

Variable Neighborhood Search (VNS)

Moustapha Cisse

Universite de Montreal

6 avril 2010

Plan

- 1 Motivations
- 2 Algorithmes de base
- 3 extensions
 - skewed variable neighborhood search
 - variable neighborhood decomposition search
- 4 Applications

Le probleme d'optimisation.

On essaye de resoudre le probleme suivant :

$$\min\{f(x)|x \in X, X \subseteq S\}$$

- S : espace de solution (probleme combinatoire si S finie et tres grand)
- X : ensemble de solutions realisables
- f : fonction objective a valeur reelle

solution optimale

$$x^* \in X : f(x^*) \leq f(x), \forall x \in X$$

trouver une solution optimale, une preuve de son optimalite et en un temps fini

Resolution par recherche locale

Methode classique de resolution :

- definir une structure de voisinage N
- definir un operateur de transformation pour ce voisinage
- chercher iterativement une meilleure solution dans l'espace defini par le voisinage
- eventuellement intensification et diversification

La procedure mene a un optimum x^* :

$$x^* \in X : f(x^*) \leq f(x), \forall x \in N$$

Resolution par recherche locale

Methode classique de resolution :

- definir une structure de voisinage N
- definir un operateur de transformation pour ce voisinage
- chercher iterativement une meilleure solution dans l'espace defini par le voisinage
- eventuellement intensification et diversification

La procedure mene a un optimum x^* :

$$x^* \in X : f(x^*) \leq f(x), \forall x \in N$$

Probleme : la solution obtenue est optimale relativement au voisinage N (**pas X !**)

VNS

Optimisation par une modification systematique du voisinage et une phase de perturbation permettant de s'eloigner des (mauvais) minima locaux basee sur les faits suivants :

- 1 Un optimum relativement a un voisinage n'est pas necessairement optimal par rapport a un autre voisinage
- 2 Un optimum **global** est optimal relativement a toute structure de voisinage
- 3 Pour plusieurs problemes, les minima relativement a differents voisinages sont proches les uns des autres.

Optimisation par une modification systematique du voisinage et une phase de perturbation permettant de s'eloigner des (mauvais) minima locaux basee sur les faits suivants :

- 1 Un optimum relativement a un voisinage n'est pas necessairement optimal par rapport a un autre voisinage
- 2 Un optimum **global** est optimal relativement a toute structure de voisinage
- 3 Pour plusieurs problemes, les minima relativement a differents voisinages sont proches les uns des autres.

VNS

Optimisation par une modification systematique du voisinage et une phase de perturbation permettant de s'eloigner des (mauvais) minima locaux basee sur les faits suivants :

- 1 Un optimum relativement a un voisinage n'est pas necessairement optimal par rapport a un autre voisinage
- 2 Un optimum **global** est optimal relativement a toute structure de voisinage
- 3 Pour plusieurs problemes, les minima relativement a differents voisinages sont proches les uns des autres.

VNS

Optimisation par une modification systematique du voisinage et une phase de perturbation permettant de s'eloigner des (mauvais) minima locaux basee sur les faits suivants :

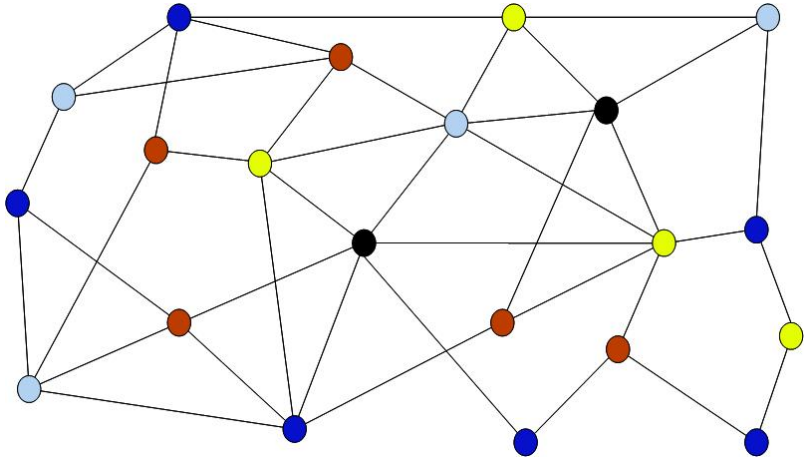
- 1 Un optimum relativement a un voisinage n'est pas necessairement optimal par rapport a un autre voisinage
- 2 Un optimum **global** est optimal relativement a toute structure de voisinage
- 3 Pour plusieurs problemes, les minima relativement a differents voisinages sont proches les uns des autres.

Optimisation par une modification systematique du voisinage et une phase de perturbation permettant de s'eloigner des (mauvais) minima locaux basee sur les faits suivants :

- 1 Un optimum relativement a un voisinage n'est pas necessairement optimal par rapport a un autre voisinage
- 2 Un optimum **global** est optimal relativement a toute structure de voisinage
- 3 Pour plusieurs problemes, les minima relativement a differents voisinages sont proches les uns des autres.

Solution : Resoudre le probleme en utilisant plusieurs voisinages $\{N_k\}$ de facon **deterministe**, **stochastique** ou **mixte**.

Structure des voisinages



changement de voisinage

L'idée centrale se trouve dans **le changement de voisinage** selon la procédure :

Algorithm 1: NeighborhoodChange(x, x', y)

```
1 if  $f(x') < f(x)$  then  
2   |  $x \leftarrow x'$  // modifier la solution courante;  
3   |  $k \leftarrow 1$  // voisinage initial;  
4 else  
5   |  $k \leftarrow k + 1$  // prochain voisinage;  
6 return  $x, k$ 
```

Structure generale du VNS

Algorithm 2: VNS(t_{max})

```

1 repeat
2    $k \leftarrow 1$  // k indice du voisinage;
3   repeat
4      $x \leftarrow \text{ChoixSolutionInitiale}(k)$  ;
5      $x' \leftarrow \text{TrouverMeilleurSolution}(x)$  ;
6      $x, k \leftarrow \text{ChangementVoisinage}(x, x', k)$  ;
7   until  $k = k_{max}$  ;
8    $t \leftarrow \text{cpuTime}()$ 
9 until  $t > t_{max}$  ;
10 return  $x$ 
  
```

Structure generale du VNS

Algorithm 3: VNS(t_{max})

```

1 repeat
2    $k \leftarrow 1$  // k indice du voisinage;
3   repeat
4      $x \leftarrow$  ChoixSolutionInitiale(k) ;
5      $x' \leftarrow$  TrouverMeilleurSolution(x) ;
6      $x, k \leftarrow$  ChangementVoisinage(x, x', k) ;
7   until  $k = k_{max}$  ;
8    $t \leftarrow$  cpuTime()
9 until  $t > t_{max}$  ;
10 return x
    
```

la nature **deterministe** ou **stochastique** de l'algorithme depend de l'implementation de (4) et (5)

Methode deterministe : VND

Variable Neighborhood Descent : effectuer changement de voisinage $N_k, k = 1, \dots, k_{max}$ de facon deterministe.

Algorithm 4: VND(x, k_{max}, t_{max})

```

1  $k \leftarrow 1$  repeat
2    $x' \leftarrow \arg \min_{y \in N_k(x)} f(y)$  // meilleur voisin ;
3    $x, k \leftarrow \text{NeighborhoodChange}(x, x', k)$  ;
4 until  $k = k_{max}$  ;
5 return  $x$ 
    
```

$$x^* \in X : f(x^*) \leq f(x), \forall k, \forall x \in N_k$$

La solution obtenue est probablement plus proche de l'optimum global.

Reduced Variable Neighborhood Search (RVNS)

RVNS : Methode plus adaptée aux problemes de grandes tailles dans lesquels une descente exhaustive dans chaque voisinage coute cher

- selection aleatoire (au lieu d'une descente) d'une solution dans le voisinage
- critere arret : temps d'execution maximal ou nombre d'iterations maximal entre 2 ameliorations
- comparable a une methode de monte carlo mais plus systematique
- le nombre de voisinage utilise est generalement 2 ou 3

Reduced Variable Neighborhood Search (RVNS)

RVNS : Methode plus adaptée aux problemes de grandes tailles dans lesquels une descente exhaustive dans chaque voisinage coute cher

- selection aleatoire (au lieu d'une descente) d'une solution dans le voisinage
- critere arret : temps d'execution maximal ou nombre d'iterations maximal entre 2 ameliorations
- comparable a une methode de monte carlo mais plus systematique
- le nombre de voisinage utilise est generalement 2 ou 3

Reduced Variable Neighborhood Search (RVNS)

RVNS : Methode plus adaptée aux problemes de grandes tailles dans lesquels une descente exhaustive dans chaque voisinage coute cher

- selection aleatoire (au lieu d'une descente) d'une solution dans le voisinage
- critere arret : temps d'execution maximal ou nombre d'iterations maximal entre 2 ameliorations
- comparable a une methode de monte carlo mais plus systematique
- le nombre de voisinage utilise est generalement 2 ou 3

Reduced Variable Neighborhood Search (RVNS)

RVNS : Methode plus adaptée aux problemes de grandes tailles dans lesquels une descente exhaustive dans chaque voisinage coute cher

- selection aleatoire (au lieu d'une descente) d'une solution dans le voisinage
- critere arret : temps d'execution maximal ou nombre d'iterations maximal entre 2 ameliorations
- comparable a une methode de monte carlo mais plus systematique
- le nombre de voisinage utilise est generalement 2 ou 3

Reduced Variable Neighborhood Search (RVNS)

RVNS : Methode plus adaptée aux problemes de grandes tailles dans lesquels une descente exhaustive dans chaque voisinage coute cher

- selection aleatoire (au lieu d'une descente) d'une solution dans le voisinage
- critere arret : temps d'execution maximal ou nombre d'iterations maximal entre 2 ameliorations
- comparable a une methode de monte carlo mais plus systematique
- le nombre de voisinage utilise est generalement 2 ou 3

Methodes stochastique : RVNS

Reduced Variable Neighborhood Search :

Algorithm 5: RVNS(x, k_{max})

```

1 repeat
2    $k \leftarrow 1$ ;
3   repeat
4      $x' \leftarrow Shake(x, k)$  // choix aleatoire ;
5      $x, k \leftarrow NeighborhoodChange(x, x', k)$  ;
6   until  $k = k_{max}$  ;
7    $t \leftarrow cpuTime()$ 
8 until  $t > t_{max}$  ;
9 return  $x$ 

```

Methodes mixtes

Ameliorer les methodes precedentes en combinant changement de voisinage deterministe et stochastique :

- 1 choisir une solution initial x
- 2 trouver une **direction de descente** a partir de x (dans un voisinage $N(x)$)
- 3 se deplacer vers le minimum dans $N(x)$ dans cette direction

Methodes mixtes

Ameliorer les methodes precedentes en combinant changement de voisinage deterministe et stochastique :

- 1 choisir une solution initial x
- 2 trouver une **direction de descente** a partir de x (dans un voisinage $N(x)$)
- 3 se deplacer vers le minimum dans $N(x)$ dans cette direction

Algorithm 6: BestImprovement(x)

```

1 repeat
2   |  $x' \leftarrow x$  ;
3   |  $x \leftarrow \arg \min_{y \in N_k(x)} f(y)$  ;
4 until  $f(x) \geq f(x')$  ;
5 return  $x$ 
    
```

Basic variable Neighborhood Search (BVNS)

Algorithm 7: BVNS(x, k_{max}, t_{max})

```

1 repeat
2    $k \leftarrow 1$ ;
3   repeat
4      $x' \leftarrow Shake(x, k)$  // choix aleatoire ;
5      $x'' \leftarrow BestImprovement(x')$  recherche deterministe;
6      $x, k \leftarrow NeighborhoodChange(x, x'', k)$  ;
7   until  $k = k_{max}$  ;
8    $t \leftarrow cpuTime()$ 
9 until  $t > t_{max}$  ;
10 return  $x$ 
  
```

variante : remplacer BestImprovement par FirstImprovement

General variable Neighborhood Search (GVNS)

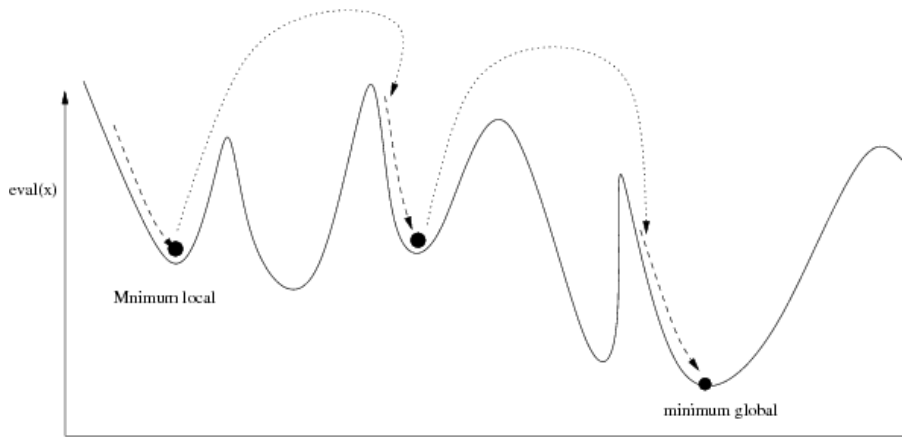
GVNS : remplacer la recherche locale par une descente sur un certain nombre de voisinages

Algorithm 8: $GVNS(x, k_{max}, l_{max}, t_{max})$

```

1 repeat
2    $k \leftarrow 1$ ;
3   repeat
4      $x' \leftarrow Shake(x, k)$  // choix aleatoire ;
5      $x'' \leftarrow VND(x', l_{max})$  //descente;
6      $x, k \leftarrow NeighborhoodChange(x, x'', k)$  ;
7   until  $k = k_{max}$  ;
8    $t \leftarrow cpuTime()$ 
9 until  $t > t_{max}$  ;
10 return  $x$ 
    
```

large valleys problem



solving the problem

Contraindre le déplacement vers des solutions éloignées en modifiant la procédure de changement de voisinage :

Algorithm 9: NeighborhoodChange(x, x', y)

```

1 if  $f(x') - \alpha\rho(x, x') < f(x)$  then
2   |  $x \leftarrow x'$  // modifier la solution courante;
3   |  $k \leftarrow 1$  // voisinage initial;
4 else
5   |  $k \leftarrow k + 1$  // prochain voisinage;
6 return  $x, k$ 
    
```

Skewed Variable Neighborhood Search (SVNS)

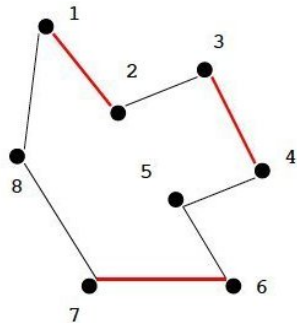
Algorithm 10: $SVNS(x, k_{max}, l_{max}, t_{max}, \alpha)$

```

1  repeat
2  |    $k \leftarrow 1$ ;
3  |   repeat
4  |   |    $x' \leftarrow Shake(x, k)$  // choix aleatoire ;
5  |   |    $x'' \leftarrow FirstImprove(x', l_{max})$  // descente;
6  |   |    $x, k \leftarrow NeighborhoodChange(x, x'', k)$  ;
7  |   until  $k = k_{max}$  ;
8  |    $x_{best} \leftarrow keepBest(x_{best}, x)$  ;
9  |    $X \leftarrow x_{best}$ ;
10 |    $t \leftarrow cpuTime()$ ;
11 until  $t > t_{max}$  ;
12 return  $x$ 
    
```

Principe du VNDS

- generer des sous problemes en fixant certaines composantes (garder k composantes libres)
- effectuer une recherche locale sur les k composantes libres



Variable Neighborhood Decomposition Search

Algorithm 11: $VDNS(x, k_{max}, l_{max}, t_{max}, t_d)$

```

1 repeat
2    $k \leftarrow 1$ ;
3   repeat
4      $x' \leftarrow Shake(x, k)$  // choix aleatoire ;
5      $t \leftarrow FreeComponents(x, x')$ ;
6      $t^* \leftarrow FirstImprovement(t)$ ;
7      $x'' \leftarrow InjecComponents(t^*, s')$  //mise a jour de s';
8      $x, k \leftarrow NeighborhoodChange(x, x'', k)$  ;
9   until  $k = k_{max}$  ;
10   $t \leftarrow cpuTime()$ 
11 until  $t > t_{max}$  ;
12 return  $x$ 
    
```

Decisions de conception

L'application du VNS a un probleme necessite de choisir :

- 1 nombre et type de voisinage a utiliser
- 2 ordre d'exploration des voisinages pendant la recherche
- 3 strategie de changement de voisinage
- 4 methode de recherche locale
- 5 critere d'arret

Decisions de conception

L'application du VNS a un probleme necessite de choisir :

- 1 nombre et type de voisinage a utiliser
- 2 ordre d'exploration des voisinages pendant la recherche
- 3 strategie de changement de voisinage
- 4 methode de recherche locale
- 5 critere d'arret

Decisions de conception

L'application du VNS a un probleme necessite de choisir :

- 1 nombre et type de voisinage a utiliser
- 2 ordre d'exploration des voisinages pendant la recherche
- 3 strategie de changement de voisinage
- 4 methode de recherche locale
- 5 critere d'arret

Decisions de conception

L'application du VNS a un probleme necessite de choisir :

- 1 nombre et type de voisinage a utiliser
- 2 ordre d'exploration des voisinages pendant la recherche
- 3 strategie de changement de voisinage
- 4 methode de recherche locale
- 5 critere d'arret

Decisions de conception

L'application du VNS a un probleme necessite de choisir :

- 1 nombre et type de voisinage a utiliser
- 2 ordre d'exploration des voisinages pendant la recherche
- 3 strategie de changement de voisinage
- 4 methode de recherche locale
- 5 critere d'arret

Decisions de conception

L'application du VNS a un probleme necessite de choisir :

- 1 nombre et type de voisinage a utiliser
- 2 ordre d'exploration des voisinages pendant la recherche
- 3 strategie de changement de voisinage
- 4 methode de recherche locale
- 5 critere d'arret

Exemple : TSP

Application au Trading Salesman Problem avec les choix :

- 1 definition de voisinage k -opt, i.e, $N_k(x)$ est l'ensemble des solutions ayant k arcs differents de x
- 2 la methode de recherche locale utilisee est 2-opt
- 3 changement de voisinage dans l'ordre des indices k

Exemple : TSP

Application au Trading Salesman Problem avec les choix :

- 1 définition de voisinage k -opt, i.e, $N_k(x)$ est l'ensemble des solutions ayant k arcs différents de x
- 2 la methode de recherche locale utilisee est 2-opt
- 3 changement de voisinage dans l'ordre des indices k

Exemple : TSP

Application au Trading Salesman Problem avec les choix :

- 1 définition de voisinage k -opt, i.e, $N_k(x)$ est l'ensemble des solutions ayant k arcs différents de x
- 2 la méthode de recherche locale utilisée est 2-opt
- 3 changement de voisinage dans l'ordre des indices k

Exemple : TSP

Application au Trading Salesman Problem avec les choix :

- 1 définition de voisinage k -opt, i.e, $N_k(x)$ est l'ensemble des solutions ayant k arcs différents de x
- 2 la méthode de recherche locale utilisée est 2-opt
- 3 changement de voisinage dans l'ordre des indices k

Exemple : TSP

Application au Trading Salesman Problem avec les choix :

- 1 définition de voisinage k -opt, i.e, $N_k(x)$ est l'ensemble des solutions ayant k arcs différents de x
- 2 la méthode de recherche locale utilisée est 2-opt
- 3 changement de voisinage dans l'ordre des indices k

Remarque : de meilleures performances peuvent être obtenues en utilisant GENIUS

Table 1

TSP: Average results for random Euclidean problems over 100 trials for $n = 100, \dots, 500$

n	Best value found			% Improvement over 2-opt	
	2-opt	VNS-1	VNS-2	VNS-1	VNS-2
100	825.69	817.55	811.95	0.99	1.66
200	1156.98	1143.19	1132.63	1.19	2.10
300	1409.24	1398.16	1376.76	0.79	2.30
400	1623.60	1602.59	1577.42	1.29	2.84
500	1812.08	1794.59	1756.26	0.96	3.07
600	1991.56	1959.76	1925.51	0.97	3.32
700	2134.86	2120.59	2089.33	0.67	2.13
800	2279.18	2242.11	2190.83	1.63	3.88
900	2547.43	2399.52	2342.01	5.81	8.06
1000	2918.10	2555.56	2483.95	12.42	14.88
Average	1869.87	1803.36	1768.67	2.73	4.43

^a Computing times in seconds CPU on a SUN SPARC 10, 135.5 Mips

1

Annexe I



Variable Neighborhood Search P. Hansen et al.
Handbook of Metaheuristics.
Springer New York, 2003.