Cron Léo 324677
Guillaumat Nathan 330000

Rapport Rendu 3

Description des algorithmes

- **Algorithme de connexion**

L'algorithme de connexion commence son parcours en cherchant un robot dans le rayon de la base (il y en a nécessairement une, un robot de communication est fixe au centre). Ensuite une fonction récursive va chercher un robot qui n'a pas déjà été visité, et qui est dans le rayon du robot de départ. Si il en trouve un, et qu'il appartient à la base qui recherche, il l'ajoute à une liste *listeremote*. Si il n'appartient pas à cette base mais qu'il est tout de même connecté, on l'ajoute à la liste *listconnect*. On appelle ensuite la fonction récursive, cette fois-ci avec le robot trouvé comme robot de départ.

- **Maintenance des robots**

Nous avons jugé que seulement deux robots avaient besoin d'être maintenus : les robots de prospection et de transport. Les robots de forage et de communication vont directement à leur but et ne bougent plus, la maintenance n'est donc pas utile dans ce cas. Les robots de transport font des aller-retour entre la base et un gisement. Lorsqu'un robot de transport est sur la base et qu'il n'a plus assez de distance pour effectuer un aller-retour, il est maintenu. Pour les robots prospecteurs, ils retournent à la base dès que la distance qu'ils peuvent encore parcourir est un petit peu supérieure à leur distance à la base.

- **Création des robots**

Les robots de communication sont créés dès que la ressource est supérieure à une limite de survie (200). Si la base en a moins de 4, elle en crée 4. Si la base a plus de 500 ressources, elle déploie un large réseau de robots de communication afin de couvrir toute la carte. Un robot foreur est créé dès qu'un gisement est découvert, que la ressource de la base est suffisante (>*survie*) et ce robot se dirige vers ce dernier. Les robots de transports sont créés s' il y a moins de robots de transport que de foreurs, si la ressource de la base est suffisante (>*survie*) et ils sont limités au nombre de 8. Un robot de prospection est créé si la ressource de la est suffisante (>*survie*) et si la base en possède moins de 10.

- **Redéfinition des buts**

Si le robot est connecté à la base (mode *remote*) :

- Prospecteurs :
    - Si il est sur une base et si il en a besoin il est maintenu, sinon il est envoyé aléatoirement le long de la diagonale de la carte.
    - Si la distance qu'ils peuvent encore parcourir est un petit peu supérieure à leur distance à la base ils retournent se faire maintenir à la base.
    - Sinon la base leur redéfinit aléatoirement un but.
    - Si ils ont trouvé un gisement ils envoient l'information à la base et celle-ci leur redéfinit un but.

- Foreurs :
    - Si le robot se trouve déjà sur un gisement, il l'adopte et en informe la base.

    - Si le robot est sur la base, le foreur est envoyé au plus grand gisement de la liste *gisements_connus* qui n'a pas encore de foreurs.

- Communication :
    - Un découpage de la carte est fait au préalable dans un attribut de base, comportant les positions des robots de communication. Si une de ces positions n'est pas encore prise par un robot de communication, et que personne d'autre n'a été envoyé dessus, alors il est envoyé sur ce point.

- Transport :
    - Si un robot de transport est sur la base, alors il est envoyé au gisement le plus gros (si les robots de transports déjà envoyés ne permettent pas de vider le gisement). Si le robot est sur un gisement, alors il revient à la base avec un chargement.

Si le robot n'est pas connecté à la base (mode *autonomous*) :

- Prospecteurs :
    - Si le robot a trouvé un gisement, alors il revient en direction de la base. Lorsqu'il est de nouveau connecté, la base lui définit un but aléatoirement.

- Transport :
    - Dès qu'il atteint son but, il retourne à la base.

- **Justification de notre approche**

Notre approche permet au plus grand nombre de base de devenir autonome car celles-ci vont toujours diriger leurs robots de transports en direction des gisements les plus gros, ce qui permet de leur rapporter facilement beaucoup de ressources. Ceci d'autant plus que chaque gisement connu de la base possède un foreur. Les robots de communications étant peu chers, couvrir toute la carte de ces robots permet à la base de connaitre en temps réel toutes les informations sur les gisements (gisement trouvé, épuisé…) et donc de prendre moins de temps dans la prise de décision. Les robots de transport étant assez cher, on préfère les maintenir que d'en recréer (ceux-ci sont d'ailleurs volontairement limités au nombre de 8 pour limiter la consommation de ressources).

Méthodologie et conclusion

Nous avons organisé notre travail en binôme à l'aide d'un drive (Google Drive), qui nous permettait de rendre accessible à tous la dernière version du code, à tout moment. Le travail a été attribué par module de la manière suivante : Nathan s'est principalement occupé des modules *Base*, *Robot* et *Gui*, tandis que Léo s'est occupé de *Simulation*, *Gisement* et *Graphic*.

Nous avons fait le choix de commencer par les modules de plus bas niveau avant de travailler sur les modules plus avancés. C'est une approche que nous jugeons toujours efficace.
Le test des modules a été moins bien réussi car nous écrivions beaucoup de code sans tester toutes les options, ce qui coûtait beaucoup de temps lors du débogage. Avec le recul il aurait fallu prendre plus de temps pour corriger les erreurs pas à pas, cela nous aurait coûté moins de temps.
Le bug le plus fréquent a bien sûr été l'omniprésence de segmentation fault, en raison des nombreux pointeurs que nous utilisons pour mener à bien le projet.

Cron Léo 324677
Guillaumat Nathan 330000

Le bug qui nous a posé le plus de problèmes est lui aussi un segmentation fault, qui survenait lorsque le pointeur de gisement que l'on enregistrait dans les robots transports et foreurs (afin de connaître le gisement qu'ils exploitent) n'existait pas. Il nous a fallu rajouter un booléen *avecGisement* pour contourner le problème, et vérifier l'existence du pointeur avant de l'utiliser.

Nous pensons avoir correctement achevé ce projet, bien qu'il soit difficile de savoir si nous pouvons répondre à toutes les situations possibles. L'environnement mis à disposition est bon, il est utile de pouvoir poser ses questions sur Discours notamment. La VM est très utile mais manque de flexibilité, et la dépendance à une connexion internet est parfois problématique.

Captures d'écran

Fichier de test rendu3_image.txt et rendu3_03.txt