VNS

Variable Neighbourhood Search

Francisco Buitrago Pavón

Javier Santos Paniego

Índice

- Marco teórico
  - Introducción (Definición y conceptos)
  - Variantes
    - VND (Descendente)
    - RVNS (Reducido)
    - BVNS (Básico)
- Ejemplo (CWP)
- Conclusiones
- Bibliografía

Marco Teórico

Introducción y conceptos

- VNS se concibe como una metaheurística para resolver problemas de optimización combinatoria.
- Propuesta por P. Hansen y N. Mladenovic en 1995.
- VNS intenta evitar quedar atrapada en óptimos locales cambiando la ***estructura de la vecindad*** donde se realiza la búsqueda.

Marco Teórico

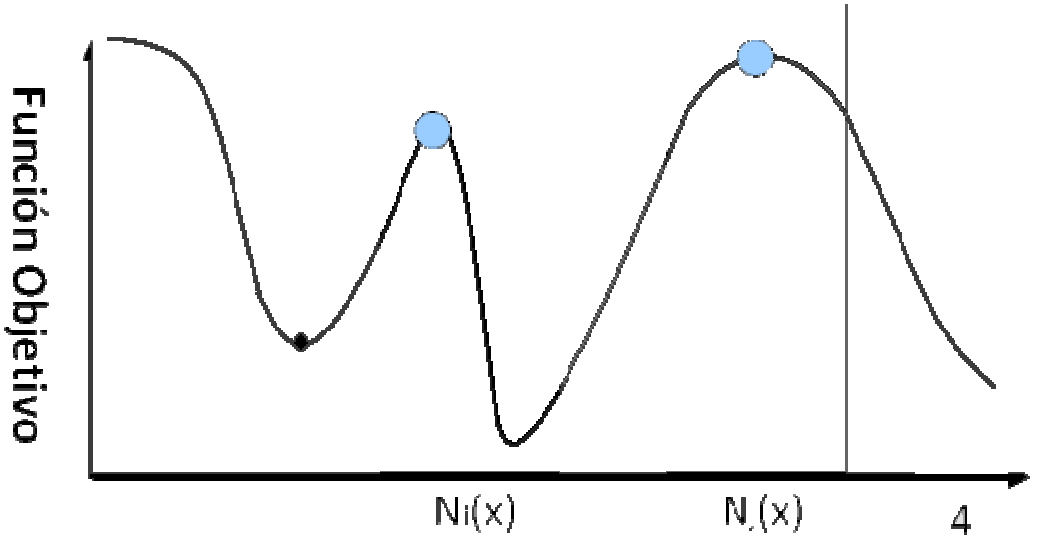
Introducción y conceptos

- Una **vecindad** es un conjunto de soluciones que se llaman vecinas a otra por haberle aplicado a ésta última un cambio.
- Una **estructura de vecindad**, dado un espacio de soluciones, se define como la función de cambio que genera un conjunto de vecindades $N_k(x)$ *con* $1 \leq k \leq k_{max}$.
- Cada N_i representa mediante su estructura de vecindad, un subconjunto del espacio de soluciones no necesariamente próximo a la solución óptima del problema.

Marco Teórico

Introducción y conceptos

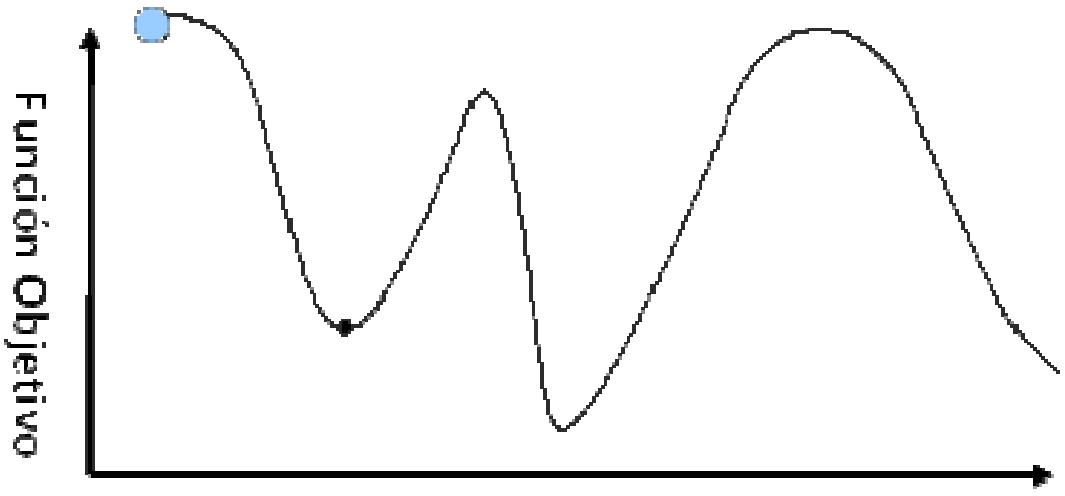
- VNS se basa en tres principios fundamentales:
  - Un óptimo local con respecto a una vecindad $N_i(x)$ no tiene por qué serlo con respecto a otra vecindad $N_j(x)$.



Marco Teórico

Introducción y conceptos

– Un óptimo global es un óptimo local con respecto a todas las posibles estructuras de vecindad



Marco Teórico

Introducción y conceptos

- Para muchos problemas, los óptimos locales, con respecto a una o varias estructuras de vecindad, están relativamente próximos

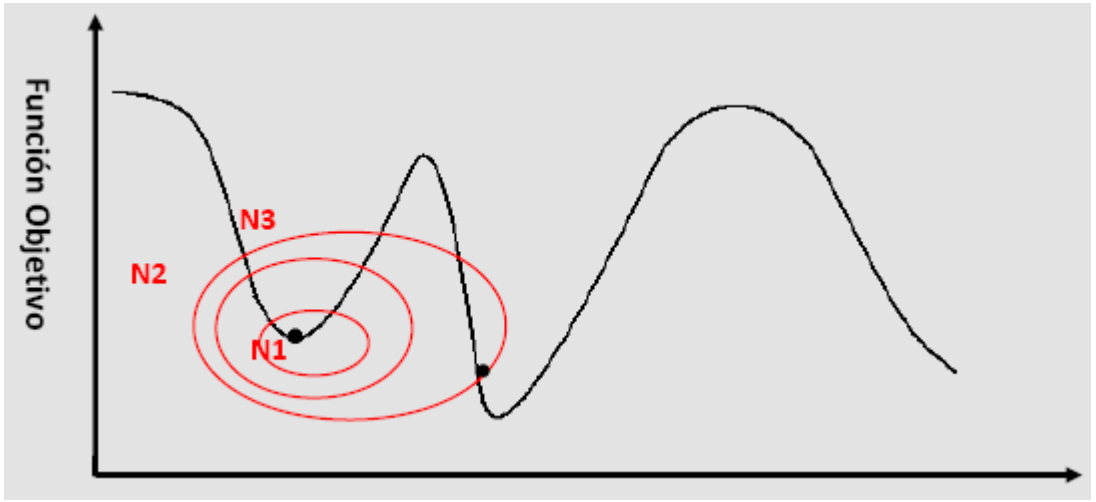
Marco Teórico

Variantes

- VND (*Variable Neighbourhood Descent)*
  - VND parte de una solución inicial aleatoria x_0 y mediante un procedimiento de búsqueda local podría mejorar la solución hasta obtener la solución x_1 (óptimo local), donde se quedaría atrapado.
  - Cuando esto ocurre, se modifica la estructura de vecindad y buscamos un nuevo óptimo local.

Marco Teórico

Variantes



Marco Teórico

Variantes

- VND (*Variable Neighbourhood Descent)*

```
var
    //Cada una de las N1 ,. . . , Nk representa una estructura de vecindad
    x0,x1: TipoSolucion;
    V : array [. . .] of TipoSolucion;
    k:integer;  //Vecindad
begin
    {x0} := GenerarSolucionInicial ();
    {V } := GenerarVecindad (N1);
    repeat
        k := 1;
        repeat
                    x1 := BusquedaLocal (x0, V ); //Empieza una búsqueda en Nk
                    if fObjetivo (x1) > fObjetivo (x0) then
                    //Suponiendo que es mejor el mayor valor de fObjetivo, aunque se trate de minimizar
                                //Nuevo mínimo local
                                x0 := x1;
                                k := 1;
                    else
                                k := k + 1;
        until k = kmax
    until  condicion_parada
end
```

Marco Teórico

Variantes

- La condición de parada viene determinada por:
  - Número de iteraciones máximas
  - Número de iteraciones sin mejora
  - Tiempo máximo

Marco Teórico

Variantes

- RVNS (*Reduced Variable Neighbourhood Search)*
  - En ciertos problemas, realizar una búsqueda local supone un alto coste computacional. Para esta clase de problemas suele aplicarse RVNS
  - La metodología a aplicar consiste en hallar una solución aleatoria por cada estructura de vecindades, en vez de un mínimo local.

Marco Teórico

Variantes

- RVNS (*Reduced Variable Neighbourhood Search)*

```
var
    //Cada una de las N1 ,. . . , Nk representa una estructura de vecindad
    x0,x1: TipoSolucion;
    V : array [. . .] of TipoSolucion;
    k:integer;  //Vecindad
begin
    {x0} := GenerarSolucionInicial ();
    {V } := GenerarVecindad (N1);
    repeat
        k := 1;
        repeat
                    x1 := ElegirAleatoria (V ); //Empieza una búsqueda en Nk
                    if fObjetivo (x1) > fObjetivo (x0) then
                    //Suponiendo que es mejor el mayor valor de fObjetivo, aunque se trate de minimizar
                                x0 := x1;
                                k := 1;
                    else
                                k := k + 1;
        until k = kmax
    until  condicion_parada
end
```

Marco Teórico

Variantes

- BVNS (*Basic Variable Neighbourhood Search)*
  - Se trata de una mezcla de las metodologías de VND y RVNS
  - La idea es seleccionar un punto aleatorio en una cierta vecindad y luego mejorarlo aplicándole una búsqueda local.

Marco Teórico

Variantes

- BVNS (*Basic Variable Neighbourhood Search)*

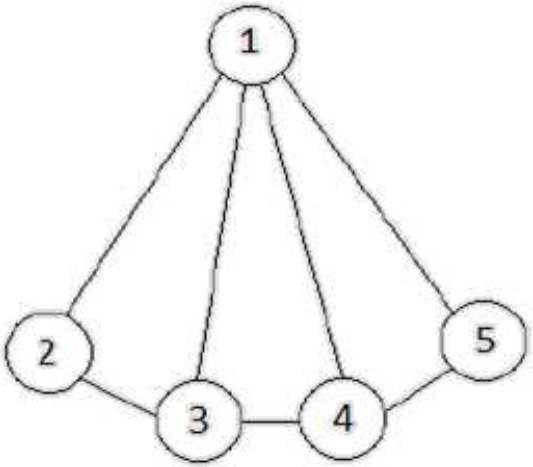
```
var
    //Cada una de las N1 ,. . . , Nk representa una estructura de vecindad
    x0,x1: TipoSolucion;
    V : array [. . .] of TipoSolucion;
    k:integer;  //Vecindad
begin
    {x0} := GenerarSolucionInicial ();
    {V } := GenerarVecindad (N1);
    repeat
        k := 1;
        repeat
                    x1 := ElegirAleatoria (V );
                    x1:= BusquedaLocal(x0,V);
                    if fObjetivo (x1) > fObjetivo (x0) then
                    //Suponiendo que es mejor el mayor valor de fObjetivo, aunque se trate de minimizar
                            x0 := x1;
                            k := 1;
                    else
                            k := k + 1;
        until k = kmax
    until  condicion_parada
end
```

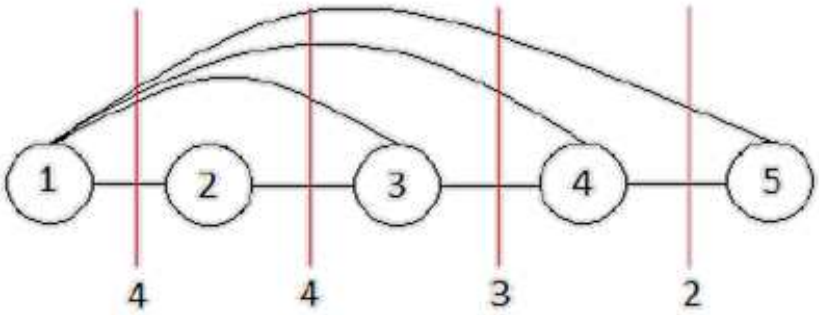
Ejemplo en cutwidth

- Dado un grafo, el problema del minimizado del cutwidth (CW) consiste en encontrar una ordenación lineal del grafo de tal forma que el máximo numero de aristas cortadas entre dos vértices consecutivos es mínima.
- Migración de redes

Ejemplo en cutwidth

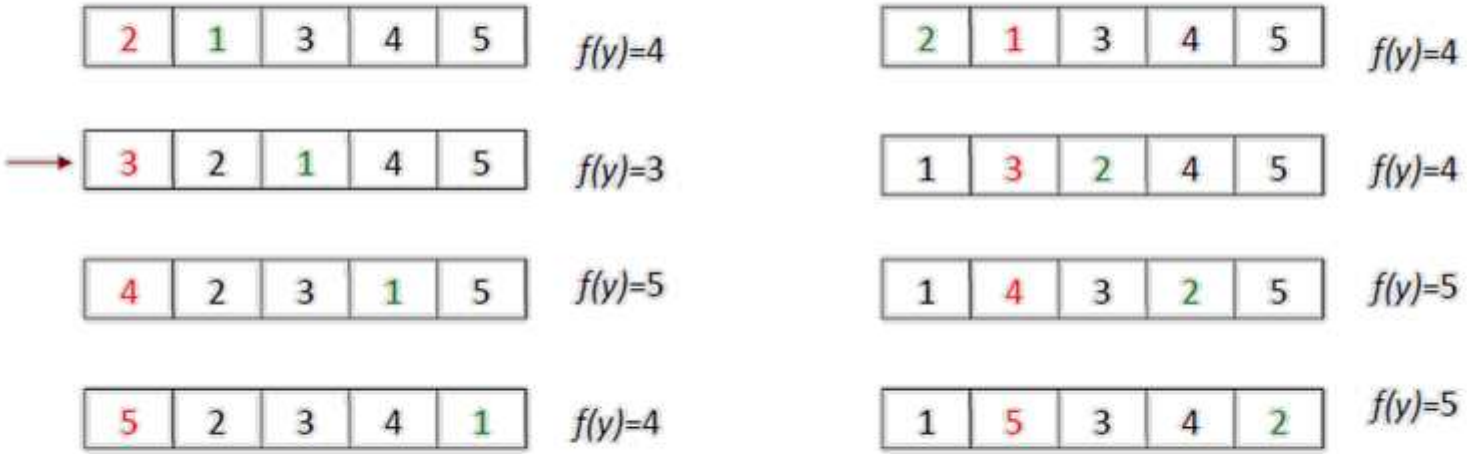
Solución Inicial → | 1 | 2 | 3 | 4 | 5 | $f(x)=4$ $Kmax = 3$





Ejemplo en cutwidth

- Ejemplo combinaciones



Ejemplo en cutwidth

- Después de realizar la búsqueda local se comparan el mejor resultado obtenido con el inicial.

if $f(y) >= f(x)$ **then**
    $x := y$;
    $k := 1$;

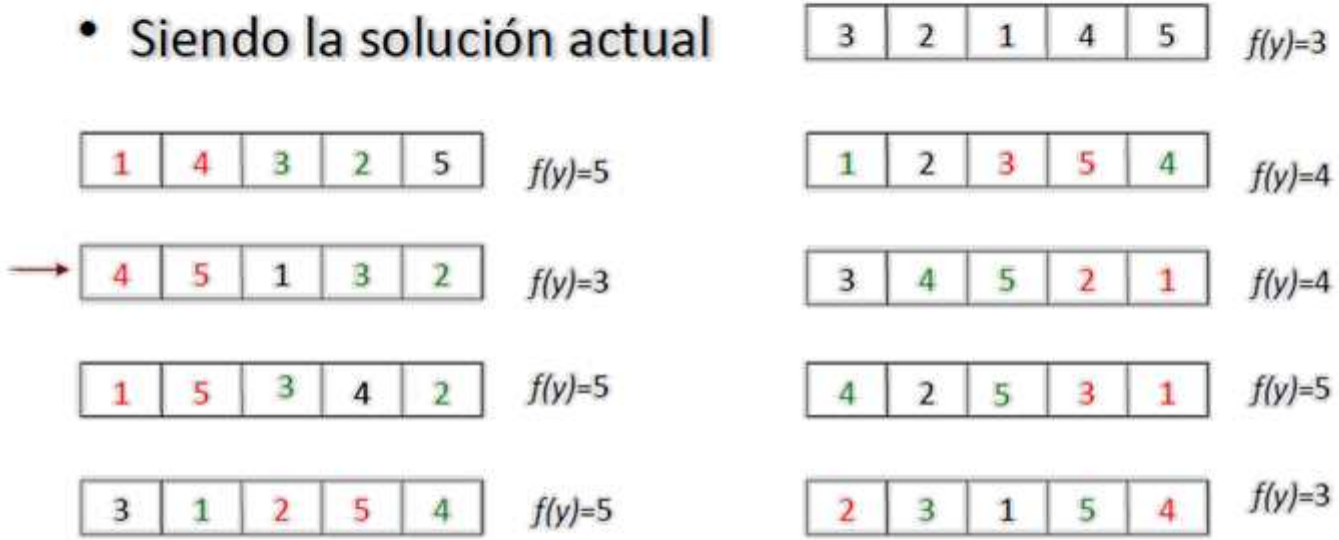
Ejemplo en cutwidth

- Para k=1 Solución actual 

$f(y)$=3
- De nuevo se realizan los intercambios , pero esta vez las soluciones encontradas no son mejores que la actual.
- Por tanto, se cambia la estructura de vecindad a k=2.

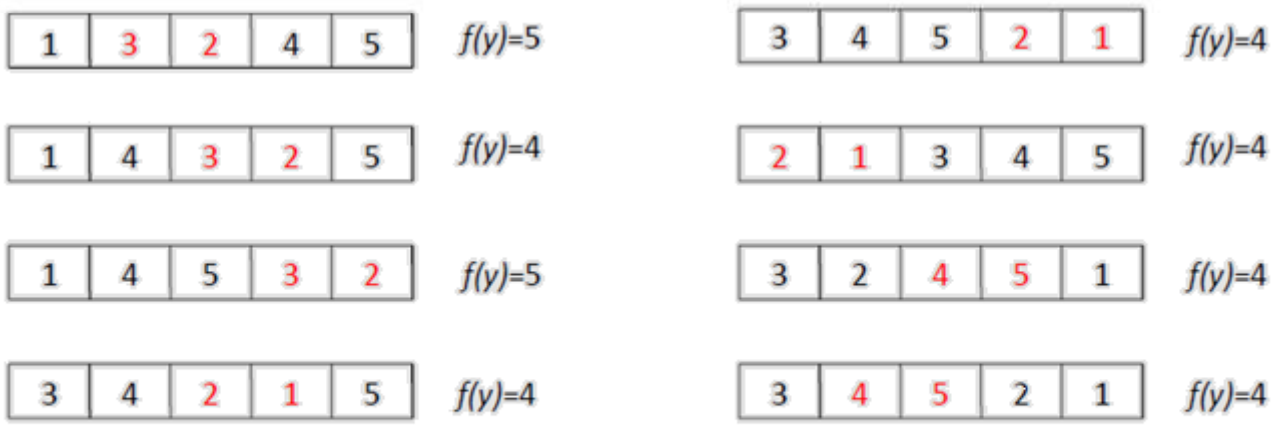
Ejemplo en cutwidth

- Para k=2 los intercambios se realizarán con pares de nodos.
- Siendo la solución actual



Ejemplo en cutwidth

- Ejemplo inserciones



Ejemplo en cutwidth

- Al realizar las inserciones encontramos una solución con $f(x)$=3. Por tanto se cambia el entorno de vecindad y k vuelve a tomar el valor inicial.

3	2	1	4	5

$f(y)$=3

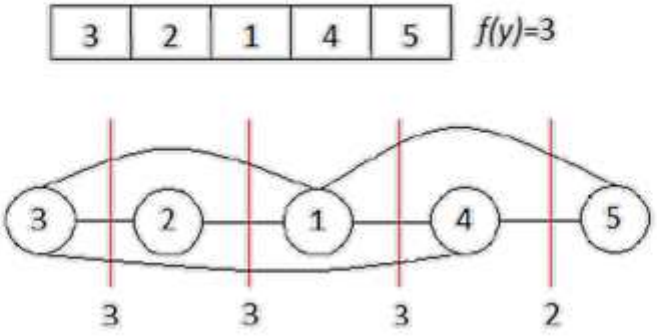
4	5	1	3	2

$f(y)$=3

if $f(y) >= f(x)$ **then**
    $x := y$;
    $k := 1$;

Ejemplo en cutwidth

- En este problema la ejecución terminaría cuando k en algún momento obtenga k_{max} o, en caso de que continuamente se encuentren nuevas soluciones, cuando se sobrepasen las condiciones de parada anteriormente nombradas.
- Ejemplo de solución:



Conclusiones

- VNS puede ser utilizado para resolver gran cantidad de problemas de carácter general.
- El cambio de estructuras de entorno nos permite tener un amplio conocimiento del espacio de soluciones.
- VNS ha tenido mucho éxito dada la posibilidad de hibridación con otras metaheurísticas (*Tabu Search, GRASP*).

Conclusiones

- Cumple las propiedades deseables de las metaheurísticas:
  - Simplicidad
  - Precisión
  - Coherencia
  - Eficacia
  - Eficiencia
  - Robustez

Bibliografía

- **El futuro de software comercial para problemas de rutas de vehículos** (*Kenneth Sörensen, Marc Sevaux y Patrick Schittekat)*
- **Variable Neighbourhood Search** (*Pierre Hansen, Nenad Mladenovic, José Andrés Moreno Pérez*)
- **Variable Neighbourhood Search** (*Nenad Mladenovic*)
- **Búsqueda en Vecindades Variables** (*Germán Ferrari, Federico Laca*)
- **Variable Neighbourhood Search** (*Miguel Ángel Moreno Álvarez, Guillermo Rey Mora*)