultades para lidiar con frentes de Pareto desconectados.

Cynthia A. Rodríguez Villalobos and Carlos A. Coello Coello, "**A New Multi-Objective Evolutionary Algorithm Based on a Performance Assessment Indicator**", in *2012 Genetic and Evolutionary Computation Conference (GECCO'2012)*, pp. 505–512, ACM Press, Philadelphia, USA, July 2012, ISBN: 978-1-4503-1177-9.

## *R2*

Algunos investigadores han recomendado el uso del indicador *R2*, el cual fue propuesto originalmente por Hansen [1998] para comparar conjuntos de soluciones utilizando funciones de utilidad [Brockhoff, 2012]. Una función de utilidad es un modelo de las preferencias del tomador de decisiones que mapea cada punto del espacio de las funciones objetivo a un valor de utilidad.

Dimo Brockhoff, Tobias Wagner and Heike Trautmann, "**On the Properties of the *R2* Indicator**, in *2012 Genetic and Evolutionary Computation Conference (GECCO'2012)*, pp. 465–472, ACM Press, Philadelphia, USA, July 2012, ISBN: 978-1-4503-1177-9.

## *R2*

Cabe indicar que *R2* es un indicador débilmente monotónico y que está correlacionado con el hipervolumen, pero tiene un costo computacional muy inferior a éste. Debido a estas propiedades, su uso se recomienda para lidiar con problemas con muchas funciones objetivo. Sin embargo, las funciones de utilidad que se requieren para calcular este indicador deben estar debidamente escaladas.

De acuerdo a Brockhoff [2012], la versión unaria del indicador *R2* para un conjunto de referencia constante se puede expresar de la forma siguiente:

$$R2(A, U) = -\frac{1}{|U|} \sum_{u \in U} \max_{\mathbf{a} \in A} \{u(\mathbf{a})\}, \tag{1}$$

donde $A$ es la aproximación al conjunto de Pareto y $U$ es un conjunto de funciones de utilidad.

Con respecto a la función de utilidad $u : \mathbb{R}^m \rightarrow \mathbb{R}$, existen varias posibilidades: lineal ponderada, Tchebycheff ponderada o Tchebycheff aumentada.

# Algoritmos Recientes

## *R2*

Existen ya varios AEMOs basados en *R*2.

Raquel Hernández Gómez and Carlos A. Coello Coello, **MOMBI: A New Metaheuristic for Many-Objective Optimization Based on the R2 Indicator**, in *2013 IEEE Congress on Evolutionary Computation (CEC'2013)*, pp. 2488–2495, IEEE Press, Cancún, México, 20-23 June, 2013, ISBN 978-1-4799-0454-9.

Dimo Brockhoff, Tobias Wagner and Heike Trautmann, *R*2 **Indicator-Based Multiobjective Search**, *Evolutionary Computation*, Vol. 23, No. 3, pp. 369–395, Fall 2015.

Alan Díaz-Manríquez, Gregorio Toscano-Pulido, Carlos A. Coello Coello and Ricardo Landa-Becerra, **A Ranking Method Based on the R2 Indicator for Many-Objective Optimization**, in *2013 IEEE Congress on Evolutionary Computation (CEC'2013)*, pp. 1523–1530, IEEE Press, Cancún, México, 20-23 June, 2013, ISBN 978-1-4799-0454-9.

# Algoritmos Recientes

## *R2*

Dúng H. Phan and Junichi Suzuki, **R2-IBEA: R2 Indicator Based Evolutionary Algorithm for Multiobjective Optimization**, in *2013 IEEE Congress on Evolutionary Computation (CEC'2013)*, pp. 1836–1845, IEEE Press, Cancún, México, 20-23 June, 2013, ISBN 978-1-4799-0454-9.

Raquel Hernández Gómez and Carlos A. Coello Coello, "**Improved Metaheuristic Based on the *R2* Indicator for Many-Objective Optimization**", in *2015 Genetic and Evolutionary Computation Conference (GECCO 2015)*, pp. 679–686, ACM Press, Madrid, Spain, July 11-15, 2015, ISBN 978-1-4503-3472-3.

## NSGA-III

El *Nondominated Sorting Genetic Algorithm III* (NSGA-III) fue propuesto por Deb y Jain [2014] como una extensión del NSGA-II diseñada específicamente para lidiar con problemas con muchas funciones objetivo (más de tres). El NSGA-III sigue utilizando el ordenamiento no dominado (el cual produce varias capas o niveles de soluciones), pero en este caso el estimador de densidad se basa en una actualización adaptativa de un conjunto de puntos de referencia bien distribuidos.

Kalyanmoy Deb and Himanshu Jain, "**An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints**", *IEEE Transactions on Evolutionary Computation*, Vol. 18, No. 4, pp. 577–601, August 2014.

# Algoritmos Culturales

Algunos sociólogos han sugerido que la cultura puede ser codificada simbólicamente y que puede transmitirse dentre y entre poblaciones, como si se tratara de otro mecanismo de herencia.

Robert Reynolds [1994] desarrolló un modelo computacional en el cual la evolución cultural es vista como un proceso hereditario que opera a dos niveles: el nivel micro-evolutivo y el nivel macro-evolutivo.

# Algoritmos Culturales

Al nivel micro-evolutivo los individuos se describen por medio de “características de comportamiento” (que pueden ser aceptables o inaceptables). Estas características se pasan de generación en generación usando varios operadores inspirados en cuestiones sociales.

Al nivel macro-evolutivo, los individuos son capaces de generar “mappa”, o sea, descripciones generalizadas de sus experiencias.

Los mappas individuales pueden mezclarse y modificarse a fin de formar “mappas” grupales.

Ambos niveles comparten un canal de comunicación.

# Algoritmos Culturales

Reynolds [1994] propuso el uso de algoritmos genéticos para modelar el proceso micro-evolutivo, y los Espacios de Versiones [Mitchell, 1978] para modelar el proceso macro-evolutivo de un algoritmo cultural.

Cada individuo tiene su propio conjunto de creencias, pero éstas se ajustan en e tiempo usando el "mapa de grupo", o sea las experiencias generales de la población. Cada individuo contribuye a tal "mapa de grupo" al final de cada generación.

Robert G. Reynolds, "**An Introduction to Cultural Algorithms**", in A.V. Sebald and L.J. Fogel (editors), *Proceedings of the Third Annual Conference on Evolutionary Programming*, pp. 131–139, World Scientific, River Edge, New Jersey, USA, 1994.

Tom Mitchell, "**Version Spaces: An Approach to Concept Learning**", PhD thesis, Computer Science Department, Stanford University, Stanford, California, USA, 1978.

# Algoritmos Culturales

Cuando un individuo mezcla su mapa individual con el del grupo, hay una cierta combinación de creencias.

Si un individuo tiene un mapa combinado menor que cierto valor aceptable, entonces se le poda del espacio de creencias.

Se usa entonces un proceso de selección para elegir los padres que serán evolucionados en la siguiente generación (de hecho, la selección puede paralelizarse).

El proceso evolutivo se efectúa con ciertos operadores que tienden a ser específicos del dominio.

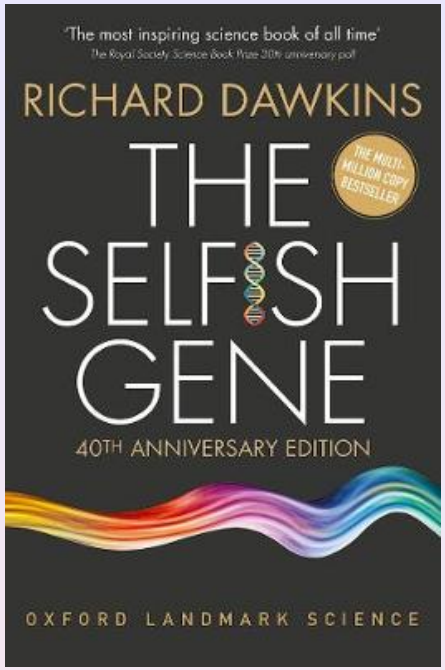
Las interacciones entre el espacio de creencias y el de la población dependen del canal de comunicación que se utilice, así como de sus protocolos.

# Algoritmos Meméticos

Pablo Moscato introdujo el concepto de **algoritmo memético** en 1989, para denotar el uso de buscadores locales combinados con estrategias poblacionales.

Pablo Moscato, "**On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms**", Caltech Concurrent Computation Program, Report 826, Pasadena, California, USA, 1989.

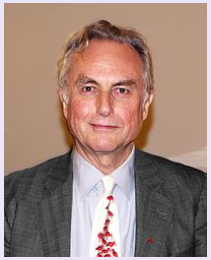
# Algoritmos Meméticos




El término **memético** tiene sus raíces en la palabra "meme", la cual fue introducida por Richard Dawkins en su libro *The Selfish Gene*.

Dawkins define a un meme como una "unidad de imitación" en la transmisión cultural.

# Algoritmos Meméticos

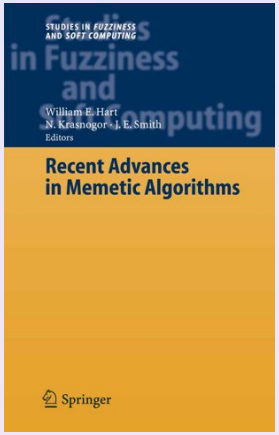


Por tanto, un **algoritmo memético** puede verse como un enfoque que intenta de imitar la evolución cultural en vez de la evolución biológica.

La diferencia principal entre los algoritmos meméticos y los algoritmos evolutivos es en la forma en la que se transmite la información.

Mientras que los genes pasan intactos, los memes típicamente son adaptados por los individuos que los transmiten.

# Algoritmos Meméticos




El acetato siguiente muestra a un algoritmo memético genérico (LS denota la búsqueda local).

Se hace notar que la posición exacta del buscador local dentro del algoritmo evolutivo depende del diseñador del algoritmo.

La búsqueda local efectuada está, evidentemente, acotada, a fin de que el tiempo computacional requerido por el algoritmo no resulte excesivo.

**Algorithm 1** Memetic EA

1: **procedure** MEMETIC EA($\mathcal{N}, g, f_m(\vec{x})$)
2: Randomly initialize population $\mathbb{P}_g$ with $\mathcal{N}$ individuals
3: Evaluate fitness $f_m(\vec{x})$ of each individual $\vec{x}$ in $\mathbb{P}_g$
4: **while** Termination condition false **do**
5: $g = g + 1$; number of generation
6: Select $\mathbb{P}'_g$ from $\mathbb{P}_{(g-1)}$ based on fitness
7: Apply genetic operators to $\mathbb{P}'_g \rightarrow \mathbb{P}''_g$
8: **Local Search** in $\mathbb{P}''_g$ neighborhood; $\mathbb{P}''_g \rightarrow \mathbb{P}'''_g$
9: Evaluate fitness $f_m(\vec{x})$ of each individual in $(\mathbb{P}''_g, \mathbb{P}'''_g)$
10: Select $\mathbb{P}_g$ from $(\mathbb{P}_{g-1}, \mathbb{P}'_g, \mathbb{P}''_g, \mathbb{P}'''_g)$
11: **end while**
12: **end procedure**

# Algoritmos Meméticos

Evidentemente, el balance entre la búsqueda que hace el algoritmo evolutivo y la del buscador local es crítica para lograr buenos resultados, y es tema de investigación en la actualidad.

Asímismo, la definición de buenos buscadores locales para espacios continuos es un tema en el que varios investigadores se encuentran trabajando actualmente, aunque tradicionalmente, los algoritmos meméticos se han usado en espacio de búsqueda discretos.

# Algoritmos Meméticos

Hay varias preguntas que se plantean al diseñar algoritmos meméticos:

1. ¿Qué tan frecuentemente deben usarse la búsqueda local con base en una probabilidad $P_{LS}$?
2. ¿Durante cuánto tiempo $T$ debe ejecutarse el buscador local?
3. ¿Sobre qué $k$ soluciones debe aplicarse la búsqueda local dado un vecindario $N(\vec{x})$, donde $\vec{x}$ es una solución actual?
4. ¿Qué tan eficiente debe ser el buscador local, con respecto a su efectividad?
5. ¿Cómo deben relacionarse los operadores de recombinación y mutación de un algoritmo evolutivo con el buscador local?

6. ¿Debiera usarse un esquema de asignación de aptitud Lamarckiano o uno Baldwiniano?

Cuando se usa **Lamarckismo**, el mejor resultado obtenido con el explorador local sustituye a la solución base (o sea, aquella a partir de la cual se inició la búsqueda local).

Cuando se usa **Baldwinismo**, la aptitud de la mejor solución obtenida por el explorador local es usada por el individuo base, pero el valor de tal solución no reemplaza a dicho individuo base.

El enfoque Lamarckiano es el más común en los algoritmos meméticos.

# Coevolución

En la naturaleza existen organismos que tienen una relación simbiótica con otros organismos.

La **Simbiosis** se define como el "fenómeno mediante el cual dos organismos disimilares viven íntimamente juntos en una relación mutuamente benéfica".

En la comunidad de computación evolutiva se han desarrollado algoritmos que adoptan simbiosis, aunque son relativamente escasos.

# Coevolución

Llamamos **coevolución** a un cambio en la composición genética de una especie (o grupo de especies) como respuesta a un cambio genético de otra. En un sentido más general, la coevolución se refiere a un cambio evolutivo recíproco entre especies que interactúan entre sí.

El término **coevolución** suele atribuírsele a Ehrlich y Raven, quienes publicaron un artículo sobre sus estudios de las mariposas y las plantas a mediados de los 1960s [Ehrlich & Raven, 1964].

Paul R. Ehrlich and Peter H. Raven, "**Butterflies and Plants: A Study on Coevolution**", *Evolution*, Vol. 18, pp. 586–608, 1964.

# Coevolución

Las relaciones entre las poblaciones de dos especies distintas **A** y **B** pueden ser descritas considerando todos los tipos posibles de interacciones.

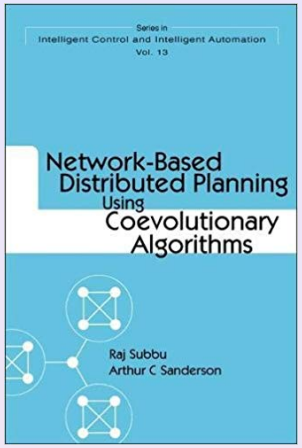
Tales interacciones pueden ser positivas o negativas, dependiendo de las consecuencias que ésta produzca en la población.

La tabla del acetato siguiente muestra todas las posibles interacciones entre dos especies diferentes.

# Coevolución

| | **A** | **B** | |
|---|---|---|---|
| Neutralismo | 0 | 0 | Las poblaciones **A** y **B** son independientes y no interactúan |
| Mutualismo | + | + | Ambas especies se benefician de la relación |
| Comensalismo | + | 0 | Una especie se beneficia de la relación, pero la otra no sufre daño, ni se beneficia |
| Competencia | - | - | Ambas especies tienen un efecto negativo entre ellas puesto que compiten por los mismos recursos |
| Depredación | + | - | El depredador (**A**) se beneficia y la presa (**B**) es afectada negativamente |
| Parasitismo | + | - | El parásito (**A**) se beneficia, mientras que el huésped (**B**) es afectado negativamente |

# Coevolución




La comunidad de computación evolutiva ha desarrollado varios enfoques coevolutivos en los que normalmente dos o más especies se relacionan entre sí usando alguno de los esquemas indicados anteriormente.

Asímismo, en la mayoría de los casos, tales especies evolucionan independientemente a través de un algoritmo evolutivo (normalmente un algoritmo genético).

# Coevolución

Existen dos clases principales de algoritmos coevolutivos en la literatura especializada:

1. Aquellos basados en relaciones de competencia (denominada **coevolución competitiva**): En este caso, la aptitud de un individuo es el resultado de "encuentros" con otros individuos [Paredis, 1997]. Este tipo de esquema coevolutivo se ha adoptado normalmente para los juegos.

Jan Paredis, "**Coevolutionary Algorithms**", in Thomas Bäck, David B. Fogel and Zbigniew Michalewics (Editors), *Handbook of Evolutionary Computation*, Oxford University Press, 1997.

2. Aquellos basados en relaciones de cooperación (denominada **coevolución cooperativa**): En este caso, la aptitud de un individuo es el resultado de una colaboración con individuos de otras especies (o poblaciones) [Potter, & De Jong 2000]. Este tipo de esquema coevolutivo se ha adoptado normalmente para resolver problemas de optimización.

M. Potter and K. De Jong, "**Cooperative Coevolution: An architecture for evolving coadapted subcomponents**", *Evolutionary Computation*, Vol. 8, No. 1, pp. 1–29, 2000.

# Coevolución

Existen varios trabajos en la literatura especializada que usan la coevolución y la simbiosis en un algoritmo evolutivo. Paredis [1995,1998] proporciona una muy buena introducción a la coevolución, discutiendo primero la **aptitud competitiva**.

Este tipo de aptitud se obtiene mediante cálculos que dependen en cierto grado de la población actual.

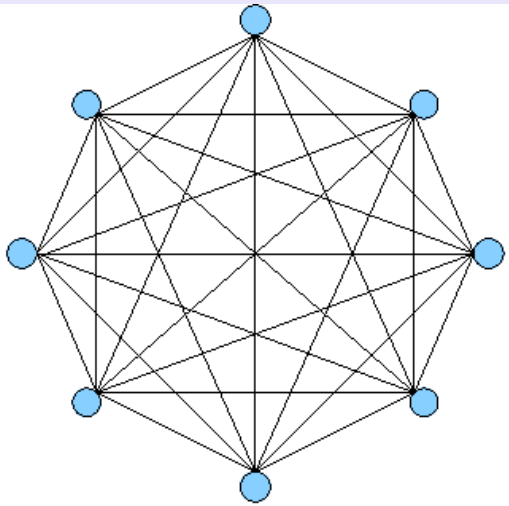
La dependencia puede ser mínima (por ejemplo, se puede depender de un solo miembro de la población) o exhaustiva (o sea, se puede depender de toda la población y combinar sus aptitudes en un solo valor escalar).

# Coevolución

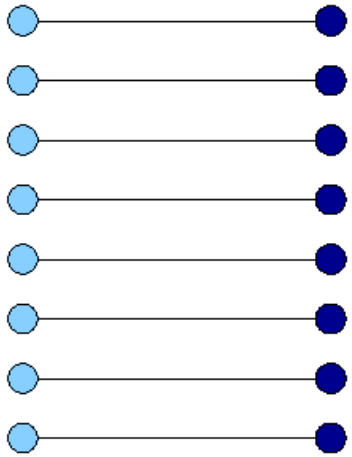

Cuatro ejemplos de funciones de aptitud competitivas incluyen: competencia total (todos vs. todos), competencia bipartita (uno vs. uno o posiblemente uno vs. muchos), aptitud de torneo (torneo binario con eliminación de un solo elemento, la cual no debe confundirse con la selección de torneo), y la competencia elitista (todos vs. el mejor).

Estos esquemas se ilustran gráficamente en el acetato siguiente.

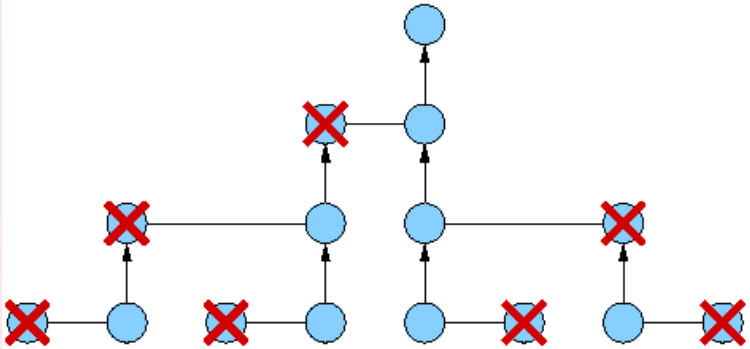
# Coevolución



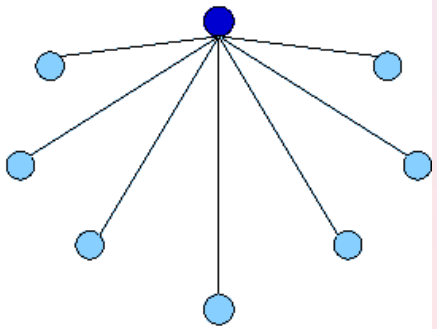
All vs. All



Bipartite



Tournament



All vs. Best

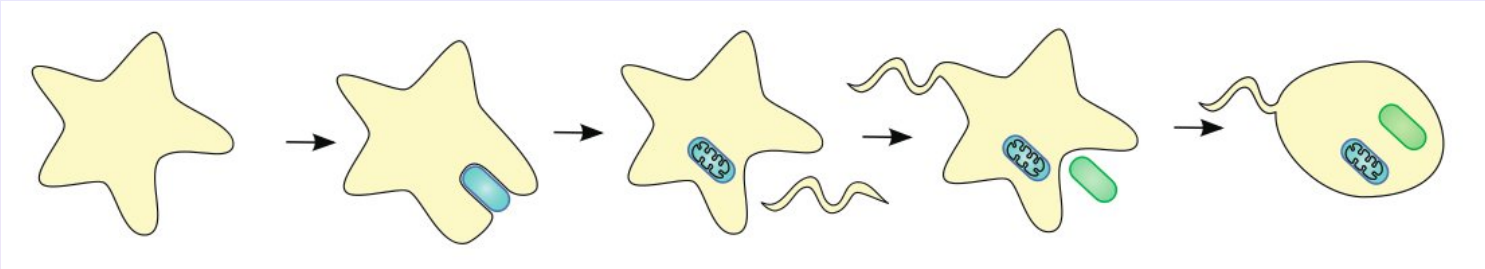
# Coevolución

La aptitud competitiva se ha aplicado extensamente en los juegos evolutivos. Algunos de los juegos en donde se han usado algoritmos evolutivos son: Gato [Angeline, 1993], Otelo [Chong, 2005], Awari [Davis, 2002], Poquer [Billings, 2001], Backgammon [Pollack, 1996] y el dilema del prisionero [Chong, 2005a].

Otra forma de coevolución es el denominado **modelo depredador-presa**. Un ejemplo de este modelo lo encontramos en el trabajo que hizo Danny Hillis con las redes de ordenamiento (*sorting networks*) [Hillis, 1990].

En este caso, se usaron dos poblaciones, una conteniendo las redes de ordenamiento y la otra consistente en un conjunto de listas con 16 números para ser ordenados.

# Coevolución



Wallin et al. [2005] introdujeron un algoritmo coevolutivo cooperativo que se basa en el concepto de **endosimbiosis**.

La **endosimbiosis** se refiere a la relación simbiótica entre un organismo y otro que vive adentro del cuerpo del huésped.

En los algoritmos evolutivos, el uso de coevolución competitiva puede evitar que las poblaciones se estanquen, usando co-especies para mantener la presión evolutiva.

D. Wallin, C. Ryan, and R.M.A. Azad, "**Symbiogenetic coevolution**", in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, pp. 1613–1620, IEEE Press, 2005.

# Coevolución

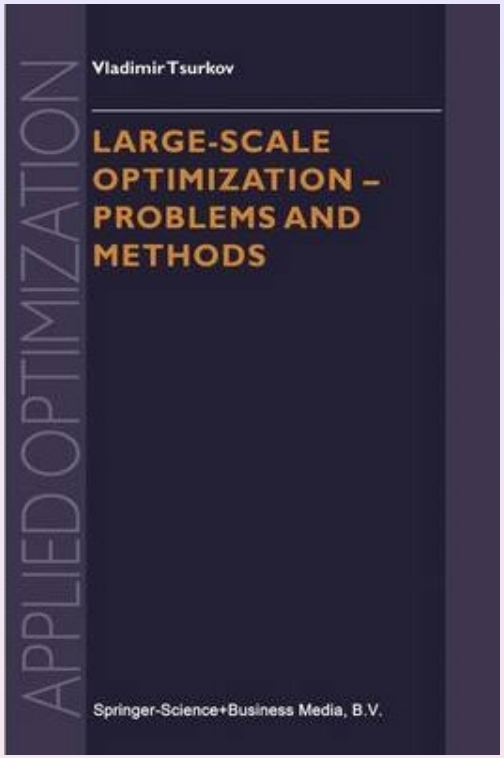
Wallin et al. [2005] propusieron el **Symbiotic Coevolutionary Algorithm** (SCA), el cual usa dos especies: los huéspedes y los parásitos.

Los **huéspedes** son una solución completa, mientras que el **parásito** es una solución parcial.

Para que se pueda evaluar un parásito, debe estar ligado a un huésped.

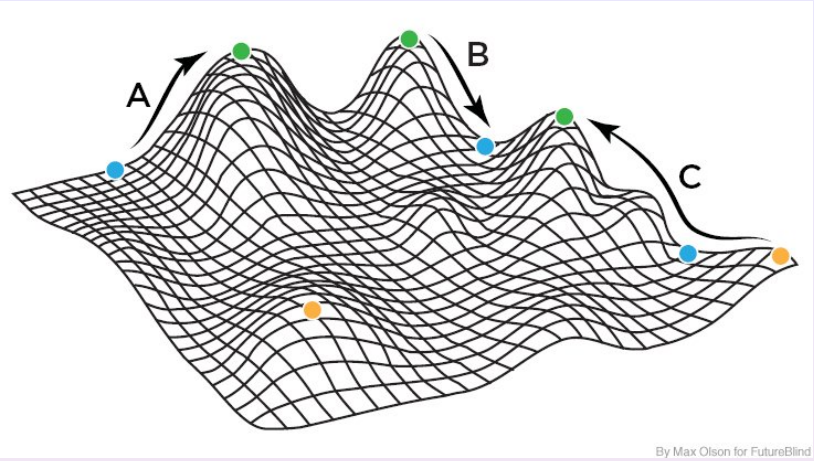
El parásito está conformado por dos cadenas. La primera cadena es un número binario con codificación de Gray, cuyo fin es dirigir los valores parasíticos a una posición dentro del huésped.

# Coevolución




La principal aplicación de los sistemas coevolutivos ha sido la optimización a gran escala (*large scale optimization*), en la cual los sistemas cooperativos han sido muy exitosos.

# Paisajes de Aptitud




El concepto de **paisaje de aptitud** (*fitness landscape*) fue introducido por Sewall Wright en 1932, como una metáfora para describir múltiples dominios de atracción en la dinámica evolutiva.

La idea de Wright fue ver el espacio de búsqueda como si tuviera múltiples picos hacia los cuales la población evolucionaría, escalándolos.

# Paisajes de Aptitud

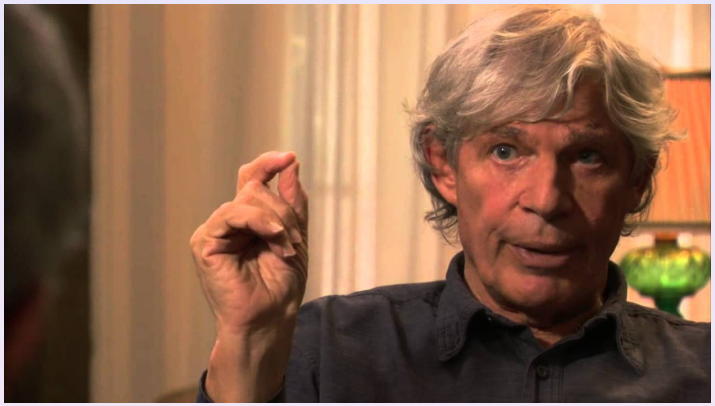
Wright estaba principalmente interesado en analizar la forma en que las poblaciones podían escapar de los óptimos locales a través de fluctuaciones estocásticas.

Por tanto, ésta es una de las primeras propuestas para usar procesos estocásticos para optimizar funciones multimodales.

El concepto de Wright fue concebido como una ayuda para visualizar el comportamiento de los operadores de selección y variación durante el proceso evolutivo.

Por tanto, los paisajes de aptitud normalmente se usan para estudiar la eficacia de los algoritmos evolutivos.

# Paisajes de Aptitud



Particularmente, los investigadores han adoptado los denominados paisajes de aptitud *NK* propuestos por Stuart Kauffman [1987] para explorar la forma en la cual la epístasis controla la rugosidad de un paisaje adaptativo.

La idea de Kauffman fue especificar una familia de funciones de aptitud que tuvieran una rugosidad que pudiera ser ajustable a través de la manipulación de un solo parámetro.

# Sistemas de Clasificadores

Desde los orígenes de los algoritmos genéticos, hubo investigadores interesados en aplicar este tipo de técnicas a problemas de aprendizaje de máquina.

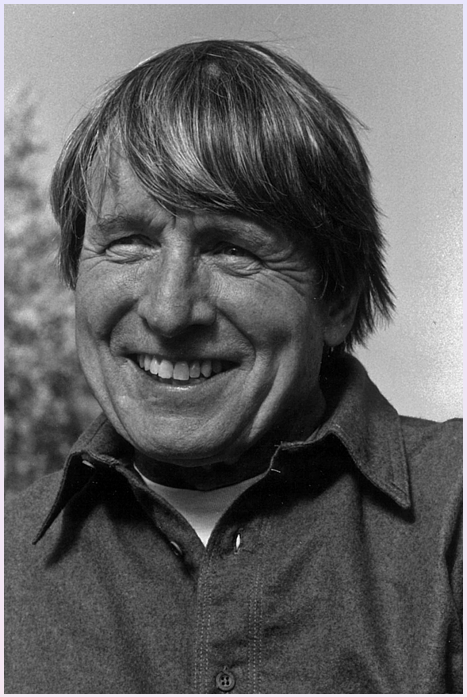
La motivación principal es que los cromosomas, los cuales representan conocimiento, son tratados como datos a ser manipulados mediante operadores genéticos y, al mismo tiempo, como código ejecutable a ser utilizado para realizar alguna tarea.

Esto dio pie al desarrollo de los denominados **sistemas de clasificadores**, en los cuales se usan algoritmos evolutivos para resolver problemas de aprendizaje (fundamentalmente, problemas de clasificación).

# Sistemas de Clasificadores

Desde los viejos días de los algoritmos genéticos, se consideraron dos tipos de enfoques para diseñar sistemas de clasificadores:

1. El primer enfoque consiste en representar todo un conjunto de reglas usando un individuo (o sea, una cadena cromosómica), mantener una población de conjuntos de reglas y usar el mecanismo de selección y los operadores de un algoritmo genético para producir nuevos conjuntos de reglas. A este tipo de técnica se le denomina **enfoque Pitt**, dado que fue desarrollado por Kenneth De Jong y sus estudiantes, cuando éste se encontraba en la Universidad de Pittsburgh.



2. John Holland desarrolló, al mismo tiempo que De Jong, otro modelo en el cual los miembros de la población son reglas individuales y un conjunto de reglas es realmente toda la población. A éste se le denomina el **enfoque Michigan**, pues fue desarrollado en la Universidad de Michigan.

# Sistemas de Clasificadores: El Enfoque Michigan

Los sistemas de clasificadores son un tipo de sistema basado en reglas con mecanismos generales para procesar reglas en paralelo, a fin de generar, de manera adaptativa, nuevas reglas, así como para probar la efectividad de las reglas existentes.

Los sistemas de clasificadores proporcionan un marco de trabajo en el cual una población de reglas codificadas como cadenas de bits evolucionan con base en recibir, intermitentemente, estímulos y refuerzo de su ambiente.

El sistema "aprende" qué respuestas son apropiadas cuando se presenta un cierto estímulo.

Las reglas en un sistema de clasificadores forman una población de individuos que evolucionan en el tiempo.

# El Enfoque Michigan

Un sistema de clasificadores consta de los siguientes componentes:

- detector y efector,
- sistema de mensajes (entrada, salida, y listas de mensajes internos),
- sistema de reglas (población de clasificadores),
- sistema de asignación de crédito (el algoritmo *bucket brigade*), y
- procedimiento genético (reproducción de los clasificadores).

# El Enfoque Michigan

El ambiente envía un mensaje (p.ej., un movimiento en un tablero), el cual es aceptado por los detectores del sistema de clasificadores y colocado en la lista de mensajes de entrada.

Los detectores decodifican el mensaje (posiblemente sub-dividiéndolo en varios mensajes) y lo colocan en la lista de mensajes internos. Los mensajes activan clasificadores; los clasificadores activados colocan mensajes en la lista de mensajes.

Estos nuevos mensajes pueden activar a otros clasificadores o pueden irse a la lista de mensajes de salida. En este último caso, los efectores del sistema de clasificadores codifican estos mensajes en un mensaje de salida, el cual se regresa al ambiente. El ambiente evalúa la acción del sistema (retroalimentación del ambiente) y se usa el algoritmo del *bucket brigade* para actualizar los clasificadores.

# El Enfoque Michigan

El enfoque Michigan puede percibirse como un modelo computacional de la cognición: el conocimiento de una entidad cognitiva se expresa como una colección de reglas que sufren modificaciones en el tiempo.

Podemos evaluar una unidad cognitiva completa en términos de su interacción con el ambiente; la evaluación de una sola regla (o sea, de un solo individuo) no tiene ningún sentido.

# El Enfoque Pitt

En el enfoque Pitt, cada individuo en la población codifica todo el conjunto de reglas (es decir, el individuo es una entidad cognitiva).

Estos individuos compiten entre sí, de manera que los individuos débiles se mueren y los fuertes sobreviven y se reproducen (usando el mecanismo de selección y los operadores convencionales de un algoritmo genético). En otras palabras, se aplica un algoritmo genético al problema de aprendizaje.

Con esto, se evita el problema de la asignación de crédito, para el cual se requiere un método heurístico (como el algoritmo del *bucket brigade*) para distribuir el crédito (positivo o negativo) entre las reglas que cooperaron para producir un comportamiento deseable o indeseable.

# El Enfoque Pitt

El enfoque Pitt, sin embargo, plantea varias preguntas interesantes. Por ejemplo: ¿debemos usar cadenas de longitud fija para representar las reglas? Esto facilita el funcionamiento del algoritmo genético, pero puede ser muy restrictivo.

De hecho, pueden usarse representaciones más complejas para los sistemas de reglas (p.ej., incorporando lógica difusa, entradas que sean números reales, operadores booleanos, etc.).

# El Futuro de la Computación Evolutiva

- **Metamerismo**: El proceso en el cual una unidad estructural es duplicada un cierto número de veces y durante ese proceso se reoptimiza para otros usos.
- **Auto-adaptación**: Evitar el uso de parámetros *ad-hoc* en los algoritmos evolutivos.
- **Técnicas que exploten arquitecturas paralelas**: Es importante explotar al máximo las arquitecturas paralelas mediante nuevos algoritmos evolutivos. Esto traerá importantes ganancias en términos de esfuerzo computacional, sobre todo al lidiar con problemas del mundo real.

# El Futuro de la Computación Evolutiva

- **Teoría**: Pruebas de convergencia, modelos matemáticos de los algoritmos evolutivos, epístasis, diversidad, etc.
- **Entender mejor la evolución natural**: Simulaciones computacionales que permitan entender las complejas interacciones que ocurren entre los seres vivos.
- **Coevolución**: Muchos investigadores han dirigido sus esfuerzos a estudiar mejor la coevolución como un modelo alternativo para resolver problemas en computación evolutiva.
- **El algoritmo genético sin parámetros**: Es, sin lugar a dudas, el sueño de los expertos en computación evolutiva.

# El Futuro de la Computación Evolutiva


- Las 3 grandes preguntas:
  1) ¿A qué tipo de problemas deben aplicarse los algoritmos evolutivos?
  2) ¿Cómo podemos mejorar nuestra comprensión del funcionamiento de los algoritmos evolutivos?
  3) ¿Qué nuevas ideas pueden aplicarse a la computación evolutiva a fin de extender el paradigma (p.ej., inspiración biológica)?