Résolution du problème de Sac à dos 0-1

Rapport de projet

Kévin Barreau

23 octobre 2015

Résumé

L'objectif du projet est d'implémenter différentes méthodes de résolution du problème de sac à dos binaire, afin des les comparer. Il s'agit des méthodes du Branch And Bound, de Programmation Dynamique (sous plusieurs implémentations), de Modèle de Flot dans un graphe, ainsi que l'utilisation du solveur Cplex. Le langage de programmation pour atteindre ce but est libre, mais sa rapidité est essentielle pour la résolution. J'ai par conséquent choisi de réaliser le projet en Java, au détriment de C++. Les résultats sont présentés en prenant en compte les comportements des algorithmes suivant les types d'instance (nombre d'objets, capacité maximale du sac, corrélation des objets). L'ensemble du projet peut se retrouver à l'adresse suivante : `https://github.com/LuminusDev/Knapsack/`.

Sommaire

1 Présentation du projet

1.1 Description du problème

Explication du problème de sac à dos, puis sac à dos binaire.

1.2 Méthodes de résolution

Pour résoudre ce problème, plusieurs algorithmes ont été implémenté.

— Branch And Bound
— Programme Dynamique (Simple Backward)
— Programme Dynamique (Forward avec liste)
— Programme Dynamique (Core avec liste)
— Plus court chemin dans un graphe

Un modèle du problème est aussi réalisé afin de résoudre les instances dans le solveur CPlex. Cela permet de vérifier les valeurs des solutions des algorithmes implémentés.

Le Branch And Bound utilise des bornes supérieures et inférieures pour couper des branches de l'arbre d'énumération des solutions, sur lequel on réalise un parcours en profondeur. Une borne supérieure est donné par la valeur de la solution partielle actuelle dans l'arbre, à laquelle on ajoute la valeur de la relaxation linéaire du problème partielle restant. Une borne inférieure correspond à la valeur de la meilleure solution complète trouvée à l'instant.

Programme Dynamique Simple backward.

Programme Dynamique Forward avec liste.

Programme Dynamique Core avec liste.

Plus court chemin graphe.

2 Implémentation des algorithmes de résolution du problème de sac à dos binaire

2.1 Branch And Bound spécialisé

2.2 Programme dynamique (simple backward)

2.3 Programme dynamique (forward en liste)

2.4 Programme dynamique (core en liste)

2.5 Plus court chemin dans un graphe

3 Comparaisons

Génération des instances avec un générateur pseudo-aléatoire possédant comme paramètres le poids min et max d'un objet, le profit min et max d'un objet. Générateur linéaire, avec un profit proportionnel au poids. Permet de créer des instances fortement corrélées (tous avec le même ratio profit/poids par exemple).

Comparaisons sur le temps d'exécution ainsi que sur l'empreinte mémoire.

Explications des instances critiques

3.1 Instance critique du Branch And Bound

Paramètres de l'instance (générateur aléatoire)

Capacité	Objets	Poids		Profit	
		min	max	min	max
2000	400	40	44	120	140

	Temps	Mémoire
BaB	>10min	Faible
Backward	0.018s	Moyenne
Forward	0.197s	Faible
Core	>10min	Moyenne
Graphe	>10min	Élevée

Table 1 – Résultats

Remarques : corrélation forte

Paramètres de l'instance (générateur aléatoire)

Capacité	Objets	Poids		Profit	
		min	max	min	max
2000	400	40	44	1	300

	Temps1	Mémoire1	Temps2	Mémoire2	Temps3	Mémoire3
BaB	22.595s	Faible	7.969s	Faible	0.015s	Faible
Backward	0.047s	Moyenne	0.062s	Moyenne	0.062s	Moyenne
Forward	0.031s	Faible	0.047s	Faible	0.11s	Faible
Core	0.063s	Faible	0.047s	Faible	0.015s	Faible
Graphe	>10min	Élevée	>10min	Élevée	>10min	Élevée

Table 2 – Résultats

Remarques : 3 aléatoires avec les mêmes paramètres de génération mais une seed différente

3.2 Instance critique du programme dynamique (simple backward)

Paramètres de l'instance (générateur linéaire)

Capacité	Objets	Poids initial	Profit initial	Corrélation
20000	10000	100	5000	-0.2

	Temps	Mémoire
BaB	>10min	Faible
Backward	0.937s	Élevée
Forward	0.011s	Faible
Core	0.083s	Faible
Graphe	>10min	Élevée

TABLE 3 – Résultats

Remarques : fortement corrélé ; régulier

3.3 Instance critique du programme dynamique (forward avec liste)

Paramètres de l'instance (générateur linéaire)

Capacité	Objets	Poids initial	Profit initial	Corrélation
20000	1000	10	10	1

Remarques

	Temps	Mémoire
BaB	0.002s	Faible
Backward	0.116s	Élevée
Forward	4m14s	Moyenne
Core	>10min	Moyenne
Graphe	>10min	Élevée

TABLE 4 – Résultats

Remarques

3.4 Instance critique du programme dynamique (core avec liste)

Paramètres de l'instance (générateur linéaire)

Capacité	Objets	Poids initial	Profit initial	Corrélation
20000	10000	100	5000	0.2

Remarques

	Temps	Mémoire
BaB	>10min	Faible
Backward	0.892s	Élevée
Forward	0.137s	Faible
Core	6.498s	Moyenne
Graphe	>10min	Élevée

Table 5 – Résultats

3.5 Instance critique du plus court chemin dans un graphe

Paramètres de l'instance

Capacité	Objets	Poids		Profit	
		min	max	min	max
400	200	10	100	1	200

	Temps	Mémoire
BaB	0.002s	Faible
Backward	0.009s	Moyenne
Forward	0.052s	Moyenne
Core	0.006s	Faible
Graphe	12.892s	Élevée

Table 6 – Résultats

Remarques