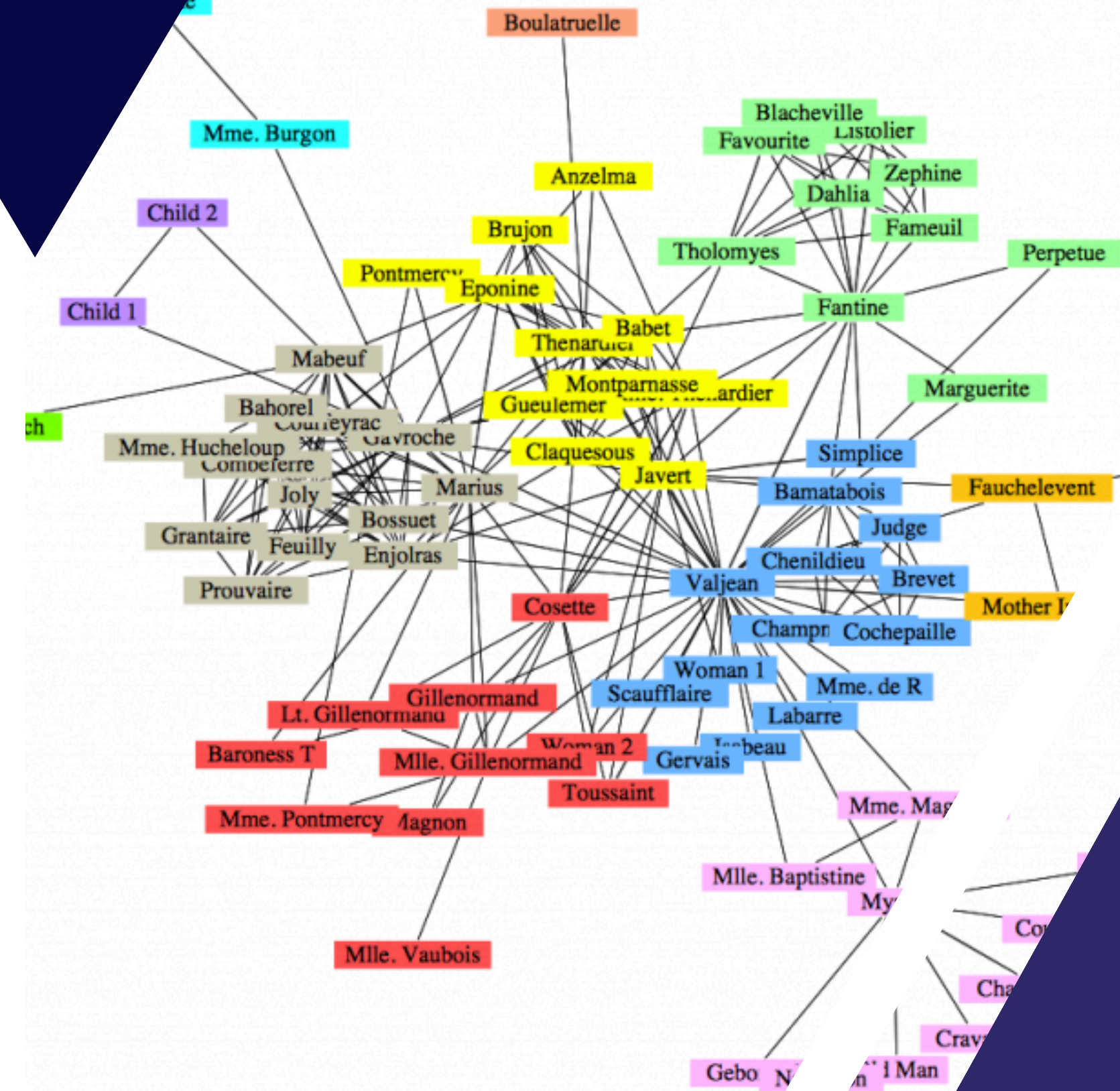


Introduction

La recherche à voisinages variables (RVV), ou **Variable Neighborhood Search** (VNS) en anglais, est une métaheuristique dont l'invention est attribuée à Nenad Mladenović et Pierre Hansen. Elle a été développée au GERAD (Groupe d'Études et de Recherche en Analyse des Décisions) à partir de 1997, situé à Montréal.




of interactions between major characters in the novel *Les Misérables*. The shortest-path version of our algorithm is $Q = 0$ and corre



Fonctionnement de l'algorithme

Comme d'autres métaheuristiques, la recherche à voisinages variables repose sur deux méthodes complémentaires : d'une part, **la recherche locale** et ses extensions, qui visent à améliorer la solution courante, et d'autre part, **les perturbations**, qui permettent d'élargir l'espace des solutions explorées.



Recherche locale

La recherche locale (RL), utilisée par un grand nombre de métaheuristiques, consiste en des améliorations successives de la solution courante par l'entremise d'une transformation élémentaire, jusqu'à ce qu'aucune amélioration ne soit possible. La solution ainsi trouvée est appelée optimum local par rapport à la transformation utilisée.

Algorithme 1: Algorithme RechercheLocale

Input : S

Input : \mathcal{N}

Posons $ameliore \leftarrow vrai$

tant que $ameliore = vrai$ **faire**

$ameliore \leftarrow faux$

pour tout $S' \in \mathcal{N}(S)$ **faire**

si S' meilleure que S **alors**

$S \leftarrow S'$

$ameliore = vrai.$

retourner S

Modifier les voisinages

Pour tirer avantage des diverses transformations qui peuvent exister pour un problème donné, et de leurs particularités, il est possible d'adapter la recherche locale afin de ne pas utiliser seulement une transformation, mais une séquence de transformations différentes. C'est ainsi qu'est construite la descente à voisinages variables (DVV).

Modifier les voisinages

De même que la recherche locale explore les diverses manières d'appliquer la transformation choisie pour améliorer la solution courante, la descente à voisinages variables (DVV) explore une série de voisinages successivement. Considérons alors une liste $N_t(S)$ pour $t=1,\dots,T$ où T est le nombre de transformations considérées.

En utilisant successivement tous les voisinages de la liste pour effectuer des recherches locales, la descente à voisinages variables ne s'arrête que lorsqu'aucun d'eux ne permet d'améliorer la solution. Un optimum local pour chacun des voisinages considérés est alors identifié.

Mais cela s'effectuera en respectant plusieurs contraintes.

Modifier les voisinages

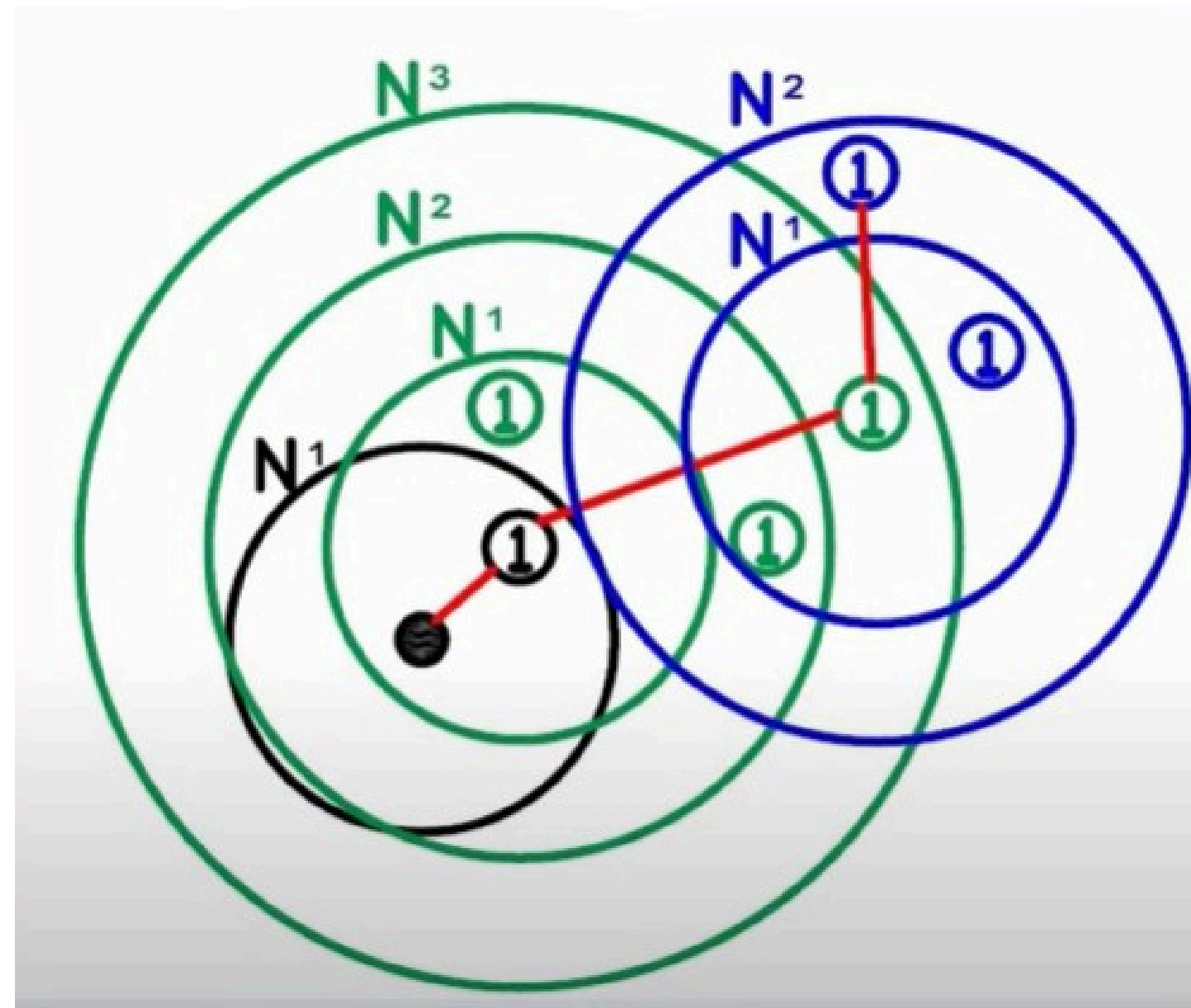


Fig : contraintes de voisinages

Modifier les voisinages

Algorithme 2: Algorithme DVV

Input : S

Input : $\mathcal{N}^t, \quad t = 1 \dots T$

Posons $ameliore \leftarrow vrai$

tant que $ameliore = vrai$ **faire**

$ameliore \leftarrow faux$

pour tout $t = 1 \dots T$ **faire**

$S' \leftarrow RechercheLocale(S, \mathcal{N}^t)$

si S' meilleure que S **alors**

$S \leftarrow S'$

$ameliore \leftarrow vrai.$

retourner S




Perturbations

Afin de réduire cet effort de calcul, plutôt que d'utiliser des solutions aléatoires comme point de départ des recherches locales, une autre approche consiste à modifier modérément la meilleure solution connue, ce que nous appelons une perturbation.

Une méthode utilisant des recherches multiples à partir de perturbations se concentrera sur des solutions proches de la meilleure solution connue et profitera des caractéristiques de cette dernière.

C'est pour cette raison que la recherche à voisinages variables ne procède pas par des recherches locales à partir de solutions aléatoires, mais à partir de solutions proches de la meilleure solution connue.



Perturbations

Algorithme 4: Algorithme PERTURBE

Input : S

Input : k

Input : \mathcal{N}

répéter k **fois**

 | Choisir aléatoirement $S' \in \mathcal{N}(S)$,
 | poser $S \leftarrow S'$.

retourner S

La recherche à voisinages variables généralise

Algorithme 6: Algorithme RVVG

Input : S

Poser $S^* \leftarrow S$ la meilleure solution connue.

Poser $k \leftarrow 1$

Définir k_{max}

répéter

$S \leftarrow PERTURBE(S^*, k),$

$S' \leftarrow DVV(S).$

si S' meilleure que S^* **alors**

$S^* \leftarrow S',$

$k = 1.$

sinon

$k \leftarrow k + 1.$

si $k > k_{max}$ **alors**

 poser $k \leftarrow 1.$

jusqu'à critère d'arrêt;

retourner $S^*.$
