CHAPITRE 4

MÉTHODES CONSTRUCTIVES

Sommaire

1 Principe des heuristiques constructives 25
2 Méthodes gloutonnes 26
3 Méthodes gloutonne aléatoires 26
4 Heuristique de plus proche voisin 27

Introduction

Ce chapitre définit et présente les méthodes constructives. Ces méthodes intuitives et simples à implémenter sont des heuristiques spécifiques qui ne peuvent s'appliquer que sur un nombre de problèmes de spécificité commune. (Ouaarab, 2015),(Sabba, 2020)(Bolduc, 2003).

1 Principe des heuristiques constructives

L'idée de base des méthodes constructives est de produire une solution, en partant d'une combinaison vide, par l'ajout itératif des composants à la sous-solution existante (solution partielle à une itération donnée). Les choix partiels sont définitifs.

Les méthodes constructives sont largement utilisées comme des auxiliaires dans d'autres méthodes. Elles sont introduites généralement pour calculer des solutions initiales, ou à l'intérieur d'autres procédures. L'algorithme 1 illustre le fonctionnement d'une méthode constructive.

Algorithme 1 : Algorithme général d'une méthode constructive

Result : S Solution complète
début
  $S \leftarrow \phi$;
  tant que *(S n'est pas une solution complète* **faire**
    Choisir un élément de solution e;
    $S \leftarrow S + e$;
  fin
fin

TABLE 4.1 – Exemple d'un problème de sac-à-dos bonaire

Objet	p_i	w_i
1	15	5
2	20	4
3	2	1
4	10	3
5	5	2

Ces heuristiques ne cherchent pas forcément une bonne solution, mais juste une solution de qualité modérée dans un temps de calcul réduit. La qualité de la combinaison construite dépend de l'heuristique choisie.

2 Méthodes gloutonnes

Les méthodes gloutonnes (greedy en anglais) sont les méthodes de construction les plus connues. A partir d'une configuration initiale, elles sélectionnent le meilleur élément de la solution à chaque itération en espérant que la solution finale soit optimale. Ces choix (décisions) se font selon des heuristiques spécifiques.

Exemple : Nous avons un sac-à-dos de capacité maximale $c = 10kg$. On veut maximiser le profit sachant qu'on dispose de 5 objets dont le poids et le profit sont présentés dans le tableau 4.1.

Une heuristique gloutonne qui peut résoudre ce problème serait d'ajouter en premier les objets ayant le meilleur rapport valeur/poids. La solution globale sera alors : $S = 2, 3, 4, 5$. Si on modélise le problème en binaire la solution sera : $S = \{0, 1, 1, 1, 1\}$.

3 Méthodes gloutonne aléatoires

Les méthodes gloutonnes sont des algorithmes très rapides. Cependant la résolution d'un problème d'optimisation combinatoire donné par ce type de méthode retourne toujours la

même solution. Cette solution peut passer de coté par la solution optimale si l'heuristique choisie n'est pas adaptée.

Pour diversifier la recherche, on ajoute un peu d'aléatoire. Un algorithme glouton aléatoire ne choisit pas le meilleur élément mais un élément e parmi **les meilleurs candidats**. L'exécution de l'algorithme, plusieurs fois, peut retourner une meilleure solution.

Exemple : Pour un problème de voyageur de commerce, on applique un algorithme glouton aléatoire de façon à choisir une ville dans un rayon spécifique (disant un rayon de 10 km).

4 Heuristique de plus proche voisin

Cette heuristique gloutonne est spécifique aux problèmes de cheminements (voyageur de commerce, chemin le plus court, arbre minimal, etc.). Elle consiste à choisir un sommet de départ et visiter son voisin le plus proche, qui n'a pas encore été visité, jusqu'à l'obtention d'une solution complète. L'algorithme 2 résume le fonctionnement de l'heuristique du plus proche voisin.

La solution obtenue par cette heuristique ne garantit pas un résultat optimal. Comme toute heuristique constructive, on pourrait améliorer la solution en exécutant n fois l'algorithme du plus proche voisin en changeant à chaque fois le point de départ.

Algorithme 2 : Algorithme de l'heuristique du plus proche voisin

Result : S Solution complète
Données : e, v_0, v : sommet
début
    $S \leftarrow \phi$;
    Initialisation du premier sommet v_0 ;
    $v \leftarrow v_0$;
    tant que *(S n'est pas une solution complète* **faire**
        Choisir le sommet e la plus proche de v ;
        si *($e == v_0$) et (le chemin est incomplet)* **alors**
            Éliminer ce voisin et chercher un autre ;
        sinon
            On élimine le voisinage de e ;
            On élimine e du voisinage des autres sommets ;
            $S \leftarrow S + e$;
            $v \leftarrow e$;
        fin
    fin
    Relier le dernier sommet v au premier sommet v_0 ;
fin