

Compte-rendu du projet d'analyse de données: applications au jeu Stardle

Charles Azais de Vergeron et Octave Feuilland

Mai 2025

1 Introduction

1.1 Présentation du jeu Stardle

Le jeu Stardle est un jeu en ligne (stardle.net) basé sur le jeu Wordle / SUTO. Le but est de deviner chaque jour un vaisseau choisi au hasard parmi une liste d'environ 200 éléments.

Chaque vaisseau est défini par deux types de caractéristiques:

- Qualitatives
 - Fabricant (Manufacturer)
 - Type
 - Role
 - Année d'apparition (Release Date)
 - En Jeu ou Non (Status)
- Quantitatives
 - Nombre de personnes dans l'équipage (Crew)
 - Valeur en \$ (Price)
 - Valeurs en monnaie du jeu (Price In game)
 - Capacité en tant que cargo (Cargo Capacity)
 - Vitesse de croisière (SCM) et maximale (Max)
 - Longeur (Length), Largeur (Beam) et Hauteur (Height)

Voici un exemple de partie du jeu :

Nous avons 3 possibilités de résultats: Vrai, Faux ou Proche (uniquement pour les caractéristiques quantitatives).

Ship	Manufacturer	Role(s)	Length	Value	Cargo	Crew	Release year
Cyclone AA	SUMBRIL	Combat	6m	\$80	0 SCU	2 †	2018
Cyclone	SUMBRIL	Expedition	6m	\$55	1 SCU	2 †	2018
	ORIGIN	Touring	5.5m	\$40	0 SCU	1 †	2023
Hurricane		Heavy Fighter	22m	\$210	0 SCU	2 †	2018
O8 Pisces		Pathfinder	16m	Not available for sale	4 SCU	3 †	2019
Freelancer	MISC	Medium Freight	38m	\$10	66 SCU	4 †	2016

Figure 1: Exemple de partie du jeu Stardle

1.2 Objectif du projet

L’objectif de ce projet est d’analyser les données du jeu Stardle afin de créer un algorithme capable de trouver un vaisseau pris au hasard dans la base de données en un minimum de tentatives et de déterminer une combinaison d’essais de vaisseaux qui permet de minimiser le nombre de tentatives.

Nous avons donc récupéré les données et recréé le jeu puis essayé de trouver un algorithme optimal pour déterminer le vaisseau le plus efficacement possible.

2 Récupération des données

La récupération des données est faite via le script Python CreateShipDatabase.py. Ce script utilise l’API officielle de Star Citizen Wiki pour récupérer les informations sur les vaisseaux.

2.1 Structure du script

Le script se décompose en plusieurs étapes principales :

1. Collecte des données

Nous utilisons l’API <https://api.star-citizen.wiki/> pour obtenir la liste complète des vaisseaux. Pour chaque vaisseau, le script récupère les données via l’API et extrait toutes les caractéristiques – dans la configuration ”All” – et uniquement celles du jeu Stardle – dans la configuration ”Stardle” –

Dans la suite du rapport, ”Stardle” désigne la configuration du jeu et ”All” la configuration complète.

2. Stockage des données

- Création de deux fichiers JSON :

- `shipList.json` : liste des vaisseaux et leurs liens
- `shipDB_All.json` : base de données complète des vaisseaux
- `shipDB_Stardle.json` : base de données avec les catégories du jeu Stardle

3 Nettoyage et préparation des données

Pour ajouter de la complexité aux données d'origine, nous avons mélangé les deux bases de données. Stardle est la base du projet. Nous avons extrait de All les valeurs de scm, max, length, beam. Nous avons aussi rajouté une colonne "price in game" qui correspond à la valeur du vaisseau dans le jeu.(via le site Erkul.com)

Pour la gestion des valeurs manquantes, nous avons étudié les dependances entre les colonnes. Nous avons remarqué que certaines colonnes étaient très corrélées entre elles. Par exemple, le prix \$ (sans valeurs manquantes) et celle du prix en jeu (avec beaucoup des valeurs manquantes). Nous avons tenté de remplir les valeurs manquantes en utilisant une regression lineaire et une regression quadratique mais ce ne fut pas concluant. Nous avons donc appliqué l'algorithme suivant:

Algorithm 1: texte

Result: Colonne sans valeurs manquantes
 texte
for *texte* **do**
 | Texte;
 | **if** *condition* **then**
 | | Texte;
 | | Texte;
 | **else**
 | | Texte;
 | **end**
end
 Texte;

4 Implementation du jeu

Nous avons recréé le jeu Stardle en Python : `stardle.py` Nous avons commencé par afficher l'intégralité des vaisseaux du jeu pour éviter au joueur de se tromper lors de la saisie. Nous reproduisons la même logique que le jeu original puis après nous avons à taper le nom du vaisseaux. Ca affiche les résultats.

5 Algorithme de jeu

En suivant les sujets abordés en cours, nous avons décidé d'aborder le problème par la théorie des graphes. Considérons les vaisseaux comme des nœuds, leurs types comme des arêtes par exemple (mais aussi leur rôle, fabricant...). L'avantage du type est qu'un même vaisseau peut en avoir deux, ce qui crée plus de connexions qu'avec les fabricants par exemple. Voici un exemple de graphe :

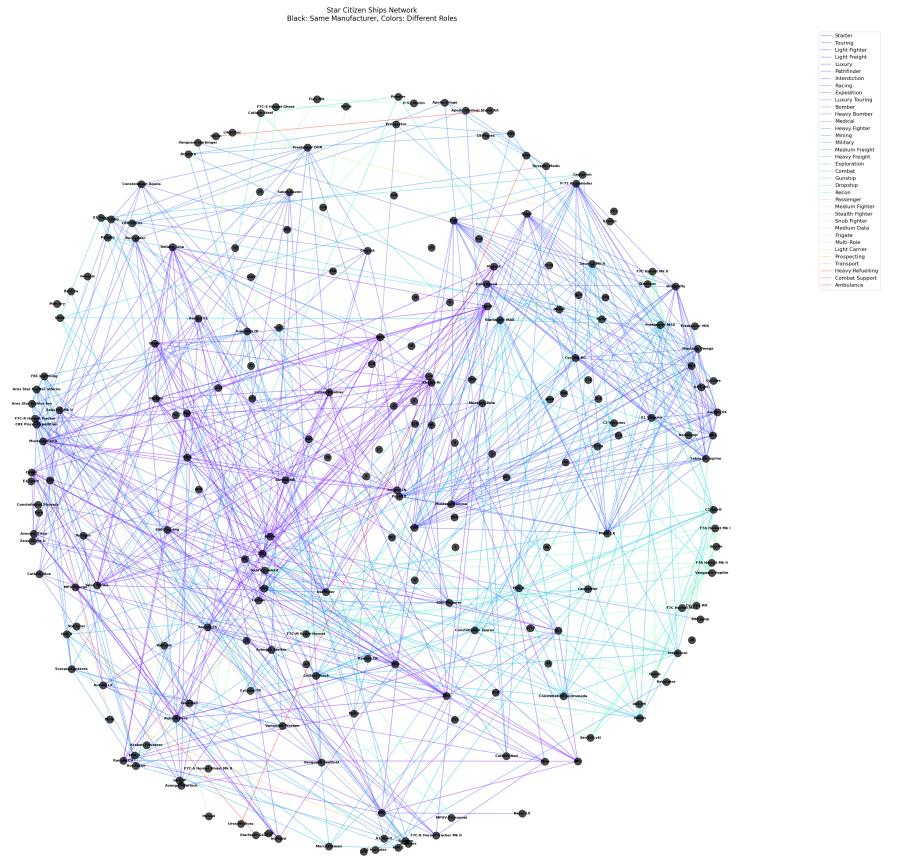


Figure 2: Graphe des vaisseaux par type

Nous étudions donc les vaisseaux du point de vue de la théorie des graphes sur les relations qualitatives (fabricant, type, rôle, année d'apparition, en jeu ou non). Nous avons arbitrairement mis un poids de 1.0 à chaque différence qualitative pour pondérer les chemins entre vaisseaux. Cela nous donne le csv `path_history.csv`. Avec ce csv, nous pouvons déterminer les éléments voisins et ainsi directement les sélectionner pour les prochaines étapes de l'algorithme.

5.1 Algorithme de Stardle

Algorithm 2: Récupération des données des vaisseaux

Data: base de données des vaisseaux

Result: Vaisseau inconnu trouvé

;

;

for chaque vaisseau dans la liste **do**

 Requête GET pour obtenir les détails du vaisseau;

if DataBaseType == "Stardle" **then**

 Extraire: nom, fabricant, rôle, longueur, prix, cargo, équipage;

 Ajouter au ShipDB;

else

 Sauvegarder toutes les données du vaisseau;

end

end

Sauvegarder ShipName dans shipList.json;

Sauvegarder ShipDB dans shipDB-[Type].json;
