

# Búsqueda de Entorno Variable

Pierre Hansen  
GERAD and HEC Montreal  
3000 ch. de la Cote-Sainte-Catherine  
Montréal H3T 2A7, Canadá  
pierre@crt.umontreal.ca

Nenad Mladenović  
Mathematical Institute, Serbian Academy of Science  
Kneza Mihajla 35, 11000 Belgrade, Yugoslavia  
nenad@mi.sanu.ac.yu

José Andrés Moreno Pérez\*  
Departamento de Estadística, I.O. y Computación  
Universidad de La Laguna, 38271 La Laguna, España  
jamoreno@ull.es

## Resumen

La Búsqueda de Entorno Variable (*Variable Neighbourhood Search*, VNS) es una metaheurística reciente para resolver problemas de optimización cuya idea básica es el cambio sistemático de entorno dentro de una búsqueda local. En este artículo presentamos las reglas básicas de la VNS y sus extensiones. Se resumen brevemente las aplicaciones más relevantes en Inteligencia Artificial que incluyen tareas clásicas, como la satisfacción de restricciones, la planificación, el aprendizaje, además del descubrimiento asistido por ordenador y la solución heurística de una variedad de problemas de optimización. Finalmente se incluyen algunas reflexiones sobre aspectos esenciales de las metaheurísticas y del análisis de los procesos de solución heurística, y la contribución de los trabajos con la VNS para estas cuestiones.

## Abstract

The Variable Neighbourhood Search (VNS) is a recent metaheuristic for solving optimization problems whose basic idea is systematic change of neighborhood within a local search. In this paper we present basic rules of the VNS and its extensions. We briefly summarize the most relevant applications in Artificial Intelligence. These applications comprise classic tasks, like constraints satisfaction, planning or learning, in addition to computer-assisted discovery and heuristic solution of a variety of optimization problems. Finally we include some reflections on the essential aspects of the metaheuristics and the analysis of the heuristic solution procedures, and the contribution of the works with VNS for these questions.

## 1. Introducción

La realización de gran cantidad de tareas de Inteligencia Artificial (IA) se puede interpretar como la resolución de un problema de op-

timización. Las Metaheurísticas son estrategias generales para diseñar procedimientos heurísticos para resolver un problema de optimización mediante un proceso de búsqueda en un cierto espacio de soluciones alternativas. Los procesos de búsqueda heurística están generalmente basados en transformaciones de las alternativas que determinan una estructura de entornos en

---

\*El trabajo del tercer co-autor ha sido parcialmente financiado con el proyecto TIC2002-04242-C03-01 que en 70 % son fondos FEDER

el espacio de soluciones. La Búsqueda de Entorno Variable (*Variable Neighbourhood Search*, VNS) es una metaheurística propuesta sólo hace unos años [62], [41] que está basada en un principio simple: cambiar sistemáticamente de estructura de entornos dentro de la búsqueda [43], [49]. Su desarrollo ha sido rápido, con muchos artículos ya publicados o pendientes de aparecer [47]. Se han realizado muchas extensiones, principalmente para permitir la solución de problemas de gran tamaño [45]. En la mayoría de ellas, se ha hecho un esfuerzo por mantener la simplicidad del esquema básico [51], [50].

En la siguiente sección se exponen las reglas básicas de la VNS. Las extensiones, incluyendo los híbridos, se consideran en la sección 3. En la sección 4 se repasan las aplicaciones prácticas más importantes de la VNS en IA. En la sección 5 se profundiza en algunas cuestiones relevantes de esta metaheurística. La sección 6 incluye algunas consideraciones sobre la VNS en relación a los aspectos esenciales de las metaheurísticas. El trabajo finaliza con unas breves conclusiones.

## 2. Esquemas Fundamentales

Un problema de optimización consiste en encontrar, dentro de un conjunto  $X$  de soluciones factibles, la que optimiza una función  $f(x)$ . Si el problema es de minimización se formula como sigue:

$$\min\{f(x)|x \in X\} \quad (1)$$

donde  $x$  representa una *solución* alternativa,  $f$  es la *función objetivo* y  $X$  es el *espacio de soluciones* factibles del problema. Una *solución óptima*  $x^*$  (o mínimo global) del problema es una solución factible donde se alcanza el mínimo de (1).

Una *estructura de entornos* en el espacio de soluciones  $X$  es una aplicación  $\mathcal{N} : X \rightarrow 2^X$  que asocia a cada solución  $x \in X$  un entorno de soluciones  $\mathcal{N}(x) \subset X$ , que se dicen *vecinas* de  $x$ . Las metaheurísticas de búsqueda local aplican una transformación o movimiento a la solución de búsqueda y por tanto utilizan, explícita o implícitamente, una estructura de entornos. Denotemos por  $\mathcal{N}_k$ ,  $k = 1, \dots, k_{max}$ , a un conjunto finito de estructuras de entornos en el espacio

$X$ . Los entornos  $\mathcal{N}_k$  pueden ser inducidos por una o más métricas introducidas en el espacio de soluciones  $X$ . La mayoría de las heurísticas de búsqueda local usan sólo una estructura de entornos.

Una solución  $x^* \in X$  es un *mínimo global* del problema (1) si no existe una solución  $x \in X$  tal que  $f(x) < f(x^*)$ . Decimos que  $x' \in X$  es un *mínimo local* con respecto a  $\mathcal{N}_k$ , si no existe una solución  $x \in \mathcal{N}_k(x') \subseteq X$  tal que  $f(x) < f(x')$ . Una búsqueda local descendente cambia la solución actual por otra solución mejor de su entorno, por tanto tienen el peligro de quedarse atascada en un mínimo local. Las metaheurísticas basadas en procedimientos de búsqueda local aplican distintas formas de continuar la búsqueda después de encontrar el primer óptimo local.

La VNS está basada en tres hechos simples:

1. Un mínimo local con una estructura de entornos no lo es necesariamente con otra.
2. Un mínimo global es mínimo local con todas las posibles estructuras de entornos.
3. Para muchos problemas, los mínimos locales con la misma o distinta estructura de entornos están relativamente cerca.

Esta última observación, que es empírica, implica que los óptimos locales proporcionan información acerca del óptimo global. Puede ser, por ejemplo, que ambas soluciones tengan características comunes. Sin embargo, generalmente no se conoce cuales son esas características. Es procedente, por tanto, realizar un estudio organizado en las proximidades de este óptimo local, hasta que se encuentre uno mejor.

Los hechos 1 a 3 sugieren el empleo de varias estructura de entornos en las búsquedas locales para abordar un problema de optimización. El cambio de estructura de entornos se puede realizar de forma determinística, estocástica, o determinística y estocástica a la vez.

### 2.1. VNS Descendente

Una búsqueda local determina una mejor solución del entorno de la solución actual. La clásica

búsqueda *greedy* descendente consiste en reemplazar iterativamente la solución actual por el resultado de la búsqueda local, mientras se produzca mejora. Si se realiza un cambio de estructura de entornos de forma determinística cada vez que se llega a un mínimo local, se obtiene la **búsqueda de entorno variable descendente** (*Variable Neighbourhood Descent*, VND). Los pasos de la VND se muestran en la figura 1.

---

#### Inicialización

Seleccionar el conjunto de estructuras de entornos  $\mathcal{N}_k, k = 1, \dots, k_{max}$ , que se usarán en el descenso; encontrar una solución inicial  $x$ ;

#### Iteraciones

Repetir, hasta que no se obtenga mejora, la siguiente secuencia:

- (1) Hacer  $k \leftarrow 1$ ;
  - (2) Repetir, hasta que  $k = k_{max}$ , los pasos:
    - (a) Exploración del entorno Encontrar la mejor solución  $x'$  del  $k$ -ésimo entorno de  $x$  ( $x' \in \mathcal{N}_k(x)$ );
    - (b) Moverse o no Si la solución obtenida  $x'$  es mejor que  $x$ , hacer  $x \leftarrow x'$  y  $k \leftarrow 1$ ; en otro caso, hacer  $k \leftarrow k + 1$ .
- 

**Figura 1: VNS Descendente; VND**

La solución final proporcionada por el algoritmo es un mínimo local con respecto a todas las  $k_{max}$  estructuras de entornos, y por tanto la probabilidad de alcanzar un mínimo global es mayor que usando una sola estructura.

Además del orden *secuencial* de las estructuras de entornos en la VND anterior, se puede desarrollar una estrategia *anidada*. Supongamos, por ejemplo, que  $k_{max} = 3$ . Entonces una posible estrategia anidada es: ejecutar la VND de la figura 1 para las dos primeras estructuras de entornos, sobre cada  $x'$  que pertenezca al tercer entorno de  $x$  ( $x' \in \mathcal{N}_3(x)$ ). Esta VNS ha sido aplicado en [8], [48] y [4].

## 2.2. VNS Reducida

La **búsqueda de entorno variable reducida** (*Reduced Variable Neighbourhood Search*, RVNS) se obtiene si se seleccionan soluciones

aleatorias de  $\mathcal{N}_k(x)$ , sin aplicarles a continuación un descenso. Los pasos de la RVNS se presentan en la figura 2.

---

#### Inicialización

Seleccionar el conjunto de estructuras de entornos  $\mathcal{N}_k, k = 1, \dots, k_{max}$ , que se usarán en la búsqueda; encontrar una solución inicial  $x$ ; elegir una condición de parada;

#### Iteraciones

Repetir, hasta que se cumpla la condición de parada, la siguiente secuencia:

- (1) Hacer  $k \leftarrow 1$ ;
  - (2) Repetir, hasta que  $k = k_{max}$ , los pasos:
    - (a) Agitación Generar al azar una solución  $x'$  del  $k$ -ésimo entorno de  $x$  ( $x' \in \mathcal{N}_k(x)$ );
    - (b) Moverse o no Si la solución obtenida  $x'$  es mejor que  $x$ , hacer  $x \leftarrow x'$  y  $k \leftarrow 1$ ; en otro caso, hacer  $k \leftarrow k + 1$ .
- 

**Figura 2: VNS Reducida; RVNS**

La RVNS es útil para instancias muy grandes de problemas en las que la búsqueda local es muy costosa. Se observa que el mejor valor para el parámetro  $k_{max}$  es frecuentemente 2. Como condición de parada se usa generalmente el máximo número de iteraciones entre dos mejoras. La RVNS se ha mostrado superior a un método de Monte-Carlo en un problema min-max continuo [65] y a la heurística del *intercambio rápido* de [78] al aplicarla al problema de la  $p$ -mediana.

## 2.3. VNS Básica

La **búsqueda de entorno variable básica** (*Basic Variable Neighbourhood Search*, BVNS) combina cambios determinísticos y aleatorios de estructura de entornos. Los pasos de la VNS básica se dan en la figura 3.

La condición de parada puede ser, por ejemplo, el máximo tiempo de CPU permitido, el máximo número de iteraciones, o el máximo número de iteraciones entre dos mejoras. Frecuentemente los entornos  $\mathcal{N}_k$  sucesivos están anidados. Obsérvese que la solución  $x'$  se ge-

---

### Inicialización

Seleccionar un conjunto de estructuras de entornos  $\mathcal{N}_k, k = 1, \dots, k_{max}$ , que se usarán en la búsqueda; encontrar una solución inicial  $x$ ; elegir una condición de parada.

### Iteraciones

Repetir, hasta que se cumpla la condición de parada, la siguiente secuencia:

- (1) Hacer  $k \leftarrow 1$ ;
  - (2) Repetir, hasta que  $k = k_{max}$ , los pasos:
    - (a) Agitación Generar al azar una solución  $x'$  del  $k$ -ésimo entorno de  $x$  ( $x' \in \mathcal{N}_k(x)$ );
    - (b) Búsqueda local Aplicar algún método de búsqueda local con  $x'$  como solución inicial; denótese con  $x''$  el mínimo local así obtenido.
    - (c) Moverse o no Si la solución obtenida  $x''$  es mejor que  $x$ , hacer  $x \leftarrow x''$  y  $k \leftarrow 1$ ; en otro caso, hacer  $k \leftarrow k + 1$ .
- 

**Figura 3: VNS Básica; BVNS**

nera al azar en el paso (2a) para evitar el ciclo, que puede ocurrir si se usa cualquier regla determinística.

## 2.4. VNS General

La búsqueda local del paso (2b) de la BVNS puede ser sustituida por la VND. Los pasos de la **búsqueda de entorno variable general** (*General Variable Neighbourhood Search*, GVNS) se muestran en la figura 4.

El uso de la búsqueda de entorno variable general (GVNS) ha dado lugar a las aplicaciones más exitosas aparecidas recientemente (ver, por ejemplo, [1], [8], [13], [14], [15], [16], [48], [72], [73]).

## 3. Extensiones de la VNS

Se han propuesto en la literatura diversas formas de extender la VNS para dotarlo de algunas características adicionales. Describimos en primer lugar varias formas sencillas de

---

### Inicialización

Seleccionar un conjunto de estructuras de entornos  $\mathcal{N}_k$ , para  $k = 1, \dots, k_{max}$ , que se usarán en la agitación; seleccionar un conjunto de estructuras de entornos  $\mathcal{N}'_j$ , para  $j = 1, \dots, j_{max}$ , que se usarán en el descenso; encontrar una solución inicial  $x$ ; elegir una condición de parada;

### Iteraciones

Repetir, hasta que se cumpla la condición de parada, la siguiente secuencia:

- (1) Hacer  $k \leftarrow 1$ ;
  - (2) Repetir, hasta que  $k = k_{max}$ , los pasos:
    - (a) Agitación Generar al azar una solución  $x'$  del entorno  $\mathcal{N}_k(x)$ ;
    - (b) Búsqueda local Aplicar la VND con las estructuras de entornos  $\mathcal{N}'_j$ , para  $j = 1, \dots, j_{max}$ ; denótese con  $x''$  la solución así obtenida.
    - (c) Moverse o no Si la solución obtenida  $x''$  es mejor que  $x$ , hacer  $x \leftarrow x''$  y  $k \leftarrow 1$ ; en otro caso, hacer  $k \leftarrow k + 1$ .
- 

**Figura 4: VNS General; GVNS**

realizar estas extensiones en la subsección 3.1. Las tres siguientes subsecciones se dedican a otras tantas extensiones que constituyen mejoras prácticas de la VNS que han permitido resolver con éxito problemas muy grandes: la búsqueda de entorno variable con descomposición (VNDS), la búsqueda de entorno variable sesgada (SVNS) y la búsqueda de entorno variable paralela (PVNS). En la subsección 3.4. se consideran las extensiones por hibridación de la VNS.

### 3.1. Extensiones básicas

Las primeras extensiones se derivan directamente de la VNS básica. La BVNS es un método descendente de la primera mejora con aleatorización. Sin mucho esfuerzo adicional se transforma en un método ascendente-descendente: en el paso (2c) hacer también  $x \leftarrow x''$  con alguna probabilidad, incluso si la solución es peor que la actual (o que la mejor solución encontrada hasta el momento). También se puede transformar en una búsqueda de la

mejor mejora: aplicar un movimiento al mejor entorno  $k^*$  entre todos los  $k_{max}$  entornos. Otras extensiones de la VNS son encontrar la solución  $x'$  en el paso (2a) como la mejor entre  $b$  (un parámetro) soluciones generadas aleatoriamente en el  $k$ -ésimo entorno, o introducir  $k_{min}$  y  $k_{paso}$ , dos parámetros que controlan el proceso de cambio de entorno: en el algoritmo anterior, en vez de  $k \leftarrow 1$  hacer  $k \leftarrow k_{min}$  y en vez de  $k \leftarrow k + 1$  hacer  $k \leftarrow k + k_{paso}$ .

### 3.2. VNS con Descomposición

La **búsqueda de entorno variable con descomposición** (*Variable Neighbourhood Decomposition Search*, VNDS) [52] extiende la VNS en un esquema de entorno variable en dos niveles basado en la descomposición del problema. Sus pasos son los presentados en la figura 5.

---

#### Inicialización

Seleccionar un conjunto de estructuras de entornos  $\mathcal{N}_k$ , para  $k = 1, \dots, k_{max}$ , que se usarán en el descenso; encontrar una solución inicial  $x$ ; elegir una condición de parada;

#### Iteraciones

Repetir, hasta que se cumpla la condición de parada, la siguiente secuencia :

- (1) Hacer  $k \leftarrow 1$ ;
  - (2) Repetir, hasta que  $k = k_{max}$ , los pasos:
    - (a) Agitación Generar al azar una solución  $x'$  del  $k$ -ésimo entorno de  $x$  ( $x' \in \mathcal{N}_k(x)$ ); denotemos por  $y$  el conjunto de los atributos de la solución presentes en  $x'$  pero no en  $x$  ( $y = x' \setminus x$ ).
    - (b) Búsqueda local Buscar el óptimo local en el espacio de  $y$  por inspección o por alguna heurística; denótese por  $y'$  a la mejor solución encontrada y por  $x''$  a la correspondiente solución en el espacio completo  $X$  ( $x'' = (x' \setminus y) \cup y'$ );
    - (c) Moverse o no Si la solución obtenida  $x''$  es mejor que  $x$ , hacer  $x \leftarrow x''$  y  $k \leftarrow 1$ ; en otro caso, hacer  $k \leftarrow k + 1$ .
- 

**Figura 5: VNS con Descomposición; VNDS**

Nótese que la única diferencia entre la VNS

básica y la VNDS está en el paso (2b): en vez de aplicar algún método de búsqueda local en el espacio completo  $X$  (empezando desde  $x' \in \mathcal{N}_k(x)$ ), en la VNDS se resuelve en cada iteración un subproblema en un subespacio  $X_k \subseteq \mathcal{N}_k(x)$  con  $x' \in X_k$ . Cuando la búsqueda local utilizada en este paso es también la VNS, aparece un esquema VNS en dos niveles.

La VNDS puede verse como la inclusión del esquema de aproximación sucesiva, que ha sido históricamente muy utilizada en optimización combinatoria, en la estructura de la VNS. Otras aplicaciones más simples de esta técnica, donde el tamaño de los subproblemas a optimizar al nivel inferior está fijo son la búsqueda de entornos grandes (LNS) [75] y el método POPMUSIC [77].

### 3.3. VNS Sesgada

Una vez que se ha encontrado la mejor solución en una gran región es necesario alejarse bastante para obtener una mejor. La **búsqueda de entorno variable sesgada** (*Skewed Variable Neighbourhood Search*, SVNS) [39] afronta la exploración de valles alejados de la solución actual. Las soluciones generadas al azar en entornos muy lejanos pueden diferenciarse substancialmente de la solución actual; por lo que la VNS degenera, en algún sentido, en una heurística de arranque múltiple (en la que se realizan iterativamente descensos desde soluciones generadas al azar). Por tanto, la VNS sesgada incorpora una compensación por la distancia desde la solución actual para evitar este inconveniente. Sus pasos son presentados en la figura 6.

La SVNS usa una función  $\rho(x, x'')$  para medir la distancia entre la solución actual  $x$  y el óptimo local encontrado  $x''$ . La distancia usada para definir los entornos  $\mathcal{N}_k$  puede también utilizarse con este propósito. La elección del parámetro  $\alpha$  debe permitir la exploración de valles lejanos a  $x$  cuando  $f(x'')$  es algo mayor que  $f(x)$ , pero no demasiado (en otro caso siempre se abandonaría la solución  $x$ ) y evitar movimientos frecuentes desde  $x$  a una solución cercana para volver a  $x$ . Un buen valor para  $\alpha$  tiene que determinarse experimentalmente en cada caso.

---

#### Inicialización

Seleccionar un conjunto de estructuras de entornos  $\mathcal{N}_k$ , para  $k = 1, \dots, k_{max}$ , que se usarán en la búsqueda; encontrar una solución inicial  $x$  y su valor  $f(x)$ ; hacer  $x^* \leftarrow x$ ,  $f^* \leftarrow f(x)$ ; elegir una condición de parada y un parámetro  $\alpha$ ;

#### Iteraciones

Repetir, hasta que se cumpla la condición de parada, la siguiente secuencia:

- (1) Hacer  $k \leftarrow 1$ ;
- (2) Repetir, hasta que  $k = k_{max}$ , los pasos:
  - (a) Agitación Generar al azar una solución  $x'$  del  $k$ -ésimo entorno de  $x$ , ( $x' \in \mathcal{N}_k(x)$ );
  - (b) Búsqueda local Aplicar algún método de búsqueda local con  $x'$  como solución inicial; denótese por  $x''$  al óptimo local obtenido;
  - (c) Mejorar o no Si  $f(x'') < f^*$  hacer  $f^* \leftarrow f(x'')$  y  $x^* \leftarrow x''$ ;
  - (d) Moverse o no Si  $f(x'') - \alpha\rho(x, x'') < f(x)$  hacer  $x \leftarrow x''$  y  $k \leftarrow 1$ ; en otro caso, hacer  $k \leftarrow k + 1$ .

---

**Figura 6: VNS Sesgada; SVNS**

### 3.4. VNS Paralela

Las **búsquedas de entorno variable paralelas** (*Parallel Variable Neighbourhood Search*, PVNS) constituyen la tercera extensión. Se han propuesto en [31] y [19] diversas formas de paralelizar la VNS que han sido aplicadas al problema de la  $p$ -mediana. En [31] se analizan tres de ellas: (i) paralelizar la búsqueda local, (ii) aumentar el número de soluciones obtenidas del entorno actual y realizar búsquedas locales en paralelo desde cada una de ellas y (iii) hacer lo mismo que en (ii) pero actualizando la información sobre la mejor solución encontrada. La segunda paralelización, cuyos pasos se muestran en la figura 7, es la que dio mejores resultados.

En [19] se muestra que asignando diferentes entornos a cada procesador e interrumpiendo su trabajo tan pronto como se mejore la solución da muy buenos resultados. En [67] también se sugieren tres estrategias para paralelizar

---

#### Inicialización

Seleccionar un conjunto de estructuras de entornos  $\mathcal{N}_k$ ,  $k = 1, \dots, k_{max}$ , que se usarán en la búsqueda; encontrar una solución inicial  $x$ ; elegir una condición de parada;

#### Iteraciones

Repetir, hasta que se cumpla la condición de parada, la siguiente secuencia:

- (1) Hacer  $k \leftarrow 1$ ;
- (2) Repetir en paralelo, hasta que  $k = k_{max}$ , para cada procesador  $p$  los pasos :
  - (a) Agitación Generar al azar una solución  $x'_p$  del  $k$ -ésimo entorno de  $x$  ( $x'_p \in \mathcal{N}_k(x)$ );
  - (b) Búsqueda local Aplicar algún método de búsqueda local con  $x'_p$  como solución inicial; denótese con  $x''_p$  el mínimo local así obtenido.
  - (c) Moverse o no Si la solución obtenida  $x''_p$  es mejor que  $x$ , hacer  $x \leftarrow x''_p$  y  $k \leftarrow 1$ ; en otro caso, hacer  $k \leftarrow k + 1$ .

---

**Figura 7: VNS Paralela; PVNS**

la VNS para resolver el problema del viajante comprador.

### 3.5. Híbridos

Dado que el cambio sistemático de estructura de entornos es una herramienta simple y muy potente, otra forma de extender la VNS ha sido incorporarla a otras metaheurísticas. Estas propuestas han dado lugar a diversas metaheurísticas híbridas.

La búsqueda tabú (*Tabu Search*, TS) [36], [37], [38] generalmente usa una estructura de entornos con respecto a la que ejecuta movimientos de ascenso y descenso explotando diferentes tipos de memoria. En principio hay dos maneras de hacer híbridos de VNS y TS: usar algún tipo de memoria para orientar la búsqueda dentro de la VNS o usar la VNS dentro de la TS. En [74], [10], [57] y [6] se proponen híbridos del primer tipo y en [8] y [22] del segundo tipo.

La metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure* [27] consta de

dos fases; en la primera fase se construyen soluciones usando un procedimiento *greedy* aleatorizado y en la segunda, las soluciones se mejoran por alguna búsqueda local o un método enumerativo. Una forma natural de hibridizar la VNS con GRASP es usar la VNS en la segunda fase de GRASP lo que ha sido aplicado en [61], [73], [1], [13], [67] [28].

La búsqueda Multi-arranque (*MultiStart Search*, MS) [60] es una metaheurística clásica que, para evitar el estancamiento de un procedimiento descendente en un óptimo local, sencillamente reinicia la búsqueda desde otra solución. En [4] se propone y analiza una heurística híbrida entre la VNS y la MS consistente en reiniciar la VNS desde otra solución generada al azar del espacio  $X$ , cuando se estanca en un mínimo local, por no encontrar ninguna mejora a través de los entornos  $\mathcal{N}_1(x), \dots, \mathcal{N}_{k_{max}}(x)$  de la solución  $x$ .

## 4. VNS en IA

Las aplicaciones de la VNS han crecido rápidamente en número desde el año 1995 en que se presentó el primer trabajo sobre esta metaheurística [62]. La VNS ha mostrado sus ventajas en diversas tareas importantes de IA aunque en la mayoría de los trabajos se aplican a problemas de optimización. Los problemas de optimización son el núcleo de la Investigación Operativa y sus aplicaciones aparecen en numerosos campos. En Inteligencia Artificial aparecen más problemas de satisfacción de restricciones en los que las heurísticas deben buscar una solución factible. Ambos tipos de problemas son menos diferentes de lo que pueda parecer a simple vista, ya que los problemas de satisfacción de restricciones se pueden expresar como problemas de optimización en una variedad de formas. Por ejemplo, como la minimización de la suma de variables artificiales que representen la violación de restricciones. Por otro lado, muchos problemas de optimización pueden formularse como problemas de satisfacción de restricciones.

En los últimos años la programación por restricciones (*Constraint Programming*, CP) ha atraído gran atención en IA por su potencial al abordar la resolución práctica de problemas de la vida real. La característica más relevante es

la forma declarativa de tratar los problemas dado que se especifican las restricciones que deben cumplir las soluciones sin especificar un procedimiento computacional para obtenerlas. Ejemplos de aplicación de búsquedas del tipo de la VNS en un contexto de CP son [34], [58] y [59].

En las siguientes subsecciones describimos los aspectos más relevantes de la aplicación de la VNS a distintos campos de IA. En particular consideramos, en la subsección 4.1, la aplicación de la VNS a algunos problemas clásicos de IA. En la subsección 4.2 se incluye la aplicación al descubrimiento en teoría de grafos, y por último, en la sección 4.3, la aplicación de la VNS a diversos problemas relevantes de optimización combinatoria donde destacan problemas surgidos en la planificación logística y distribución.

### 4.1. Tareas clásicas de IA

La VNS se ha aplicado a diversos problemas clásicos de la IA. Entre ellos destacan los problemas de satisfactibilidad, el aprendizaje en Redes Bayesianas, los problemas de Clasificación y la Planificación.

Los **problemas de satisfactibilidad** [24] han recibido muchísima atención en IA. El problema de la **satisfactibilidad**, en forma clausal, consiste en determinar si un conjunto de cláusulas (todas en forma disyuntiva o todas en forma conjuntiva) construida sobre  $n$  variables lógicas tiene solución o no. En [39] se aplica la VNS y la SVNS para el *problema de satisfactibilidad máxima ponderada* que consiste en, asignado un peso positivo a cada cláusula, encontrar la solución que maximice la suma de los pesos de las cláusulas satisfechas.

Las **redes bayesianas** [68] constituyen una de las metodologías más importantes para tratar las relaciones de dependencia. Las redes bayesianas son estructuras que permiten representar la dependencia o independencia entre variables aleatorias y constituyen herramientas eficientes para realizar inferencias a través de la fórmula de Bayes. Una red bayesiana se representa mediante un grafo dirigido con el conjunto de variables como vértices y las aristas representado dependencias. Un grafo completo como estructura de una red bayesiana implica demasiados parámetros para ser útil, por lo que

se busca una red con un menor número de aristas y con cierto nivel de ajuste a los datos. El tipo de red bayesiana a considerar se restringe a un tipo de grafo específico; en muchos casos a un árbol o a una cadena. El problema es construir una red bayesiana que se ajuste y aprenda de un conjunto de realizaciones conjuntas de las variables. En [12] se usa una VNS basada en ordenaciones para el aprendizaje en redes bayesianas asociadas a permutaciones. Para cada permutación de las variables se busca la red bayesiana entre las compatibles con la permutación.

El agrupamiento o *clustering* ([54]) es una técnica relevante del análisis exploratorio de datos. El problema se contempla en el contexto de la **clasificación** no supervisada, donde no se conocen a priori las clases de los objetos disponibles. Desde la perspectiva del aprendizaje no supervisado o de descubrimiento, la tarea se concreta en la obtención de la clasificación óptima, por partición y agrupamiento, de un conjunto dado de objetos a partir de los atributos que los describen. La calidad de la clasificación se mide por el grado de similitud que existe entre los objetos que quedan encuadrados en una misma clase y la disimilitud entre los objetos asignados a clases distintas. Las clases suelen identificarse por los centroides correspondientes. Con el criterio de la mínima *Suma de Cuadrados* hay que dividir  $n$  puntos del espacio euclídeo  $q$ -dimensional entre clases, y determinar los centroides de cada una de ellas de forma que se minimice la suma de los cuadrados de las distancias euclídeas entre cada punto y el centroide correspondiente. Diversas versiones de la VNS han sido comparadas con éxito en [48] con heurísticas clásicas para esta tarea. En [4] se usa la estructura VNS para abordar la versión difusa o borrosa de este problema (Fuzzy Clustering Problem) donde los puntos se asignan en diferentes grados a varias clases. Desde la perspectiva de la clasificación supervisada la tarea se concreta en la obtención de una regla de clasificación o clasificador basada en la descripción de los objetos, cuando se dispone de la clasificación de algunos de ellos. En [32] se muestra como la VNS es capaz de proporcionar clasificadores, del tipo del vecino más próximo, mejores que los proporcionados por los métodos constructivos usuales (ver [79]).

La **planificación** de tareas [71] es uno de los problemas de optimización a los que se les ha

prestado más atención en IA. En los problemas de secuenciación las soluciones vienen dadas por permutaciones del conjunto tareas. Varios problemas de secuenciación en una máquina con diversas funciones objetivos son resueltos con VNS y VND en [5]. En [22] se consideran problemas con varias máquinas que pueden trabajar simultáneamente en algunas tareas, pero respetando un grafo de precedencia entre las tareas. En [29] y [56] se estudia la VNS en problemas donde además se consideran restricciones en los recursos. Los problemas de planificación de horarios de enfermeras en un hospital han sido resueltos en [10]. En [55] se aplica la VNS a otro problema de planificación.

## 4.2. Descubrimiento en Teoría de Grafos

La metaheurística VNS se ha mostrado muy útil en el descubrimiento asistido por ordenador en Teoría de Grafos. Una *invariante* de la teoría de grafos es una variable definida en una familia de grafos cuyo valor no depende de la numeración de sus vértices o aristas. Numerosos problemas de la teoría de grafos pueden verse como la optimización de invariantes dentro de una familia de grafos, posiblemente restringida. La VNS se ha utilizado para aproximar soluciones de tales problemas, llegando a refutar algunas conjeturas y establecer otras, algunas de las cuales han llegado a ser posteriormente probadas formalmente [16].

En [16] se muestra como se utilizan esquemas de la VNS y la VND dentro del sistema **Auto-GraphiX** (AGX) para la optimización de invariantes de teoría grafos. El descenso se realiza generando al azar, en primer lugar, un grafo  $G$  con un número dado de vértices y aristas. Se calcula el valor de la invariante para dicho grafo y se examina el efecto de este valor al hacer algunos cambios elementales en el grafo, como la eliminación o inserción de una arista, el desplazamiento de una arista o la sustitución de una arista por un camino de longitud 2 uniendo sus extremos a través de un tercer vértice no adyacente. Los entornos definidos de esta forma se ordenan de menor a mayor tamaño. Se determina el mejor movimiento en cada uno de ellos y si mejora el valor de la invariante en estudio se realiza el correspondiente cambio en  $G$ . Después de que se haya encontrado un mínimo local se

consideran los entornos definidos por la distancia de Hamming y con ellos se aplica la VNS. La distancia de Hamming entre dos grafos es el número de aristas distintas en los dos grafos, por lo que los movimientos considerados son los de cambiar unas aristas por otras.

El sistema AGX ha sido utilizado para encontrar un grafo verificando unas restricciones, para encontrar el valor óptimo (o cercano al óptimo) de una invariante sujeta a restricciones, para refutar una conjetura, para sugerir (o apretar) una conjetura y hasta para obtener una prueba. En [16] se usa AGX para refutar tres conjeturas del sistema Graffiti ([25], [23] [26]) y ajustar otras, y se llega a probar una de ellas. Otros resultados relevantes de la Teoría de Grafos Química se han obtenido con AGX en [14], [15], [40], [21], [2]

### 4.3. Problemas de Optimización

La VNS se ha aplicado con éxito a multitud de problemas de optimización combinatoria, entre los que se encuentran los problemas más relevantes de planificación logística: los problemas de empaquetado, de localización y de rutas. Además se han abordado problemas donde las soluciones son ciertos conjuntos de aristas, como los árboles, y algunos problemas de optimización continua.

Los **problemas de empaquetado** constituyen una clase de problemas importantes en contextos de logística y distribución. En [30] se usa la VNS básica para una de las versiones básicas: el problema de empaquetado unidimensional (*bin-packing problem*, BPP) en el que se tiene que empaquetar un conjunto de objetos de diferente tamaño en el menor número de cajas o segmentos de capacidad fija.

Los problemas de **localización**, muy relevantes en la planificación logística, y los de agrupamiento o *clustering* (dentro de la clasificación no supervisada) tienen características comunes que los hacen similares al ser abordados por búsquedas metaheurísticas. El problema de la  $p$ -mediana es el problema más extensamente estudiado en Teoría de Localización. Consiste en determinar  $p$  ubicaciones, entre un conjunto de  $m$  localizaciones posibles para un servicio, de forma que se minimice la suma de las distan-

cias de  $n$  usuarios a su punto de servicio más cercano. Este problema ha sido abordado con éxito en [41], [78] y [52] por medio de una VNS básica, una VNS reducida y una VNDS, respectivamente. En [31] y [19] se proponen y prueban diversas VNS paralelas para este problema. En [64] se aplica la VNS al problema del  $p$ -centro en el que hay que minimizar, en lugar de la suma, el máximo de las distancias de los usuarios a sus respectivos puntos de servicios más próximos. Diversas variantes de la VNS han sido aplicadas con éxito en [8] y [9] al problema múltiple de Weber que es la versión continua del problema de la  $p$ -mediana, en el que las  $p$  localizaciones pueden elegirse en todo el plano donde también están ubicados los usuarios. En [52] se muestra la aplicación de la RVNS y la VNDS para el problema simple de localización de plantas (*Simple Plant Location Problem*, SPLP) en el que se debe decidir el número  $p$  de localizaciones, minimizando la suma del coste total que implica la separación de los usuarios a los puntos de servicio, y el coste de la selección de cada ubicación para el servicio. El problema de asignación cuadrática (*Quadratic Assignment Problem*, QAP) es otro problema relevante que surge en localización y ha sido abordado con la VNS en [17].

Otro tipo de problemas de optimización combinatoria importante, sobre todo en el contexto de la planificación logística, son los **problemas de rutas**. Tanto las versiones clásicas del problema del vendedor o viajante de comercio (*Travelling Salesman Problem*, TSP), del problema de ruta de vehículo (*Vehicle Routing Problem*, VRP) y del problema de ruta de arcos (*Arc Routing Problem*, ARP), así como algunas de sus extensiones han sido abordadas con la VNS.

El problema del viajante de comercio (*Travelling Salesman Problem*, TSP) consiste en, dadas  $n$  ciudades con las distancias o costes entre ellas, encontrar una ruta de mínimo coste (es decir, una permutación de las ciudades que minimiza la suma de las  $n$  distancias entre ciudades adyacentes en la ruta). En [11], [43] y [63] se aplican procedimientos del tipo de la VNS con distinto tipo de movimiento para el TSP. En [33] se aplica una VNDS similar con dos esquemas de descomposición diferentes. En [76] y [67] se usa una VNS para resolver otra importante extensión del TSP denominada *problema del comprador* en el que, dada una partición del

conjunto de clientes, hay que visitar al menos uno de cada conjunto de la partición.

Un problema de ruta de vehículos (*Vehicle Routing Problem*, VRP) consiste en diseñar las rutas desde un depósito para visitar un conjunto dado de clientes con mínimo coste. Los clientes tienen demandas conocidas y los vehículos tienen capacidad limitada. En [6] y [17] se aplican la VNS y la VND en la resolución de la variante del problema en el que cada cliente puede ser visitado en cierto intervalo de tiempo que se denomina VRPTW (*Time Windows*). En [20] se explora el uso de una VND para un VRP con recogida.

En [74] se propone una combinación de la VNS y la TS que usa varias estructuras de entornos para el problema del ciclo mediano (*Median Cycle Problem*, MCP). En este problema se debe determinar la ruta de menor longitud que recorra parte de las ciudades con una cota superior para la suma de las distancias desde las ciudades no incluidas en el recorrido hasta la ciudad más cercana de la ruta. En [63] se aplica una VNS a una versión del problema del viajante de comercio con clientes de recogida y entrega de mercancías. La ruta tiene que recorrer todos los clientes de recogida antes que los de entrega, partiendo y llegando al depósito. En [35], [44] y [46] se aborda con una VNS de forma exitosa problemas de rutas de arcos en los que las rutas deben recorrer todas las aristas o arcos de un grafo o red.

En otros problemas con múltiples aplicaciones relevantes las soluciones son **árboles**, como en el problema de Steiner, donde hay que determinar el subárbol de menor longitud que conecta todos los nodos terminales dados en un grafo. Se han probado diversas versiones de la VND para estos problemas en [61] y [73]. Para otras versiones, donde el grado de cada vértice en el subgrafo solución está acotado, también ha sido probado con éxito la VNS en [72]. La VNS ha sido probada en [66] para el problema de encontrar el subárbol de menor longitud con exactamente  $k$  aristas. En [13] se aborda el problema de determinar el subárbol que minimiza la suma de las longitudes de las aristas incluidas en el grafo y el peso de los vértices no conectados. En [1] se aplica una VND al problema del árbol filogenético (*the phylogeny problem*).

Para otros muchos de los problemas de op-

timización definidos en grafos, las soluciones vienen dadas por conjuntos de aristas. Una forma usual de dotar al espacio de una métrica, que puede usarse para aplicar distintas versiones de la VNS, consiste en el tamaño de la diferencia simétrica. El problema del máximo subgrafo completo (*clique*) ha sido abordado con una VNS en [53]. La efectividad de la VNS ha sido probada en [28] para resolver problemas de corte máximo, donde el objetivo es dividir el grafo en dos partes de forma que la suma de los pesos de las aristas que los unen sea máximo. Este tipo de problemas tiene aplicaciones importantes en el diseño VLSI. La VNS ha sido aplicada a un problema de diseño del grafo del tendido de cables en [18] y en el diseño de un grafo para la distribución de petróleo en [7].

Dentro de los problemas de optimización continua se ha probado la VNS en el problema de programación bilineal en [3] (en particular para una aplicación al problema del refinado en la industria del petróleo) y en el problema de optimización minimax continuo del diseño de un código polifásico en [65] y [47].

## 5. VNS entre las metaheurísticas

Las heurísticas tradicionales en optimización realizaban búsquedas locales descendentes por lo que se bloquean con el primer óptimo local encontrado. Las metaheurísticas proporcionan métodos, determinísticos o estocásticos, para escaparse de los óptimos locales de mala calidad. Puesto que el valor de tales óptimos locales frecuentemente difiere considerablemente del valor del óptimo global, especialmente si hay muchos óptimos locales, el impacto práctico de las metaheurísticas ha sido inmenso.

En contraste con este éxito, la teoría de metaheurísticas va más retrasada. Frecuentemente se obtienen buenas heurísticas, con algo de inventiva y gran esfuerzo en el ajuste de numerosos parámetros, pero las razones de porqué funcionan tan bien como lo hacen son desconocidas. La situación es incluso peor para los híbridos; es difícil discernir si el mérito lo tiene una de las componentes o se está obteniendo beneficio de la interacción.

Tres ideas para comprender mejor de estas cuestiones para cualquier metaheurística de búsqueda, que han sido utilizadas con la VNS, son el estudio de la topografía de los óptimos locales y de las trayectorias seguidas por los procesos de búsqueda, y de las fases de mejora y deterioro de las soluciones.

El estudio de la topografía de los mínimos locales da lugar a una descripción de los *perfiles de montañas y valles* encontrados durante la búsqueda. Cuando se aplica la VNS, el descenso desde una solución  $x'$ , seleccionada al azar, puede volver al óptimo local  $x$  alrededor del que están centrados los entornos actuales, o a otro óptimo local  $x''$  cuyo valor puede ser mejor o no que el de  $x$ . Se ha estudiado en [39] la probabilidad de estos tres sucesos como una función de la distancia de  $x$  a  $x'$  al aplicarla al *problema de la máxima satisfactibilidad ponderada*. Conviene también observar si  $x''$  está más cerca de  $x$  que  $x'$  (lo que puede ser interpretado como que  $x''$  pertenece al mismo valle que  $x$ , con rugosidades locales en relieve) o no (lo que puede interpretarse como un indicio de que se ha encontrado un nuevo gran valle). El relieve suministra esta información en base a una determinada cantidad de descensos desde puntos en entornos sucesivamente más amplios. Los perfiles varían considerablemente con la calidad del óptimo local  $x$ . Cuando  $x$  es una mala solución es suficiente alejarse un poco para obtener, con alta probabilidad, un óptimo local mejor. Cuando el óptimo local  $x$  es bueno, o muy bueno, debe alejarse bastante más para encontrar un nuevo gran valle y, además, la probabilidad de encontrar una solución mejor que la actual es entonces baja. Esto ilustra una debilidad del esquema básico de la VNS que tiende a degenerar en un *arranque múltiple* cuando la distancia de  $x$  a  $x'$  se hace grande. El remedio es el proporcionado por el esquema de la VNS sesgada.

Para algunos problemas se dispone de instancias para las que se conoce el óptimo global. Es corriente analizar las búsquedas heurística determinando lo frecuente que se alcanza el óptimo global o el promedio de la diferencia relativa entre el valor óptimo del objetivo y el alcanzado por la búsqueda. Sin embargo, se obtiene mucha más información si se consideran las propias soluciones y no sólo su valoración. Una herramienta para poner en práctica esta idea ha sido desarrollada para aplicar una VNS para el problema del vendedor en [33]. Esta he-

rramienta presenta en pantalla la solución óptima para el caso de entrenamiento bajo estudio, la solución actual y la diferencia simétrica entre estas dos soluciones. Esto indica cuanta mejora queda por hacer y donde. Además, una rutina permite también la representación de la *diferencia* entre la solución actual en una iteración y en la siguiente. Finalmente, como las representaciones de las soluciones para problemas grandes pueden ser difíciles de leer, y muchos problemas, en particular los problemas euclídeos, permiten una descomposición natural, una rutina de *enfoque* permite la representación de la información mencionada anteriormente para subproblemas seleccionados; es decir, en alguna región del espacio en la que se traza la ruta.

Además de usar las herramientas de visualización anteriores para guiar la búsqueda, se puede evaluar el trabajo realizado paso a paso por una VNS u otra metaheurística. Se pueden estudiar variantes en cada uno de esos pasos y recopilar información detallada para sacar conclusiones que pueden no ser evidente en un rendimiento global de la heurística. Esto puede dar lugar al descubrimiento de fenómenos inesperados y, la profundización proporcionada por su explicación, puede a su vez dar lugar a principios para mejorar las heurísticas. En [42] se ha observado que seleccionando la primera mejora (es decir, el primer movimiento que reduce el valor de la función objetivo) se obtienen mejores resultados que seleccionando la mejor mejora (es decir, el movimiento que más reduce el valor de la función objetivo) empezando desde una solución elegida al azar. El estudio sistemático de este fenómeno y su explicación en podría dar lugar a muchas mejoras para una gran variedad de metaheurísticas.

Es conveniente realizar algunas reflexiones sobre las propiedades *deseables* de las metaheurísticas. Estas buenas cualidades son las que propician o garantizan el interés práctico y teórico de las propuestas. En [50] se analiza la medida en que la VNS en relación a otras metaheurísticas se ajustan a tales propiedades. La lista de tales propiedades incluye las siguientes:

- (i) *Simplicidad*: la metaheurística debe estar basada en un principio simple y claro.
- (ii) *Precisión*: los pasos de la metaheurística deben formularse en términos precisos.

- (iii) *Coherencia*: las operaciones del algoritmo heurístico deben derivarse naturalmente de los principios de la metaheurística.
- (iv) *Efectividad* o *Eficacia*: los procedimientos deben proporcionar soluciones óptimas o cercanas a la óptima para todos o al menos para la mayoría de los casos realistas.
- (v) *Eficiencia*: los algoritmos deben emplear un tiempo computacional moderado para proporcionar soluciones óptimas o cercanas a la óptima.
- (vi) *Robustez*: el rendimiento de las heurísticas debe ser consistente sobre una variedad de casos.
- (vii) *Amigable*: las heurísticas deben expresarse claramente, ser fáciles de entender y usar.
- (viii) *Innovación*: los principios de la metaheurística, y/o la eficiencia y efectividad de las heurísticas derivadas de ella, deben dar lugar a nuevos tipos de aplicación.

A continuación se examina brevemente como la VNS se ajusta a cada una de las propiedades y se comentan las ventajas que reporta.

La VNS está basada en un principio simple y relativamente no explorado: el cambio sistemático de la estructura de entornos durante la búsqueda. La *simplicidad* de la metaheurística contribuye a dotarla de la amplia aplicabilidad que se refleja en la variedad de las aplicaciones ya comentadas. Las reglas *precisas* en la descripción de la forma de efectuar tales cambios son cruciales. Todos los pasos de los esquemas básicos y extendidos de la VNS se traducen *coherentemente* de los principios en que se inspira.

Para validar la *efectividad* de una metaheurística, éstas deberían encontrar soluciones óptimas para la mayoría de los problemas de un banco de casos para el que se conozcan las soluciones, cuando se disponga de ellos. La VNS tiene probada *eficacia* en la resolución de problemas de varios bancos de prueba con resultados óptimos o muy cercanos a los óptimos, y con tiempo computacional moderado (o al menos razonable).

La metaheurística VNS se ha mostrado muy *eficiente* en muchos experimentos, ya que obtiene

tan buenos o mejores resultados que la mayoría de las metaheurísticas en muchos problemas y de una forma mucho más rápida. Además, la VNS se muestra *robusta* ya que ha probado su rendimiento en multitud de problemas, y no sólo con un ajuste fino de parámetros a algún conjunto de entrenamiento.

La *amigabilidad* de los sistemas basados en la VNS cuentan con la ventaja de que los principios básicos en que se basa son fáciles de aplicar, y muy fáciles de usar. Esta cualidad se traduce en que en las aplicaciones realizadas, el número de parámetros se mantienen a un mínimo o incluso están ausentes. El potencial de la VNS para la *innovación* ya ha sido ilustrado por varios tipos de nuevas investigaciones: análisis de movimientos y su selección, aplicación a la generación de columnas enfocada y estabilizada, y el programa AGX de descubrimiento científico asistido por ordenador.

La búsqueda por entorno variable es, por tanto, una metaheurística reciente con el merito de ajustarse a las cualidades de la lista anterior. Esto, junto con la frecuencia con la que las metaheurísticas incorporan las ideas en las que se basa la VNS para beneficiarse de sus efectos positivos, justifica un papel destacado de la VNS en el campo de las metaheurísticas.

Están claros algunos aspectos del trabajo que aún queda por hacer. Aunque la VNS y otras metaheurísticas se han mostrado claramente útiles para la resolución aproximada de muchos problemas difíciles, les cuesta resolver problemas muy grandes. El tamaño de los problemas abordados se limita en la práctica por las herramientas disponibles para resolverlos más que por la necesidad de los potenciales usuarios de estas herramientas. Cuando las metaheurísticas se aplican a instancias realmente grandes sus fortalezas y debilidades aparecen más claramente. La mejoras en este sentido se presentan altamente deseables, tanto para las metaheurísticas en general, como para la VNS en particular.

## 6. Conclusiones

En el campo de las metaheurísticas hay frecuentes oportunidades para nuevas ideas donde aplicar la intuición más que la deducción. Los

conceptos principales raramente se definen con precisión y todavía hay muy pocos teoremas significativos. Los intentos por organizarlo son numerosos, pero ninguna propuesta ha conseguido una aceptación general. Cada metaheurística tiene su propio punto de vista pueden explicar otras heurísticas en su propio vocabulario y absorber ideas de todo el campo (generalmente en forma de híbridos). Con algunos argumentos o la reutilización de términos en la descripción de unas metaheurísticas y otras, se puede interpretar que una de ellas es la otra definida de manera incompleta (porque no se especifica algún elemento importante o es descrito por alguna vaga metáfora), o como un caso particular (porque se restringe el tipo de herramienta aplicada a un tipo de problema). La peor consecuencia de esta situación es la proliferación de reclamaciones de paternidad de las ideas. Son más importantes los éxitos claros en problemas particulares de las metaheurísticas que las controversias. Para establecer el interés de las metaheurísticas se deben considerar globalmente las cualidades deseables. Las comparativas de eficiencia no deberían tener el papel tan dominante, algunas veces exclusivo, como el que tiene en muchos artículos. El propósito de la investigación debe ser la comprensión, no la competición. Otras cualidades de las heurísticas y las metaheurísticas distintas a la eficiencia pueden ser más importantes a la larga, particularmente, la simplicidad, precisión, coherencia y, sobre todo, la innovación.

## Referencias

- [1] A. Andreatta and C. Ribeiro. Heuristics for the phylogeny problem. *Journal of Heuristics*, 8:429–447, 2002.
- [2] M. Aouchiche, G. Caporossi, and P. Hansen. Variable neighborhood search for extremal graphs 8: Variations on graffiti 105. *Congressus Numerantium*, 148:129–144, 2001.
- [3] C. Audet, J. Brimberg, P. Hansen, and N. Mladenović. Pooling problem: alternate formulation and solution methods. *Les Cahiers du GERAD*, G-2000-23, 2000.
- [4] N. Belacel, P. Hansen, and N. Mladenović. Fuzzy J-means: a new heuristic for fuzzy clustering. *Pattern Recognition*, 35(10):2193–2200, 2002.
- [5] M. den Besten and T. Stützle. Neighborhoods revisited: Investigation into the effectiveness of variable neighborhood descent for scheduling. In *MIC'2001*, pages 545–549, Porto, 2001.
- [6] O. Bräysy. A reactive variable neighborhood search algorithm for the vehicle routing problem with time windows. *INFORMS Journal on Computing*, to appear, 2003.
- [7] J. Brimberg, P. Hansen, K.-W. Lih, N. Mladenović, and M. Breton. An oil pipeline design problem. *Les Cahiers du GERAD*, G-2000-73, 2000. to appear in *Operations Research*, 2003.
- [8] J. Brimberg, P. Hansen, N. Mladenović, and É. Taillard. Improvements and comparison of heuristics for solving the multisource weber problem. *Operations Research*, 48(3):444–460, 2000.
- [9] J. Brimberg and N. Mladenović. A variable neighborhood algorithm for solving the continuous location-allocation problem. *Studies in Locational Analysis*, 10:1–12, 1996.
- [10] E.K. Burke, P. De Causmaecker, S. Petrović, and G.V. Berghe. Variable neighborhood search for nurse rostering. In *MIC'2001*, pages 755–760, Porto, 2001.
- [11] E.K. Burke, P. Cowling, and R. Keuthen. Effective local and guided variable neighborhood search methods for the asymmetric travelling salesman problem. *Lecture Notes in Computer Science*, 2037:203–212, 2001.
- [12] L.M. de Campos and J.M. Puerta. Stochastic local search algorithms for linear belief networks: searching in the space of orderings. *Lecture Notes in AI*, 2143:228–239, 2001.
- [13] S. Canuto, M. Resende, and C. Ribeiro. Local search with perturbations for the prize-collecting Steiner tree problem in graphs. *Networks*, 38:50–58, 2001.
- [14] G. Caporossi, D. Cvetković, I. Gutman, and P. Hansen. Variable neighborhood search for extremal graphs 2: Finding graphs with extremal energy. *Journal of*

- Chemical Information and Computer Science*, 39:984–996, 1999.
- [15] G. Caporossi, I. Gutman, and P. Hansen. Variable neighborhood search for extremal graphs 4: Chemical trees with extremal connectivity index. *Computers and Chemistry*, 23:469–477, 1999.
- [16] G. Caporossi and P. Hansen. Variable neighborhood search for extremal graphs 1: The AutoGraphiX system. *Discrete Mathematics*, 212:29–44, 2000.
- [17] R. Cordone and R. W. Calvo. A heuristic for the vehicle routing problem with time windows. *Journal of Heuristics*, 7(2):107–129, 2001.
- [18] M.C. Costa, F.R. Monclar, and M. Zrikem. Variable neighborhood search for the optimization of cable layout problem. In *MIC'2001*, pages 749–753, Porto, 2001.
- [19] T. Crainic, M. Gendreau, P. Hansen, N. Hoeb, and N. Mladenović. Parallel variable neighborhood search for the  $p$ -median. In *MIC'2001*, pages 595–599, Porto, 2001.
- [20] J. Crispim and J. Brandao. Reactive tabu search and variable neighborhood descent applied to the vehicle routing problem with backhauls. In *MIC'2001*, pages 631–636, Porto, 2001.
- [21] D. Cvetković, S. Simić, G. Caporossi, and P. Hansen. Variable neighborhood search for extremal graphs 3: On the largest eigenvalue of color-constrained trees. *Linear and Multilinear Algebra*, 49(2):143–160, 2001.
- [22] T. Davidović, P. Hansen, and N. Mladenović. Variable neighborhood search for multiprocessor scheduling with communication delays. In *MIC'2001*, pages 737–741, Porto, 2001.
- [23] E. DeLaVina, Graffiti.pc, Graph Theory Notes of New York, XLII, 26-30, (2002).
- [24] Jun Gu Dingzhu Du and Panos Pardalos, editors. *Satisfiability Problem: Theory and Applications*, volume 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. 1997.
- [25] S. Fajtlowicz. On conjectures of graffiti. *Discrete Mathematics*, 72:113–118, 1988.
- [26] S. Fajtlowicz. Written on the wall, 2003 <http://www.math.uh.edu/~siemion/>.
- [27] T. Feo and M. Resende. Greedy randomized adaptive search. *Journal of Global Optimization*, 6:109–133, 1995.
- [28] P. Festa, P. Pardalos, M. Resende, and C. Ribeiro. GRASP and VNS for max-cut. In *MIC'2001*, pages 371–376, Porto, 2001.
- [29] K. Fleszar and K.S. Hindi. Solving the resource-constrained project scheduling problem by a variable neighborhood search. *European Journal of Operational Research*, to appear, 2003.
- [30] K. Fleszar and K.S. Hindi. New heuristics for one-dimensional bin-packing. *Computers & Operations Research*, 29(7):821–839, 2002.
- [31] F. García López, M.B. Melián Batista, J.A. Moreno Pérez, and J.M. Moreno Vega. The parallel variable neighborhood search for the  $p$ -median problem. *Journal of Heuristics*, 8, 377–390, 2001.
- [32] M. García-Torres, J. A. Moreno Pérez, and J. Marcos Moreno Vega. VNS para Clasificación Supervisada. MAEB 03, 343-352, 2003.
- [33] M. Gendreau, P. Hansen, M. Labbé, and N. Mladenović. Variable neighborhood search for the traveling salesman problem. (in preparation), 2001.
- [34] M. Gendreau, G. Pesant, and L.-M. Rousseau. Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of Heuristics*, 8(1):43–58, 2002.
- [35] G. Ghiani, A. Hertz, and G. Laporte. Recent algorithmic advances for arc routing problems. *Les Cahiers du GERAD*, G-2000-40, 2000.
- [36] F. Glover. Tabu search - Part I. *ORSA Journal of Computing*, 1:190–206, 1989.
- [37] F. Glover. Tabu search - Part II. *ORSA Journal of Computing*, 2:4–32, 1990.
- [38] F. Glover and M. Laguna. *Tabu search*. Kluwer, Boston, 1997.

- [39] P. Hansen, J. Jaumard, N. Mladenović, and A.D. Parreira. Variable neighbourhood search for maximum weight satisfiability problem. *Les Cahiers du GERAD*, G-2000-62, 2000.
- [40] P. Hansen and H. Mélot. Variable neighbourhood search for extremal graphs 9: Bounding the irregularity of a graph. *Les Cahiers du GERAD*, G-2002-39, 2002.
- [41] P. Hansen and N. Mladenović. Variable neighborhood search for the  $p$ -median. *Location Science*, 5:207–226, 1997.
- [42] P. Hansen and N. Mladenović. First improvement may be better than best improvement: An empirical study. *Les Cahiers du GERAD*, G-99-40, 1999.
- [43] P. Hansen and N. Mladenović. An introduction to variable neighborhood search. In S. Voss et al., editor, *Metaheuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 433–458. Kluwer, 1999.
- [44] P. Hansen and N. Mladenovic. A separable approximation dynamic programming algorithm for economic dispatch with transmission losses. *GERAD report*, G-2000-31, 2000.
- [45] P. Hansen and N. Mladenovic. Developments of variable neighborhood search. *Les Cahiers du GERAD*, G-2001-24, 2001.
- [46] P. Hansen and N. Mladenović. Developments of variable neighborhood search. In C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 415–440. Kluwer, 2001.
- [47] P. Hansen and N. Mladenović. Industrial applications of the variable neighborhood search metaheuristics. In G. Zaccour, editor, *Decisions and Control in Management Science*, pages 261–274. Kluwer, 2001.
- [48] P. Hansen and N. Mladenović. J-means: A new local search heuristic for minimum sum-of-squares clustering. *Pattern Recognition*, 34(2):405–413, 2001.
- [49] P. Hansen and N. Mladenovic. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.
- [50] P. Hansen and N. Mladenovic. Variable neighbourhood search. In Fred Glover and Gary A. Kochenberger, editors, *Handbook of Metaheuristics*, chapter 6. Kluwer, 2003.
- [51] P. Hansen and N. Mladenovic. Variable neighbourhood search. In Panos M. Pardalos and Mauricio G. C. Resende, editors, *Handbook of Applied Optimization*. 2002.
- [52] P. Hansen, N. Mladenović, and D. Pérez Brito. Variable neighborhood decomposition search. *Journal of Heuristics*, 7(4):335–350, 2001.
- [53] P. Hansen, N. Mladenović, and D. Urošević. Variable neighborhood search for the maximum clique. *Les Cahiers du GERAD*, G-2001-08, 2001.
- [54] J.A. Hartigan. *Clustering Algorithms*. John Wiley, New York, 1975.
- [55] K.S. Hindi, K. Fleszar, and C. Charalambous. An Effective Heuristic for the CLSP with Setup Times. *Journal of the Operational Research Society*, 54 (5), 490–498, 2003.
- [56] R. Kolish and S. Hartman. Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis. In *Project scheduling: recent models, algorithms and applications*. Kluwer, 1999.
- [57] V. Kovačević, M. Čangalović, M. Ašić, D. Dražić, and L. Ivanović. Tabu search methodology in global optimization. *Computers & Math. with Appl*, 37:125–133, 1999.
- [58] L. Lobjois, M. Lemaitre, and G. Verfaillie. Large Neighbourhood Search using Constraint Propagation and Greedy Reconstruction for Valued CSP resolution. In *14th European Conference on Artificial Intelligence (ECAI'2000)*, Berlin, August 2000, 2001. Workshop on Modelling and Solving Problems with Constraints.
- [59] S. Loudin and P. Boizumault. VNS/LDS + CP: A hybrid method for constraint optimization in anytime contexts. In *MIC'2001*, pages 761–765, Porto, 2001.
- [60] R. Martí. Multi-start methods. In Fred Glover and Gary A. Kochenberger, editors,

- Handbook of Metaheuristics*, chapter 12. Kluwer Academic, 2002.
- [61] S.L. Martins, M.G.C. Resende, C.C. Ribeiro, and P. Pardalos. A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy. *Journal of Global Optimization*, 17:267–283, 2000.
- [62] N. Mladenović. A variable neighborhood algorithm - a new metaheuristic for combinatorial optimization. In *Abstracts of Papers Presented at Optimization Days*,, page 112, 1995.
- [63] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24:1097–1100, 1997.
- [64] N. Mladenovic, M. Labbe, and P. Hansen. Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, 2003 (to appear).
- [65] N. Mladenović, J. Petrović, Kovačević-Vujčić. V., and M. Čangalović. Solving spread spectrum radar polyphase code design problem by tabu search and variable neighborhood search. *European Journal of Operational Research*, 2004 (to appear).
- [66] N. Mladenović and D. Urošević. Variable neighborhood search for the  $k$ -cardinality. In *MIC'2001*, pages 749–753, Porto, 2001.
- [67] L.S. Ochi, M.B. Silva, and L. Drummond. Metaheuristics based on GRASP and VNS for solving traveling purchaser problem. In *MIC'2001*, pages 489–494, Porto, 2001.
- [68] J. Pearl. *Probabilistic reasoning in intelligent systems*. Morgan and Kaufman, 1998.
- [69] G. Pesant and M. Gendreau. A view of local search in constraint programming, principles and practice of constraint programming,. *Lecture Notes in Computer Science*, 1118:353–366, 1996.
- [70] G. Pesant and M. Gendreau. A constraint programming framework for local search methods. *Journal of Heuristics*, 5:255–279, 1999.
- [71] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems (2nd Edition)*. Prentice Hall, 2001.
- [72] C.C. Ribeiro and M.C. Souza. Variable neighborhood descent for the degree-constrained minimum spanning tree problem. *Discrete Applied Mathematics*, 118:43–54, 2002.
- [73] C.C. Ribeiro, E. Uchoa, and R. Werneck. A hybrid GRASP with perturbations for the Steiner problem in graphs. *INFORMS Journal of Computing*, 14:228–246, 2002.
- [74] I. Rodríguez, J.M. Moreno Vega, and J.A. Moreno Pérez. Variable neighborhood tabu search and its application to the median cycle problem. *European Journal of Operations Research*, to appear, 2003.
- [75] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming (CP'98)*, pages 417–431. 1998.
- [76] M.B. Silva, L. Drummond, and L.S. Ochi. Variable neighborhood search for the traveling purchaser problem. In *27th Int. Conference on Computational Industrial Engineering*, 2000.
- [77] E. Taillard and S. Voss. POPMUSIC - partial optimization metaheuristic under special intensification conditions. In C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 613–630. Kluwer, 2001.
- [78] R. Whittaker. A fast algorithm for the greedy interchange for large-scale clustering and median location problems. *INFOR*, 21:95–108, 1983.
- [79] D. Randall Wilson and Tony R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38-3:257–286, 2000.