Búsqueda de configuraciones de parámetros óptimas mediante Harmony Search.

Ernesto Luis Estevanell Valladares e.estevanell@estudiantes.matcom.uh.cu
Grupo C412

Rodrigo Sua García Eternod r.garcia@estudiantes.matcom.uh.cu
Grupo C412

Leonardo Fleitas Alvarez l.fleitas@estudiantes.matcom.uh.cu
Grupo C412

Marcos Toranzo Alfonso m.toranzo@estudiantes.matcom.uh.cu
Grupo C412

Facultad de Matemática y Computación, 4to. Año,
Universidad de La Habana

Tutor: MSc. Fernando Rodríguez Flores

Resumen

Los parámetros tienen un papel fundamental en cada algoritmo y afectan directamente a los resultados que se obtienen. Su optimización es un problema comúnmente atacado por la ley de la prueba y error o por un análisis particular del problema. La automatización de las configuraciones de parámetros sigue siendo una tarea atacada de muchas maneras. En este trabajo se presentará un enfoque metaheurístico para dicha tarea mediante el algoritmo Harmony Search.

Palabras Clave: Metaheurística, Optimización, Configuración.
Tema: Optimización de parámetros, Metaheurística, Modelos de optimización.

1 Introducción

El diseño e implementación de algoritmos es una tarea ardua, mas no se debería subestimar el esfuerzo que viene luego de las pruebas: el ajuste de parámetros.

La estabilidad y calidad de los resultados de los algoritmos es dictada usualmente por los valores de los parámetros que conforman su configuración por lo que la tarea de encontrar un ajuste óptimo de la misma merece atención. Sin embargo, predominan métodos extenuantes basados en la experimentación, prueba y error para la 'apropiada' selección configuración de parámetros.

Desde hace años vienen surgiendo nuevas propuestas para lograr la automatización de la selección de valores de los parámetros, pero mayormente estos no ofrecen un trato diferenciado en dependencia de la naturaleza (continua o discreta) de los mismos.

La problemática antes mencionada sirvió de motivación para la implementación del algoritmo Harmony Search, una apuesta por un acercamiento Metaheurístico, propuesto en [1].

1.1 Objetivos:

Ofrecer una solución metaheurística para la selección óptima de parámetros en algoritmos de optimización, así como crear una interfaz de usuario capaz de ofrecer el uso de esta propuesta como servicio, pudiendo aceptar **algoritmos objetos** programados en cualquier lenguaje de programación.

1.2 Estructura de documento:

En la sección 1 se ofrecerá una introducción al problema tratado, así como el objetivo establecido para este trabajo. En la sección 2 se establecerán las pautas y consideraciones generales a tener en cuenta, así como los preliminares necesarios, para lograr una comprensión correcta del contenido plasmado. En la sección 3 se da una definición formal del problema, así como la estrategia de solución adoptada y la interfaz de usuario implementada. En la sección 4 se discutirán los resultados obtenidos.

2 Consideraciones Generales y Preliminares

Esta sección expondrá consideraciones tomadas en relación al problema en cuestión, así como ciertas definiciones de gran importancia.

2.1 Metaheurísticas

Son una familia de algoritmos aproximados de propósito general. Suelen ser procedimientos iterativos que guían una heurística subordinada de búsqueda, combinando de forma inteligente distintos conceptos para explorar y explotar adecuadamente el espacio de búsqueda.

La heurística de búsqueda, el espacio y los métodos de exploración y explotación se definen en dependencia del problema a tratar y conforman el algoritmo a utilizar. Más adelante en este informe se definirá de manera formal el problema y el algoritmo utilizado.

2.2 Problema, algoritmo y parámetros

Sea A un algoritmo para la solución del problema:

$$P: \min_{x \in \Omega} f(x)$$

A se denomina algoritmo objeto y normalmente tiene asociado un conjunto de parámetros que tienen participación directa a la hora de obtener resultados de una instancia especifica de P. Por Ejemplo, la mayoría de algoritmos genéticos implementan operaciones de *"Mutation"* y *"Crossover"* y cuentan con parámetros como *mutation rate*, *crossover rate*.

Los valores de parámetros están incluidos en algún dominio y suelen ser:

- Nominales: [bueno, regular, malo]
- Numéricos:
  - Discretos: [1,2,3,8]
  - Continuos: (1,5] valores reales

Los parámetros nominales pueden ser traducidos a numéricos discretos para facilitar su tratado, pero la próxima propuesta no trata con dicha conversión.

Se conoce como selección de parámetros, configuración de parámetros u optimización de configuraciones, al problema de encontrar un vector de valores válidos de parámetros para A de forma tal que al aplicar el algoritmo a las instancias de P con la configuración seleccionada se obtengan los mejores valores posibles. Tomando en cuenta que:

- Θ es el conjunto de todas las configuraciones de parámetros válida para A
- Π es el conjunto de todas las instancias de P
- $c(\theta, \pi)$ es el costo de la solución obtenida al aplicar $A(\theta)$, $\theta \in \Theta$, a $\pi \in \Pi$
- D es una distribución sobre Π que indica la probabilidad de que la instancia $\pi \in \Pi$sea seleccionada para ser resuelta.
- $o_D(\theta, \Pi)$ es la estimación o costo observado de las soluciones obtenidas al aplicar $A(\theta), \theta \in \Theta$ en la solución de cada elemento de Π, según D y $c(\theta, \pi)$, su valor debe estar en correspondencia

con la calidad de las soluciones obtenidas al utilizar la configuración θ y es una medida de la efectividad o desempeño del algoritmo objeto con esa configuración,

entonces el problema de configuración de parámetros se traduce a hallar θ^*tal que:

$$\theta^* \in \arg\min_{\theta \in \Theta} o_D(\theta, \Pi)$$

Los valores de $c(\theta, \pi)$ y $o_D(\theta, \Pi)$ dependen en gran medida de la naturaleza de A. Si A es un algoritmo exacto o determinista entonces $c(\theta, \pi)$ es el propio valor de la solución encontrada. En cambio, si A es heurístico o no determinista, el valor de $c(\theta, \pi)$ puede ser estimado a partir del costo de las soluciones obtenidas en todas las evaluaciones.

3 Presentación del problema y Estrategia de Solución

En esta sección se presentan formalmente el problema y la estrategia de solución empleada, así como una introducción a la interfaz de usuario propuesta.

3.1 Presentación del Problema

Retomando el problema planteado anteriormente, se había afirmado que A puede ser determinista o no. El espacio de configuraciones validas se conforma con el producto cartesiano de los conjuntos de valores factibles para cada uno de los parámetros del algoritmo, es decir:

Definición 3.1: Si el algoritmo A para la solución del problema P tiene m parámetros, entonces el conjunto o espacio de configuraciones factibles se define como:

$$\Theta = \Theta_1 \times \ldots \times \Theta_m$$

Donde Θ_i es el conjunto de valores factibles para el parámetro $i (i \in [1 \ldots m])$. Cada $\Theta_i$es un intervalo $[l_i, u_i]$ real o entero si el parámetro numérico correspondiente es de dominio continuo o discreto respectivamente, l_i es la cota inferior y u_i es la cota superior para los valores del parámetro i.

En dependencia de la clase de problemas enmarcados en la definición de P pueden existir infinitas instancias de los mismos, por lo que en la práctica se necesita estimar el valor de $o_D(\theta, \Pi)$ considerando un conjunto finito de ellas.

Como se expresa en [1], el cálculo del desempeño de una configuración es extremadamente difícil. En el caso de A ser un algoritmo exacto, el costo de la solución encontrada para una instancia $\pi \in \Pi$ $c(\theta, \pi)$ es el mismo cada vez que se ejecute, mientras que en caso de A sea no determinista se requeriría evaluar más de una vez cada instancia con una misma configuración.

Entonces, siguiendo las pautas marcadas se decidió utilizar el criterio de dominancia establecido por el actual MSc. Oscar Luis en [1].

Definición 3.2: Sean $\theta_1, \theta_2 \in \Theta$ configuraciones válidas para el algoritmo A y $\pi \in \Pi$ una instancia del problema que resuelve. Se dice que θ_1 domina a θ_2 sobre π si:

- $c(\theta_1, \pi) < c(\theta_2, \pi)$, si A es un algoritmo determinista.
- $P\big(c(\theta_1, \pi) = opt(\pi)\big) > P(c(\theta_2, \pi) = opt(\pi))$, si A es un algoritmo no determinista.

Donde $P\big(c(\theta, \pi) = opt(\pi)\big)$ para $\theta \in \Theta$ y $\pi \in \Pi$ es la probabilidad de que A encuentre el óptimo conocido para π utilizando la configuración θ. La estimación de esta probabilidad es un problema a tratar de por sí.

Por lo general, seleccionar la mejor configuración para un algoritmo debe involucrar más de una instancia del problema y normalmente no es factible considerar todas las instancias (pueden ser infinitas). Como se ha debe manejar un conjunto finito de ellas es necesario extender el concepto de dominancia anterior.

Definición 3.3: Sean $\theta_1, \theta_2 \in \Theta$ configuraciones válidas para el algoritmo A y $\Pi_A \in \Pi$ un conjunto finito de instancias del problema que resuelve. Se dice que θ_1 domina a θ_2 sobre Π_A si:

$$\sum_{\pi \in \Pi_A} \delta(\theta_1, \theta_2, \pi) > 0$$

Donde:

$$\delta(\theta_1, \theta_2, \pi) = \begin{cases} 1 & \theta_1 \text{ domina a } \theta_2 \; sobre \; \pi \\ -1 & \theta_2 \text{ domina a } \theta_1 \; sobre \; \pi \\ 0 & en \; otro \; caso \end{cases}$$

Si el resultado de la suma es 0 entonces θ_1 y θ_2 no son dominadas sobre Π_A y de ser negativo entonces θ_2 domina a θ_1 sobre ese conjunto.

Luego, en presencia de un conjunto de entrenamiento o análisis, el problema se traduce en la búsqueda de configuraciones válidas no dominadas.

3.2 Harmony Search

El creador de esta relativamente joven metaheuristica se inspiró en la forma en la que los jazzistas improvisan nuevas melodías con una metodología sencilla y ordenada.

Como es habitual, esta metaheuristica se apropia de la jerga del medio que sirvió de inspiración en su creación. Mientras que en algoritmos genéticos se conocen términos como población, individuo, cromosomas, mutación, cruzamiento, en *Harmony Search* se habla de melodías para referirse a las soluciones y de improvisaciones a las nuevas soluciones.

Harmony Search fue creada para resolver problemas de optimización donde cada una de las variables pertenece a un intervalo cerrado real o a un conjunto discreto y finito de valores, esto es:

$$\min f(x)$$
s.a:
$l_i \leq x_i \leq u_i \quad x_i$ continua
$x_i \in \{x_i(1) \ldots x_i(K_i)\} \; x_i$ discreta

Para facilitación de la implementación se considera el trabajo con las variables discretas como intervalos al igual que las continuas, puesto a que todo valor del conjunto de valores posible puede "*mappearse"* a un intervalo discreto de valores.

El primer paso, como es habitual en las metaheurísticas, es inicializar una población HM (***H**armony **M**emory*) de soluciones de tamaño HMS dado como parámetro. El valor de cada una de las componentes de estas soluciones es seleccionado con probabilidad uniforme en el conjunto de valores factibles correspondiente. En cada iteración el algoritmo improvisa una nueva melodía, tomando en cuenta los parámetros $HMCR$ (***H**armony **M**emory **C**onsidering **R**ate*) y PAR (***P**itch **A**djustment **R**ate*). Luego se compara la solución con la peor en memoria, se desecha la de peor calificación. El proceso se repite, y termina cuando se satisface algún criterio de parada que por lo general es una cantidad máxima de iteraciones.

3.2.1 Improvisación

El proceso de improvisación construye una solución componente a componente: para cada una con probabilidad $HMCR$ selecciona uno de los valores almacenados en memoria para esa misma componente y con probabilidad $1 - HMCR$ lo selecciona aleatoriamente del conjunto de valores factibles. Si el valor seleccionado de la memoria, con probabilidad PAR es perturbado, en caso contrario no se modifica. Si se perturba el valor de la componente se tiene en cuenta si es continua o discreta y se elige una de estas dos opciones siguientes:

$$x_i = x_i + \left(\max_{j \in [1 \ldots HMS]} \left(x_i^j \right) - x_i \right) * rand(0,1)$$
$$x_i = x_i - \left(x_i - \min_{j \in [1 \ldots HMS]} \left(x_i^j \right) \right) * rand(0,1)$$

3.3 Estrategia de solución

La propuesta de solución del problema planteado anteriormente asumirá que se cuenta para una situación concreta con un algoritmo objeto. Para cada uno de los parámetros del algoritmo se tendrá el rango de valores admisibles y si es discreto o continuo, lo que es equivalente al conjunto de soluciones factibles. Como se propone una interfaz de uso para un usuario se asume que el algoritmo tiene control sobre un conjunto de entrenamiento el cual se asume finito y

representativo de todas las posibles instancias del problema.

Harmony Search compara dos soluciones, que en este caso serán dos configuraciones del algoritmo objeto, a partir del valor de la función evaluada en ellas, sin embargo, solo conocemos la relación de dominancia entre dos configuraciones, con lo cual es necesario adaptar la metaheuristica a este concepto en el desarrollo de la estrategia de solución.

En presencia de algoritmos no deterministas, estimar el valor de $P\big(c(\theta,\pi) = opt(\pi)\big)$ requiere más de una ejecución del algoritmo con la misma configuración en la misma instancia. Asumimos que el algoritmo ya realiza este cálculo en su implementación, por cuenta del usuario, ya que previamente se contempló que el algoritmo en sí tendría el control sobre su conjunto de entrenamiento, es decir: Se esperaran tantos valores como tamaño del conjunto de entrenamiento (expresando cada uno la probabilidad planteada en caso de ser no determinista).

Sobre dichos resultados del conjunto de entrenamiento se calculará la dominancia según la definición **3.3**. En ausencia del valor óptimo de la instancia se empleará el mejor valor que se obtenga tras la evaluación de las configuraciones.

3.4 Interfaz de Usuario

Ya se ha presentado un algoritmo metaheurístico para la automatización de las configuraciones de parámetros, pero no es ahí donde reside en totalidad el objetivo de este trabajo, comparte con otro requerido: La comunicación con el usuario y la aceptación de algoritmos sin discriminación de lenguajes.

El protocolo de comunicación es sencillo, se realizará mediante una consola y actuará como un *"shell"*, donde se le listará al usuario un conjunto de funcionalidades y se le guiará en la recolección de datos necesarios para la utilización del algoritmo propuesto.

La no discriminación de lenguajes depende de las condiciones del sistema operativo en el que se trabaje en ese momento ya que se le pedirá una instrucción valida en la app de consola usada que permita la ejecución de un fichero que contenga una implementación del algoritmo propuesto.

La implementación del algoritmo deberá posibilitar la introducción de parámetros mediante la entrada estándar al inicio de ejecución utilizando los parámetros del sistema. Un ejemplo de instrucción que deberá tener éxito en la ejecución de un algoritmo que recibe dos parámetros:

```
>>> python algorithm.py param1 param2
```

Se esperará como valor de retorno de la ejecución del fichero y, mediante la salida estándar, un conjunto de valores numéricos que representaran el costo (si A es determinista) o la probabilidad de que el valor sea el óptimo de la instancia (si A es no determinista) de tamaño igual a la dimensión del conjunto de entrenamiento establecido. Sobre dicho listado de respuesta se llevará a cabo la comparación siguiendo las definiciones de dominancia previas.

La respuesta que se le presentará al usuario al final de una ejecución exitosa de la solución propuesta estará compuesta por los valores de la mejor configuración encontrada, el valor de evaluación de dicha configuración y el tiempo que tomó el cómputo de la misma facilitando futuros análisis estadísticos o de eficiencia.

4 Resultados experimentales

Las pruebas se realizarán con un algoritmo determinista y otro no determinista. Los algoritmos escogidos son:

- GIG method incremental versión presentado en [2] (no determinista).
- Gradient Descent (determinista).

4.1 Protocolo

Se realizarán 10 corridas del algoritmo Harmony Search por cada algoritmo de prueba tomando los siguientes valores de parámetros:

- $HCMR$ con valor 0.99.
- PAR con valor inicial 1 y final 0.
- Tamaño de memoria 50.

Los valores deberían variar en dependencia del escenario en el que será utilizado el algoritmo, como pasa para las mayorías de heurísticas.

4.2 Método GIG

El algoritmo *GIG* (**G**rammar **I**nference by **G**enetic search) propuesto presenta una configuración de 4 parámetros a optimizar:

- ritmo de mutación, valores reales entre 0 y 1 (**m.r**).
- ritmo de cruzamiento, valores reales entre 0 y 1 (**c.r**).
- cantidad de generaciones, valores enteros (**Ga**).
- tamaño de población, valores enteros (**Ps**).

Al ser este algoritmo no determinista, se aseguró en su implementación más de 1 corrida por instancia de problema en el conjunto de entrenamiento. Se utilizó un conjunto de entrenamiento de tamaño 5.

La **tabla 1** muestra los valores de la configuración alcanzados por el algoritmo Harmony Search en cada corrida del mismo, exponiendo los resultados por cada instancia del problema en el conjunto de entrenamiento.

4.3 Gradient Descent

El algoritmo Gradient Descent propuesto presenta una configuración de 3 parámetros a optimizar:

- Ritmo de aprendizaje, valores reales entre 0 y 1 (**eta**).
- Cantidad de iteraciones, valores enteros (**Ci**).
- Tolerancia a la condición de parada (épsilon), valores reales entre 0 y 1 (**e**).

Se utilizó un conjunto de entrenamiento de tamaño 3.

La **tabla 2** muestra los valores de la configuración alcanzados por el algoritmo Harmony Search en cada corrida del mismo, exponiendo los resultados por cada instancia del problema en el conjunto de entrenamiento.

4.4 Concluciones

El método Harmony Search ha llegado a demostrar buenos resultados en la práctica, logrando establecerse un paso más en la automatización de las configuraciones de parámetros. Sin embargo, prueba ser una aproximación extremadamente lenta, sobre todo trabajando con algoritmos heurísticos como algoritmos objetos. Se ha logrado implementar una interfaz sencilla de usuario capaz de aplicar la estrategia de solución planteada a cualquier algoritmo de optimización sin discriminación del lenguaje en el que se encuentre programado.

5 Recomendaciones

Al ser el Harmony Search un algoritmo metaheurístico de por si, se explica en parte su tiempo práctico de ejecución a la hora de trabajar con ciertos algoritmos no deterministas por lo que se propone encontrar otros métodos (preferiblemente no heurísticos) para atacar el problema en cuestión.

Referencias

[1] Oscar Luis Vera Pérez. Una Propuesta para la Selección Automática de Configuraciones. Universidad de la Habana, 2011.

[2] Pierre Dupont. Regular Inference from Positive and Negative Samples by Genetic Search.

ANEXOS

Runs	eta	Ci	e	Results
1	0.0377	8	0.0580	0.1225934810943101, 1.1210439280401991, 0.41812441562184327
2	0.0522	9	0.2044	6.466834293801865, 0.6680884420788206, 0.47589662505802777
3	0.06	18	0.06	1.2369769630893055, 0.12920164756391397, 0.26499871765449273
4	0.08	6	0.0678	788.0002854951166, 0.2552576667589772, 0.6389043669670719
5	0.08	14	0.08	1425609.4819852435, 0.05839701261985085, 0.5254710124194046
6	0.02	18	0.02	0.3928495234060555, 0.11988064768107647, 0.5817982185177071
7	0.0620	15	0.4909	1.401699984580708, 1.3549394327981206, 0.3712609743318876
8	0.1008	10	0.0328	238630.4178441716, 0.021339393412977695, 0.31214498053014433
9	0.06	9	0.06	0.20142354717180652, 0.3301011445523816, 0.41142692386985663
10	0.04	4	0.04	0.4523209135834048, 1.524824490589463, 0.265875989558271

Tabla 2. Resultados experimentales con el método Gradient Descent en 10 corridas del Harmony Search

Runs	m.r	c.r	Ga	Ps	Results
1	0.87	0.8822	5	10	100.0, 93.0, 100.0, 91.0, 100.0
2	0.3736	0.3838	20	6	100.0, 100.0, 100.0, 92.0, 100.0
3	0.7754	0.5133	11	19	100.0, 100.0, 100.0, 94.5, 100.0
4	0.7398	0.7398	14	4	100.0, 100.0, 100.0, 97.0, 100.0
5	0.6152	0.6152	13	10	100.0, 100.0, 100.0, 95.0, 100.0
6	0.7576	0.7576	16	8	100.0, 100.0, 100.0, 92.0, 100.0
7	0.8466	0.8466	12	11	100.0, 94.5, 100.0, 89.0, 100.0
8	0.4728	0.4726	12	12	100.0, 100.0, 100.0, 91.0, 100.0
9	0.3481	0.3481	20	9	100.0, 98.5, 100.0, 95.5, 100.0
10	0.1322	0.3837	11	4	100.0, 97.5, 100.0, 93.0, 100.0

Tabla 1. Resultados experimentales con el método GIG en 10 corridas del Harmony Search.