ISSN 1870-4069

Gramática evolutiva con estrategia evolutiva como núcleo de una hiperheurística de generación aplicada al problema de empacado

Carlos Ávila[1], Marco Sotelo-Figueroa[2], Héctor Puga[1], Manuel Ornelas[1], Martín Carpio[1]

[1] Universidad de Guanajuato, División de Ciencias Económico Administrativas, Departamento de Estudios Organizacionales, Guanajuato, GTO, México

[2] Tecnologico Nacional de México, Instituto Tecnológico de León, Departamento de Estudios de Posgrado e Investigación, León, GTO, México

uribeabc@gmail.com, masotelo@ugto.mx, pugahector@yahoo.com, mornelas67@yahoo.com.mx, jmcarpio61@hotmail.com

Resumen. Las hiperheurísticas son herramientas que permiten la generación de metodologías para la solución de determinados problema. Existen dos tipos de hiperheurísticas, las de Generación y las de Selección; si bien el objetivo de ambas es generar metodologías los insumos de ambas cambian, las de Selección requieren de una serie de heurísticas de bajo nivel mientras que las de Generación requieren una serie de elementos heurísticos los cuales describen el problema. Dichas hiperheurísticas están compuestas por un núcleo el cual es un algoritmo de optimización, el cual por su naturaleza tiene una serie de parámetros. La optimización de dichos parámetros es una rama de estudio que permite buscar cuales son aquellos bajo los cuales el algoritmo tiene el mejor desempeño en una instancia de un problema. En el presente trabajo se propone el uso de una Gramática Evolutiva (GE) con un motor de búsqueda basado en una Estrategia Evolutiva (ES) como núcleo de una hiperheurística de generación debido a que cuentan con un proceso de auto adaptación que permite eliminar el proceso de optimización de parámetros. La propuesta es aplicada al Problema de Empacado y los resultados son comparados usando la prueba estadística no paramétrica de Friedman contra los obtenidos por una hiperheurística con motor de búsqueda basado en Gramática Evolutiva usando Optimización por Cumulo de Partículas (PSO) y Optimización Evolutiva por Cúmulo de Partículas (PESO).

Palabras clave: Hiperheurística, programación genética, gramáticas evolutivas, problema de empacado, estrategia evolutiva.

Carlos Ávila, Marco Sotelo-Figueroa, Héctor Puga, Manuel Ornelas, Martín Carpio

Evolutionary Grammar with an Evolution Strategy as a Generation Hiperheuristic Core Applied to the Bin Packing Problem

Abstract. Hyperheuristics are tools that allow the generation of methodologies for solving certain problems. There are two types of hyperheuristics, Generation and Selection; although the objective of both is to generate methodologies, their inputs change, the selection ones require a series of low level heuristics while those of Generation require a series of heuristic elements which describe the problem. These hyperheuristics are composed by a core which is usually an optimization algorithm, which by nature has a series of parameters. The optimization of these parameters is a branch of study that aims to find those values under which the algorithm has the best performance in an instance of a problem. In the present work we propose the use of an Evolutionary Grammar (GE) with a search engine based on an Evolution Strategy (ES) as the core of a generation hyperheuristic because of it's self-adaptive nature that allows to eliminate the process of parameter optimization. The proposal is applied to the Bin Packing Problem and the results are compared using Friedman's non-parametric statistical test against those obtained by a search engine hyperheuristic based on Evolutionary Grammar using Particle Swarm Optimization (PSO) and Particle Evolutionary Swarm Optimization (PESO).

Keywords: Hiperheuristic, genetic programming, evolutionary grammar, bin packing problem, evolution strategies.

1. Introducción

Las Hiperheurísticas (HH) [5,4] son herramientas que permiten la selección o generación de metodologías de búsqueda de soluciones de un determinado problema. Existen muchos trabajos que se han dedicado al estudio de las características y aplicaciones de las hiperheurísticas [11,30,13,29,17]. Las hiperheurísticas de generación son utilizadas cuando es necesario encontrar una metodología que permita obtener un resultado de un problema[5,4,11]; para esto es necesario tener caracterizado el problema de manera que se puedan obtener los elementos representativos del mismo, conocidos como componentes heurísticos. Dentro de los componentes que conforman las hiperheurísticas de generación esta el núcleo que requiere del uso de alguna metodología de construcción que permita hacer combinaciones de componentes heurísticos de bajo nivel.

Las Gramáticas Evolutivas (GE) han sido utilizadas en varias ocasiones como núcleo de las hiperheurísticas de generación obteniendo buenos resultados [3,**?**,24,26,23,2,27,25]. Una GE necesita de un algoritmo o motor de búsqueda que le permita evolucionar los individuos y así poder encontrar soluciones [16,15], algunos de los algoritmos usados como motor de búsqueda de GE son

Algoritmo Genético (GA) [3,18,23,2], Optimización por Cúmulo de Partículas (PSO) [26,27], Optimización Evolutiva por Cumulo de Partículas (PESO) [26], Optimización por Colonia de Abejas (BSO) [27] y Evolución Diferencial (DE) [24,25] entre otros. Uno de los mayores retos a la hora de implementar Algoritmos Evolutivos (AE) como los antes mencionados es el de la optimización de parámetros que en sí mismo es un problema difícil [21,14,22]. En el presente artículo se propone el uso de una Estrategia Evolutiva (ES) como motor de búsqueda dado que es un algoritmo auto-adaptativo [7,12] ya que adapta los parámetros del mismo en el transcurso de su ejecución.

La hiperheurística implementada es aplicada al problema de empacado (BPP) que es un problema de optimización combinatoria de tipo NP-difícil que busca empacar un conjunto de piezas en la menor cantidad de contenedores, donde todos los contenedores tienen la misma capacidad [20,19]. Los resultados son comparados usando la prueba no parámetrica de Friedman con los obtenidos en [25].

2. Hiperheurísticas de generación

Las Hiperheurísticas (HH) [5,4] son técnicas que trabajan en un espacio de búsqueda de heurísticas o metodologías en lugar de buscar soluciones directamente. Dentro del marco de las hiperheurísticas existen dos enfoques, el de selección que selecciona heurísticas y el de generación que se dedica a la construcción de nuevas metodologías para la solución de problemas.

Las hiperheurísticas requieren un conjunto de heurísticas o componentes heurísticos de bajo nivel que trabajan en un espacio de búsqueda de soluciones para el problema que se está trabajando, y un algoritmo que trabaje fuera del dominio del problema como núcleo de búsqueda de heurísticas.

Una parte fundamental de las hiperheurísticas de generación es la necesidad de contar con componentes heurísticos para poder construir heurísticas funcionales para los problemas con los que se está trabajando. Sin estos componentes el núcleo hiperheurístico no puede funcionar. Comúnmente un núcleo hiperheurístico de generación utiliza alguna variante de Programación Genética (GP) como elemento generador de heurísticas que han sido aplicadas en distintas ocasiones con resultados satisfactorios [3] [18,24] [26] [23] [2] [27] [25].

Uno de los algoritmos usados por las hiperheurísticas como núcleo de búsqueda son las Gramáticas Evolutivas (GE).

Las GE son una forma de programación genética basada en gramáticas que permite construir programas de forma automática usando un proceso de mapeo basado en una gramática [16] [15] [26]. Las GE trabajan con individuos formados por una cadena binaria llamada genotipo en la cual los codones son representados por un valor entero. En GE el genotipo es transformado en fenotipo que puede ser una función o programa por medio de un proceso de mapeo, como se puede ver en la Figura 1. La gramática está formada por la tupla $\{N, T, P, S\}$ donde N representa los caracteres no terminales, T los caracteres terminales, P las reglas de producción y S el carácter no terminal con el que se inicia el mapeo.

El proceso de mapeo utiliza la ecuación (1) para realizar el mapeo:

$$Regla = c\%r, \tag{1}$$

donde c es el codón actual y r el número de reglas de producción para el no terminal que está siendo mapeado.

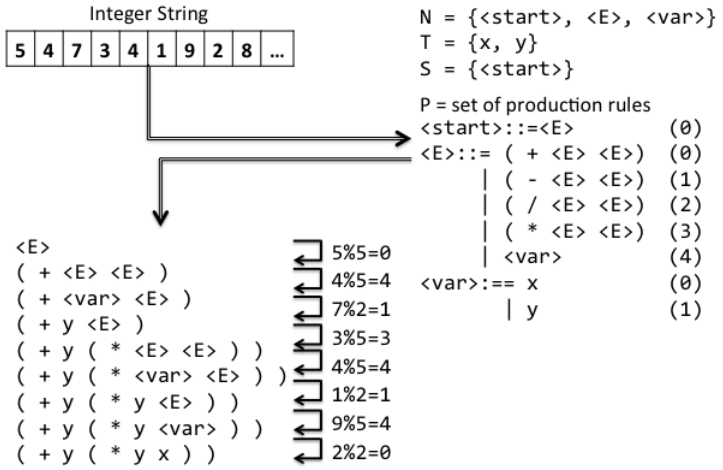

Fig. 1. Ejemplo de una gramática evolutiva BNF y su proceso de mapeo [26].

2.1. Estrategias evolutivas

Las Estrategias Evolutivas (ES) [7] son un tipo de Algoritmo Evolutivo (EA) que esta basado en el concepto de la evolución de la evolución. Las ES son algoritmos de naturaleza auto-adaptativa lo que les permite ajustar los parámetros de dicha estrategia de forma automática lo que implica ajustar el tamaño y la dirección de los pasos de muta de cada individuo [7,12] [1]. De acuerdo con [12] la auto-adaptación consiste en la búsqueda del espacio de parámetros de forma implícita en la ejecución del algoritmo.

En las ES los individuos o soluciones son representados por tuplas de la siguiente forma:

$$X(t) = (x(t), \sigma(t)), \tag{2}$$

donde $x(t)$ es el genotipo y $\sigma(t)$ es el vector de parámetros de la estrategia.

En [7,1] se indican los pasos a seguir para implementar una estrategia evolutiva:

- **Inicializar:** se inicializa el genotipo y los parámetros de la estrategia de cada individuo y se calcula el fitness.

- **Cruzar:** se genera una nueva generación de hijos a partir de la generación de padres haciendo uso de un operador de cruza.
- **Mutar:** a partir de los parámetros de la estrategia y de un operador de muta se muta a los individuos de la nueva generación.
- **Evaluar:** se calcula el fitness de cada individuo de la nueva generación.
- **Seleccionar:** los individuos con mejor fitness son seleccionados como la nueva generación de padres.

Una ES $(\mu+\lambda)$ genera λ hijos a partir de μ padres, donde $1 \leq \mu \leq \lambda < \infty$. La siguiente generación está conformada por los mejores individuos de μ y λ. La ES $(\mu + \lambda)$ implementa elitismo para conservar a los mejores padres. El Algoritmo 1 presenta la estructura de una ES $(\mu + \lambda)$ como se explica en [7].

Algorithm 1 Algoritmo de una ES $(\mu + \lambda)$.

```
1: Inicializar población de padres μ
2: while Criterio de paro no alcanzado do
3:    while hijos < λ do
4:       Cruzar individuos de población μ para obtener un hijo h
5:       Mutar h
6:       Evaluar h
7:    end while
8:    Seleccionar mejores individuos de μ y λ para nueva población μ
9: end while
```

3. Problema de empacado

El Problema de Empacado (BPP por su nombre en ingles Bin Packing Problem) [20,19] es un problema clásico de investigación operacional de tipo NP-difícil que consiste en empacar una cantidad x de piezas en la menor cantidad posible de contenedores, cada uno con la misma capacidad. El BPP unidimensional se define de la siguiente forma [19,6]. Se toma un conjunto $J = \{1, ..., n\}$ que contiene n número de piezas, cada una con un tamaño o peso w_j que deben ser empacadas en la menor cantidad de contenedores m sin exceder la capacidad c del contenedor. Por esa razón se establece que $w_j \leq c$.

Se han propuesto diferentes funciones objetivo para el problema de empacado [26], entre ellas se encuentran la ecuación (3), que mide la diferencia entre los contenedores usados y el límite superior teoríco de contenedores necesarios, y la (4), que calcula el espacio libre que queda en los contenedores.

$$Fitness = B - \frac{\sum_{i=1}^{n} w_i}{C}, \tag{3}$$

$$Fitness = 1 - \left(\frac{\sum_{i=1}^{n} ((\sum_{j=1}^{m} w_j x_{ij})/C)^2}{n} \right), \tag{4}$$

donde

- B es el número de contenedores usados,
- n el número de contenedores,
- m es el número de piezas,
- w_j es la j-ésima pieza, considerando

$$x_{ij} = \left\{ \begin{array}{c} 1 \text{ si la pieza j está en el contenedor i,} \\ 0 \text{ de otra forma,} \end{array} \right\}$$

- C es la capacidad del contenedor.

En [20] se propone agrupar las instancias de prueba en función a (5).

$$w \in [v_L X, v_U X], \tag{5}$$

donde w es la pieza, v_L es el límite inferior y v_U es el limite superior en relación a la capacidad del contenedor X.

A partir de la función (5) se obtienen tres diferentes grupos de instancias como se muestra en [25] y se propone en [9]. Los grupos se muestran en la Tabla 1.

Tabla 1. Condiciones de agrupamiento de las instancias.

Grupo	**Condición**
Triplets	$v_L = \frac{1}{4}$ y $v_U = \frac{1}{2}$
Hard	$\bar{w} \approx \frac{1}{3}$ o cercano a $\frac{1}{n}$ donde $n \geq 3$
Regular	cualquier otro caso

4. Diseño experimental

Los experimentos están configurados en base a los realizados en [26] donde se usa una GE con PSO y PESO como motor de búsqueda de un núcleo hiperheurístico de generación. En el presente trabajo se hace uso de un núcleo hiperheurístico de generación con una ES $(\mu + \lambda)$ como motor de búsqueda. La gramática 1.1 está basada en la heurística $FirstFit$ y fue propuesta en [28].

La función objetivo utilizada es la que recompensa a los contenedores con menos espacio libre propuesta en [8], Ecuación (4).

Las instancias de pruebas usadas son las siguientes:

- **Binpack:** utilizadas por E. Falkenauer [10] tienen dos tipos de instancias, uniformemente distribuidas y triplets que contienen grupos de 3 piezas que deben ser empacadas en un mismo contenedor para obtener una solución óptima.
- **BinData:** utilizadas por A. Scholl, R.Klein y C. Jürgens [19] contienen tres conjuntos de instancias uniformemente distribuidas.
- **Wäescher:** utilizadas por G. Wäescher y T. Gau [31] son 17 instancias consideradas de las mas difíciles.

$$
\begin{aligned}
\langle \text{N} \rangle &\models \langle \text{inicio} \rangle, \langle \text{expr} \rangle, \langle \text{expr2} \rangle, \langle \text{op} \rangle, \langle \text{var} \rangle \\
\langle \text{T} \rangle &\models \text{F}, \text{C}, \text{S}, +, \text{-}, /, *, \text{abs}, <= \\
\langle \text{inicio} \rangle &\models (\langle \text{expr} \rangle) <= (\langle \text{expr} \rangle) \\
\langle \text{expr} \rangle &\models (\langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle) \mid \langle \text{var} \rangle \mid abs(\langle \text{expr2} \rangle) \\
\langle \text{expr2} \rangle &\models (\langle \text{expr2} \rangle \langle \text{op} \rangle \langle \text{expr2} \rangle) \mid \langle \text{var} \rangle \\
\langle \text{var} \rangle &\models \text{F} \mid \text{C} \mid \text{S} \\
\langle \text{op} \rangle &\models + \mid * \mid \text{-} \mid /
\end{aligned}
$$

Gramática 1.1. Gramatica basada en la Heurística FirstFit [28].

- **Hard28:** utilizadas por J.E. Schoenfield son instancias que contienen entre 160 y 200 piezas con una capacidad del contenedor de 1000.

Se realizó un proceso de agrupamiento como se muestra en la sección 3. Se toma una instancia de cada uno de los grupos con la finalidad de usarla como parte del entrenamiento de la hiperheurística de generación, y los valores reportados son los que se obtienen al aplicar la heurística a todo el grupo. Los parámetros usados por la ES, PSO y PESO, el Cuadro 2, fueron obtenidos de [26].

Tabla 2. Parametros usados.

Parametros	PESO y PSO	ES
Tamaño de la Población	50	
Hijos	-	100
w	1.0	-
ϕ_1	0.8	-
ϕ_2	0.5	-
Llamadas a Función	1000	

Se realizaron 51 experimentos independientes sobre el conjunto de instancias de entrenamiento. Posteriormente se obtuvo la heurística media de los 51 experimentos para aplicarla a cada una de las instancias y poder calcular el rendimiento con la ecuación (4). Para finalizar se reagrupan las instancias en los conjuntos originales y se suman los rendimientos de todas las instancias de cada conjunto para calcular el rendimiento de la hiperheurística para cada conjunto de prueba.

5. Resultados

En el Cuadro 3 se muestran las heurísticas obtenidas para cada grupo de instancias y se puede observar que son heurísticas equivalentes a la heurística

$FirstFit$, dicha heurística es en la que se basa la gramática utilizada en el presente trabajo. El Cuadro 4 muestra los resultados obtenidos para cada conjunto de prueba después de calcular el rendimiento de cada instancia y reagruparlas en su conjunto original, los resultados de PSO y PESO se tomaron de los obtenidos en [26], dichos resultados corresponden a los obtenidos al aplicar la heurística $FirstFit$.

Tabla 3. Heurísticas generadas para cada grupo.

Grupo	**Heurística**
Triplets	$((abs(F) + S)) \leq (abs(C))$
Hard	$(abs((S + F))) \leq (C)$
Regular	$(F) \leq ((C - S))$

Tabla 4. Resultados obtenidos.

Instancia	**ES**	**PSO**	**PESO**
bin1data	316.10602	316.10602	316.10602
bin2data	93.38851	93.38851	93.38851
bin3data	1.885825	1.885825	1.885825
binpack1	2.604965	2.604965	2.604965
binpack2	2.396851	2.396851	2.396851
binpack3	2.133326	2.133326	2.133326
binpack4	1.935722	1.935722	1.935722
binpack5	0.0	0.0	0.0
binpack6	0.0	0.0	0.0
binpack7	0.0	0.0	0.0
binpack8	0.0	0.0	0.0
hard28	0.65535	0.65535	0.65535

Se aplicó la prueba estadística no paramétrica de Friedman para determinar si existen diferencias en el rendimiento de los algoritmos, el valor de la prueba es de 0.0 con un p-valor de 1.0 lo cual implica que no existe evidencia estadística para decir que existen diferencias en el rendimiento de los algoritmos.

6. Conclusiones

En base a los resultados obtenidos usando la Estrategia Evolutiva $(\mu + \lambda)$ como motor de búsqueda de una hiperheurística de generación podemos concluir que dicha metaheurística permite obtener resultados que compiten con los del estado del arte.

La prueba no paramétrica de Friedman no mostró evidencia de que la Optimización por Cumulo de Partícula, Optimización Evolutiva por Cúmulo de Partículas o la Estrategia Evolutiva tuvieran un mejor desempeño al ser usadas como motor de búsqueda en la hiperheurística de generación; sin embargo, los parámetros tanto de la Optimización por Cumulo de Partícula y Optimización Evolutiva por Cúmulo de Partículas fueron obtenidos mediante un proceso de optimización de parámetros.

Los resultados obtenidos muestran que es posible sustituir el proceso de optimización de parámetros, que conlleva a aumentar el tiempo de computo al buscar los paramétros de dichas metaheurísticas para que puedan ser usadas como núcleo de la hiperheurística, por un metaheurística que tiene un proceso auto-adaptativo.

Referencias

1. Bäck, T., Foussette, C., Krause, P.: Evolution Strategies, pp. 7–45. Springer Berlin Heidelberg, Berlin, Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-40137-4_2
2. Basgalupp, M.P., Barros, R.C., Barabasz, T.: A grammatical evolution based hyper-heuristic for the automatic design of split criteria. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation. pp. 1311–1318. GECCO '14, ACM, New York, NY, USA (2014), http://doi.acm.org/10.1145/2576768.2598327
3. Burke, E.K., Hyde, M.R., Kendall, G.: Grammatical evolution of local search heuristics. IEEE Transactions on Evolutionary Computation 16(3), 406–417 (June 2012)
4. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: a survey of the state of the art. Journal of the Operational Research Society 64(12), 1695–1724 (2013), http://dx.doi.org/10.1057/jors.2013.71
5. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: A Classification of Hyper-heuristic Approaches, pp. 449–468. Springer US, Boston, MA (2010), http://dx.doi.org/10.1007/978-1-4419-1665-5_15
6. Dokeroglu, T., Cosar, A.: Optimization of one-dimensional bin packing problem with island parallel grouping genetic algorithms. Computers & Industrial Engineering 75, 176 – 186 (2014), http://www.sciencedirect.com/science/article/pii/S036083521400182X
7. Engelbrecht, A.P.: Computational intelligence: an introduction. John Wiley & Sons (2007)
8. Falkenauer, E., Delchambre, A.: A genetic algorithm for bin packing and line balancing. In: Proceedings 1992 IEEE International Conference on Robotics and Automation. pp. 1186–1192 vol.2 (May 1992)
9. Falkenauer, E.: Tapping the Full Power of Genetic Algorithm through Suitable Representation and Local Optimization: Application to Bin Packing, pp. 167–182. Springer Berlin Heidelberg, Berlin, Heidelberg (1995), http://dx.doi.org/10.1007/978-3-642-61217-6_8
10. Falkenauer, E.: A hybrid grouping genetic algorithm for bin packing. Journal of Heuristics 2(1), 5–30 (1996), http://dx.doi.org/10.1007/BF00226291

11. Kheiri, A., Özcan, E.: An iterated multi-stage selection hyper-heuristic. European Journal of Operational Research 250(1), 77 – 90 (2016), http://www.sciencedirect.com/science/article/pii/S0377221715008255
12. Kramer, O.: Evolutionary self-adaptation: a survey of operators and strategy parameters. Evolutionary Intelligence 3(2), 51–65 (2010), http://dx.doi.org/10.1007/s12065-010-0035-y
13. Martin, M.A., Tauritz, D.R.: Hyper-heuristics: A study on increasing primitive-space. In: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation. pp. 1051–1058. GECCO Companion '15, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2739482.2768457
14. Nannen, V., Smit, S.K., Eiben, A.E.: Costs and Benefits of Tuning Parameters of Evolutionary Algorithms, pp. 528–538. Springer Berlin Heidelberg, Berlin, Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-87700-4_53
15. O'Neil, M., Ryan, C.: Grammatical Evolution, pp. 33–47. Springer US, Boston, MA (2003), http://dx.doi.org/10.1007/978-1-4615-0447-4_4
16. O'Neill, M., Ryan, C.: Grammatical evolution. IEEE Transactions on Evolutionary Computation 5(4), 349–358 (Aug 2001)
17. Pillay, N.: A review of hyper-heuristics for educational timetabling. Annals of Operations Research 239(1), 3–38 (2016), http://dx.doi.org/10.1007/s10479-014-1688-1
18. Sabar, N.R., Ayob, M., Kendall, G., Qu, R.: Grammatical evolution hyper-heuristic for combinatorial optimization problems. IEEE Transactions on Evolutionary Computation 17(6), 840–861 (Dec 2013)
19. Scholl, A., Klein, R., Jürgens, C.: Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. Computers & Operations Research 24(7), 627 – 645 (1997), http://www.sciencedirect.com/science/article/pii/S0305054896000822
20. Schwerin, P., Wäscher, G.: The bin-packing problem: A problem generator and some numerical experiments with ffd packing and mtp. International Transactions in Operational Research 4(5), 377 – 389 (1997), http://www.sciencedirect.com/science/article/pii/S0969601697000257
21. Smit, S.K., Eiben, A.E.: Comparing parameter tuning methods for evolutionary algorithms. In: 2009 IEEE Congress on Evolutionary Computation. pp. 399–406 (May 2009)
22. Smit, S.K., Eiben, A.E.: Parameter Tuning of Evolutionary Algorithms: Generalist vs. Specialist, pp. 542–551. Springer Berlin Heidelberg, Berlin, Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-12239-2_56
23. Sosa-Ascencio, A., Ochoa, G., Terashima-Marin, H., Conant-Pablos, S.E.: Grammar-based generation of variable-selection heuristics for constraint satisfaction problems. Genetic Programming and Evolvable Machines 17(2), 119–144 (2016), http://dx.doi.org/10.1007/s10710-015-9249-1
24. Sotelo-Figueroa, M.A., Soberanes, H.J.P., Carpio, J.M., Huacuja, H.J.F., Reyes, L.C., Soria-Alcaraz, J.A.: Evolving and reusing bin packing heuristic through grammatical differential evolution. In: 2013 World Congress on Nature and Biologically Inspired Computing. pp. 92–98 (Aug 2013)
25. Sotelo-Figueroa, M.A., Puga Soberanes, H.J., Carpio, J.M., Fraire Huacuja, H.J., Reyes, L.C., Soria Alcaraz, J.A.: Clustering Bin Packing Instances for Generating a Minimal Set of Heuristics by Using Grammatical Evolution, pp. 151–162. Springer International Publishing, Cham (2015), http://dx.doi.org/10.1007/978-3-319-10960-2_10

26. Sotelo-Figueroa, M.A., Soberanes, H.J.P., Carpio, J.M., Huacuja, H.J.F., Reyes, L.C., Soria-Alcaraz, J.A.: Improving the bin packing heuristic through grammatical evolution based on swarm intelligence. Mathematical Problems in Engineering 2014 (2014), http://dx.doi.org/10.1155/2014/545191
27. Sotelo-Figueroa, M.A., Soberanes, H.J.P., Carpio, J.M., Fraire Huacuja, H.J., Reyes, L.C., Alcaraz, J.A.S., Espinal, A.: Generating Bin Packing Heuristic Through Grammatical Evolution Based on Bee Swarm Optimization, pp. 655–671. Springer International Publishing, Cham (2017), http://dx.doi.org/10.1007/978-3-319-47054-2_43
28. Sotelo-Figueroa, M.A., Soberanes, H.J.P., Carpio, J.M., Fraire Huacuja, H.J., Reyes, L.C., Soria Alcaraz, J.A.: Evolving Bin Packing Heuristic Using Micro-Differential Evolution with Indirect Representation, pp. 349–359. Springer Berlin Heidelberg, Berlin, Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-33021-6_28
29. Swiercz, A., Burke, E.K., Cichenski, M., Pawlak, G., Petrovic, S., Zurkowski, T., Blazewicz, J.: Unified encoding for hyper-heuristics with application to bioinformatics. Central European Journal of Operations Research 22(3), 567–589 (2014), http://dx.doi.org/10.1007/s10100-013-0321-8
30. Urra, E., Cabrera-Paniagua, D., Cubillos, C.: Towards a distributed hyperheuristic deploy architecture. In: Proceedings of the 7th Euro American Conference on Telematics and Information Systems. pp. 31:1–31:4. EATIS '14, ACM, New York, NY, USA (2014), http://doi.acm.org/10.1145/2590651.2590682
31. Wäscher, G., Gau, T.: Heuristics for the integer one-dimensional cutting stock problem: A computational study. Operations-Research-Spektrum 18(3), 131–144 (1996), http://dx.doi.org/10.1007/BF01539705