Projet SMA

Allan DES COURTILS, Thibault THOMAS

---

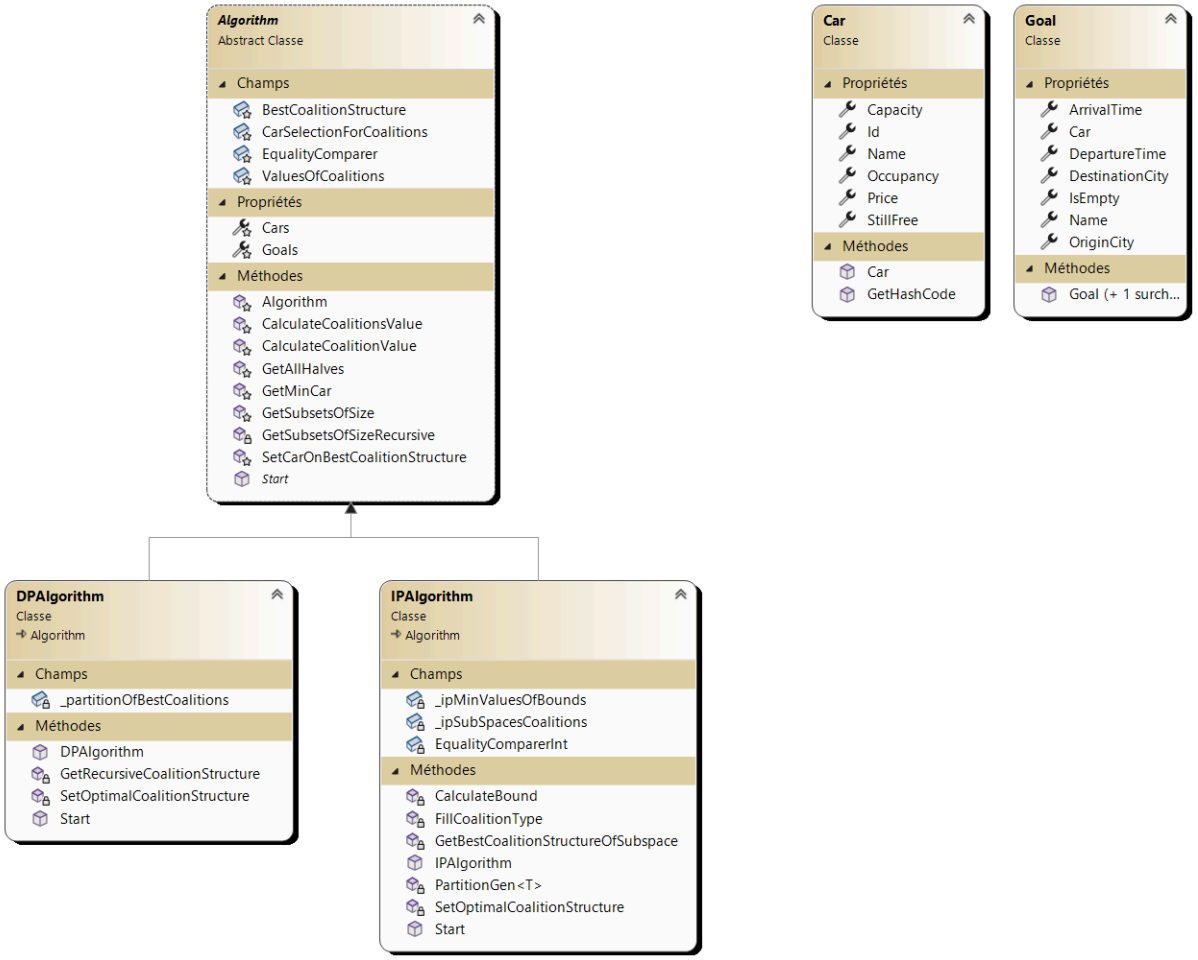
1. Présentation

Le but de ce projet est de créer un système de voyage communautaire similaire à celui de Blablacar. On dispose d'une part d'un ensemble de voyageurs, avec chacun leur ville de départ et une ville de destination, et de l'autre d'un ensemble de véhicules.

Il s'agit d'optimiser le nombre de véhicules occupés pour que tous les voyageurs atteignent leur destination en respectant les dates de départ et d'arrivée ainsi que le prix souhaités. Il s'agit d'un problème de CSG (génération de structure de coalition).

Nous avons choisi de développer le projet avec le langage open-source C# de Microsoft (version 12 du langage et version 8 de .NET). L'architecture et la syntaxe sont de fait assez semblables à celles que l'on pourrait retrouver en Java.

2. Architecture choisie



3. Fonctionnalités implémentées

- **Fonction de calcul de valeur des coalitions (optionnel) → *CalculateCoalitionValue*** :

Afin de calculer les valeurs des coalitions, et de ne pas leurs attribuer une valeur aléatoire, nous avons décidé de réaliser une fonction de calcul qui se base sur plusieurs paramètres :

- **La ville de départ et d'arrivée** : on ajoute 1 000 000 de points par agent qui n'ont pas la même ville de départ, pareil pour la ville d'arrivée.
- **La date de départ** : on ajoute 1 point par minute de décalage par rapport à la date de départ la plus proche.
- **Le prix de la voiture** : on ajoute le prix de la voiture en point, divisé par le nombre d'agents.
- **Pénalité maximale** : s'il n'y a pas assez de voitures par rapport au nombre de groupes de la coalition, on renvoie la valeur maximum du type double.

Le but est donc de minimiser le nombre de points.

- **Algorithme Dynamic Programming (DP) → *DPAlgorithm.cs***
- **Algorithme Improved Dynamic Programming (IDP) → *IPAlgorithm.cs***

4. Démonstration

Dans notre démonstration nous avons **9 agents** tel que :

Nom	**Ville départ**	**Ville arrivée**	**Date départ**
agent1	Paris	Lyon	12/25/2024 10:40:00
agent2	Paris	Lyon	12/25/2024 20:30:00
agent3	Paris	Lyon	12/25/2024 10:30:00
agent4	Paris	Lyon	12/25/2024 20:50:00
agent5	Lyon	Paris	12/26/2024 10:30:00
agent6	Lyon	Paris	12/26/2024 10:30:00
agent7	Marseille	Lyon	12/25/2024 10:40:00
agent8	Paris	Lyon	12/25/2024 10:20:00
agent9	Paris	Lyon	12/25/2024 10:40:00

et **3 types de voiture** :

Nom	**Prix**	**Capacité**
Car1	90	4
Car2	80	3
Car3	200	5

On sait que les deux algorithmes vont avoir le même résultat (un résultat optimal), et à vu d'oeil on sait qu'ils vont regrouper les agent faisant Lyon→Paris et Paris→Lyon ensemble, en faisant soit des voitures de 4, soit de 2 en fonction des dates de départ (10h30 et 20h30) avec une voiture seul pour la personne faisant Marseille→Paris. Ils vont se répartir dans 4 voitures selon le code couleur utilisé dans le tableau.
En regardant la vidéo, on obtient bien ce même résultat pour les deux algorithmes.

5. Fonctionnalités manquantes

- Algorithme Effective Dynamic Programming (EDP)
- Prise en compte de certaines préférences en plus des contraintes standards : nombre de passagers, fumeur ou non, bavard ou non…