Variable Neighborhood Search

Christopher Expósito Izquierdo, J. Marcos Moreno Vega

cexposit@ull.es, jmmoreno@ull.es

Departamento de Ingeniería Informática y de Sistemas
Universidad de La Laguna

Variable neighborhood search

Observaciones

- **Observación 1:** Un óptimo local bajo una estructura de entorno no necesariamente es óptimo local bajo otra estructura de entorno.
- **Observación 2:** Un óptimo global es óptimo local bajo cualquier estructura de entorno.
- **Observación 3:** En muchos problemas de optimización los óptimos locales están relativamente cerca unos de otros.

Variable neighborhood search

Elementos

- Una solución inicial X.
- Un conjunto de estructuras de entorno $N_k(\cdot),\ k = 1, \ldots, k_{max}$.
- Un método (llamado de agitación), $SH(X, k, X')$, que genera aleatoriamente una solución $X' \in N_k(X)$.
- Un criterio de parada.

Variable neighborhood search

Estructuras de entorno. p-mediana discreta

$$N(X) = \{Y : Y \text{ se obtiene de } X \text{ al intercambiar un punto mediana por otro punto no mediana}\}$$

$$N_1(X) = N(X)$$
$$N_k(X) = \bigcup_{X' \in N_1(X)} N_{k-1}(X')$$

Variable neighborhood search

Estructuras de entorno. Ciclo mediano

$$
\begin{aligned}
N_1(X) &= \{Y : Y \text{ se obtiene de } X \text{ eliminando un vértice del ciclo}\} \\
N_2(X) &= \{Y : Y \text{ se obtiene de } X \text{ añadiendo un vértice al ciclo}\} \\
N_3(X) &= \{Y : Y \text{ se obtiene de } X \text{ al intercambiar un vértice del ciclo por otro no perteneciente a él}\}
\end{aligned}
$$

$$N(X) = N_1(X) \cup N_2(X) \cup N_3(X)$$

$$
\begin{aligned}
N'_1(X) &= N(X) \\
N'_k(X) &= \bigcup_{X' \in N'_1(X)} N_{k-1}(X')
\end{aligned}
$$

Variable neighborhood search

Algorithm 1: p-medianaSH(X, k, X')

$X' \leftarrow X$;
$r \leftarrow 0$;
repeat
  k movimientos de intercambio seleccionados aleatoriamente
  Seleccionar al azar (i, j) tal que $1 < i \leq p,\ p < j \leq n$;
  $X_{ij} \leftarrow X' \setminus \{v_i\} \cup \{v_j\}$;
  $X' \leftarrow X_{ij}$;
  $r \leftarrow r + 1$
until $r = k$;
return X';

Variable neighborhood search

Algorithm 2: cicloMedianoSH(X, k, X')

$X' \leftarrow X$;
$p \leftarrow$ número de vértices en el ciclo;
$r \leftarrow 0$;
repeat
  Seleccionar al azar (i, j) tal que $i, j \in \{2, \ldots, n\}$;
  if $(i < p$ **and** $j > p)$ **then** $X_{ij} \leftarrow X' \setminus \{v_i\} \cup \{v_j\}$; Intercambio
  if $(i > p$ **and** $j < p)$ **then** $X_{ij} \leftarrow X' \setminus \{v_j\} \cup \{v_i\}$; Intercambio
  if $(i > p$ **and** $j > p)$ **then** $X_{ij} \leftarrow X' \cup \{v_j\}$; Inserción
  if $(i < p$ **and** $j < p)$ **then** $X_{ij} \leftarrow X' \setminus \{v_i\}$; Eliminación
  $X' \leftarrow X_{ij}$;
  $r \leftarrow r + 1$
until $r = k$;
return X';

Variable neighborhood search

Cómo usar el conjunto de estructuras de entorno

- De forma determinística

  Búsqueda variable descendente

- De forma aleatoria

  Búsqueda por entornos variable reducida

- Alternando fases determinísticas con fases aleatorias

  Búsqueda por entornos variable básica

Variable neighborhood search

Búsqueda variable descendente (VND)

datos: $N_k(\cdot),\ k = 1, \ldots, k_{max}$, estructuras de entorno

entrada: X, solución de inicio

salida: X_{mej}, mejor solución obtenida

propósito: Escapar de los óptimos locales

propuesta: Usar de forma determinística las estructuras de entorno. Cuando se alcanza un óptimo local, se emplean las siguientes estructuras de entorno para buscar una solución mejor que dicho óptimo, volviendo al entorno inicial cuando esto sea cierto.

Variable neighborhood search

Algorithm 3: VND(X, X_{mej})

repeat
  $X_{mej} \leftarrow X$;
  $k \leftarrow 1$;
  repeat
    $X' \leftarrow \arg\min\{f(Y) : Y \in N_k(X)\}$;
    if $f(X') < f(X)$ **then**
      $X \leftarrow X'$;
      $k \leftarrow 1$;
    else
      $k \leftarrow k + 1$;
    end
  until $k = k_{max}$;
until $f(X) > f(X_{mej})$;
return X_{mej};

Variable neighborhood search

Búsqueda por entornos variable reducida (RVNS)

datos: $N_k(\cdot),\ k = 1, \ldots, k_{max}$, estructuras de entorno
Criterio de parada

entrada: X, solución de inicio

salida: X_{mej}, mejor solución obtenida

propósito: Escapar de los óptimos locales

propuesta: Usar de forma aleatoria las estructuras de entorno durante el proceso de búsqueda

Variable neighborhood search

Algorithm 4: RVNS(X, X_{mej})

$X_{mej} \leftarrow X$;
repeat
  $k \leftarrow 1$;
  repeat
    SH(X_{mej}, k, X');
    if $f(X') < f(X_{mej})$ **then**
      $X_{mej} \leftarrow X''$;
      $k \leftarrow 1$;
    else
      $k \leftarrow k + 1$;
    end
  until $k = k_{max}$;
until *Criterio de parada*;
return X_{mej};

Variable neighborhood search

Búsqueda por entornos variable básica (VNS)

datos: $N_k(\cdot),\ k = 1, \ldots, k_{max}$, estructuras de entorno
BL$(\cdot, \cdot)$, búsqueda local
Criterio de parada

entrada: X, solución de inicio

salida: X_{mej}, mejor solución obtenida

propósito: Realizar una búsqueda inteligente en la región factible alternando la intensificación en el entorno de la solución actual (Búsqueda local) con la exploración en sus alrededores (Agitación)

propuesta: Usar de forma determinística y aleatoria los entornos para escapar de los óptimos locales

Variable neighborhood search

Algorithm 5: VNS(X, X_{mej})

$X_{mej} \leftarrow X$;
repeat
  $k \leftarrow 1$;
  repeat
    SH(X_{mej}, k, X') ; — Agitación
    BL(X', X'') ; — Búsqueda local
    if $f(X'') < f(X_{mej})$ **then** — Mover o no
      $X_{mej} \leftarrow X''$;
      $k \leftarrow 1$;
    else
      $k \leftarrow k + 1$;
    end
  until $k = k_{max}$;
until *Criterio de parada*;
return X_{mej};

Variable neighborhood search

Búsqueda por entornos variable general (GVNS)

datos: $N_k(\cdot),\ k = 1, \ldots, k_{max}$, estructuras de entorno
$N_l(\cdot),\ l = 1, \ldots, l_{max}$, estructuras de entorno para la mejora (VND)
Criterio de parada

entrada: X, solución de inicio

salida: X_{mej}, mejor solución obtenida

propósito: Incrementar la eficiencia del VNS

propuesta: Sustituir la búsqueda local del VNS por una búsqueda variable descendente (VND)

Variable neighborhood search

Algorithm 6: GVNS(X, X_{mej})

$X_{mej} \leftarrow X$;
repeat
  $k \leftarrow 1$;
  repeat
    SH(X_{mej}, k, X');
    VND(X', X'');
    if $f(X'') < f(X_{mej})$ **then**
      $X_{mej} \leftarrow X''$;
      $k \leftarrow 1$;
    else
      $k \leftarrow k + 1$;
    end
  until $k = k_{max}$;
until *Criterio de parada*;
return X_{mej};

Skewed VNS (SVNS)

datos: $N_k(\cdot),\ k = 1, \ldots, k_{max}$, estructuras de entorno
$\rho(\cdot, \cdot)$, función de distancia entre soluciones
α, parámetro
BL$(\cdot, \cdot)$, búsqueda local
Criterio de parada

entrada: X, solución de inicio

salida: X_{mej}, mejor solución obtenida

propósito: Favorecer la exploración de subregiones de la región factible alejadas de la subregión actual

propuesta: Usar una función de distancia para aceptar soluciones de no mejora siempre que estas estén suficientemente alejadas de la solución actualmente considerada

Variable neighborhood search

Algorithm 7: SVNS(X, X_{mej})

$X_{mej} \leftarrow X$;
repeat
  $k \leftarrow 1$;
  repeat
    SH(X, k, X');
    BL(X', X'');
    if $f(X'') < f(X_{mej})$ **then** $X_{mej} \leftarrow X''$;
    if $f(X'') < f(X) + \alpha \cdot \rho(X, X'')$ **then**
      $X \leftarrow X''$;
      $k \leftarrow 1$;
    else
      $k \leftarrow k + 1$;
    end
  until $k = k_{max}$;
until *Criterio de parada*;
return X_{mej};

Variable neighborhood search

Búsqueda por entornos variable con descomposición (VNDS)

datos: $N_k(\cdot),\ k = 1, \ldots, k_{max}$, estructuras de entorno
BL$(\cdot, \cdot)$, búsqueda local
Criterio de parada

entrada: X, solución de inicio

salida: X_{mej}, mejor solución obtenida

propósito: Resolver eficiente y eficazmente problemas de gran tamaño

propuesta: Descomponer el espacio de búsqueda alrededor de la solución actual

Variable neighborhood search

Algorithm 8: VNDS(X, X_{mej})

$X_{mej} \leftarrow X$;
repeat
  $k \leftarrow 1$;
  repeat
    SH(X_{mej}, k, X'); $Y \leftarrow X' \setminus X$;
    VNS(Y, Y'); $X'' \leftarrow (X \setminus Y) \cup Y'$;
    BL(X'', X''');
    if $f(X''') < f(X_{mej})$ **then**
      $X_{mej} \leftarrow X'''$;
      $k \leftarrow 1$;
    else
      $k \leftarrow k + 1$;
    end
  until $k = k_{max}$;
until *Criterio de parada*;
return X_{mej};

Licencia