Compte rendu de réunion n°10

**Objet** : Benchmarks et travaux proches de notre problématique. Avancée des testssurles développements génériques et modifications apportées. Article à rédiger.

Date : 09/06/2015 à 10h30.

Participants : Véronique DESLANDRES (**VDE**), Kevin ESPENEL (**KES**).

Rédigé par : Kevin ESPENEL.

**Ordre du jour :**

1 – Benchmarks et autres travaux

2 – Etat d'avancement des tests et développements de la solution générique

3 – Article à rédiger

**Déroulement :**

**1 – Benchmarks et autres travaux**

Vu benchmarks de Véronique sur le wiki, VRPs et problèmes liés au job shop et flow shop où l'on a doit organiser l'utilisation de ressources pour arriver à une allocation optimale.

=>Mail envoyé par Véronique à propos des benchmarks à Thibaut VIDAL (sur son site il référence différentes publications pour différents VRP et autres problèmes d'optimisation ainsi que des benchmarks).

=>Les problèmes abordés dans le domaine des job/flow shops s'approchent de notre problème dans la mesure où l'on cherche une allocation optimale de ressources critiques. Il faut **trouver des benchmarks introduisant le dynamisme** géré par notre solution pour mettre en valeur cet aspect. En général les benchmarks se basent plutôt sur des données fournies à la base et pour lesquelles il faut trouver une bonne allocation, sans l'aspect dynamique où les données évoluent pendant la résolution et impliquent de modifier l'allocation donnée initialement.

Le problème est que la **représentation des données pour ce domaine d'application s'éloigne des données relatives aux livraisons** que nous avions initialement. Il faudra donc certainement **redévelopper une stratégie de génération de propositions** pour les agents, l'algorithme ACO développé pour calculer les propositions initiales sera trop éloigné des données fournies.

=> Les problèmes abordés dans le VRP sont également souvent statiques mais il existe des benchmarks dynamiques. Certains domaines se rapprochent de notre problème comme le VRPTW et le VRPPD ou les notions de fenêtres de temps et de capacités sont intégrées au VRP. Cependant ils **restent proches du VRP** où l'on ne cherche qu'à trouver le meilleur chemin à emprunter pour un seul véhicule.

Les **VRP dynamiques** introduisent la modification possible des données d'entrée pendant la résolution, ce qui implique de modifier la solution donnée initialement. On peut apparenter à notre problème les **VRPPD (Pickup and Delivery) dynamiques**, où de nouveaux clients à livrer peuvent apparaitre après le début de la tournée du livreur. En effet dans notre cas on peut **considérer l'apparition ou la disparition d'un client comme l'occupation ou la libération d'une aire de livraison**. Ainsi il peut être intéressant d'étudier les benchmarks de ce domaine pour voir si l'on peut les adapter à notre problème. L'avantage serait que les données étant plus proches de celles traitées initialement, nous pourrions **réutiliser les algorithmes** de génération initiale des propositions et de génération de nouvelles propositions.

Je continue d'étudier le domaine des DVRP pour chercher des benchmarks correspondant à notre problème et alimenter mon état de l'art. Véronique continue de rechercher des benchmarks plus généraux.

**2 –Etat d'avancement des tests et développements de la solution générique**

- Toutes les méthodes **de résolution ont été revues et corrigées** pour être sûr que les erreurs ne proviennent pas de celles-ci.

- **Résultats toujours pas identiques. Les différences constatées sur de petits échantillons de 2 à 3 agents ont été corrigées** mais d'autres problèmes plus difficiles à localiser apparaissent lors de l'introduction d'un plus grand nombre d'agents.

- Mauvaise approche au départ, il aurait fallut **prendre plus de temps pour modéliser l'adaptation du code et évaluer les risques**. J'ai décidé de revoir l'approche car il restait des différences constatées dans les résultats et de plus en plus difficiles à localiser, entrainant une perte de temps considérable et forte incertitude sur le temps restantpour tout corriger.

- **Certaines méthodes ne sont pas suffisamment documentées et il est donc difficile d'identifier leur comportement attendu.** Par exemple le nombre de propositions filles de chaque nœud est recalculé lors des phases d'influence mais modifié ensuite. Ainsi ce nombre n'est pas à jour durant la phase de perception située avant la phase d'influence suivante.

Cela **a entrainé beaucoup d'erreurs lors de l'adaptation du nouveau code à l'ancienne version**. De plus les méthodes mal documentées sont **difficilement maintenables**, il est donc nécessaire de refaire une modélisation globale plus propre pour s'assurer d'avoir moins d'erreurs.

- J'ai passé **trop de temps à essayer de corriger les erreurs en comparant les résultats des deux versions** =>Etude des résultats puis recherche des causes d'erreurs et correction. Il **aurait fallut limiter le risque d'erreurs à la source** dès le départ en refaisant certaines parties.

Il vaut mieux **reprendre des bases plus saines** plutôt que de continuer à rechercher des erreurs par l'analyse des résultats car cela prend trop de temps. =>**2 à 3 jours pour avoir un code maitrisé et plus maintenable pour la suite puis nouveaux tests avec beaucoup moins de risques d'erreur**.

Réponse :

Le dernier paragraphe ci-dessus donne des éléments d'analyse sur le constat fait dans le 3ème paragraphe si dessus.

Plus précisément le mauvais choix fait a été de supposer que mes modifications s'adapteraient facilement au code existant, car les modifications que j'ai apportées visaient le contenu des propositions et la détections de conflits entre les propositions présentes dans l'environnement. Or il a fallut modifier le lien de l'agent à la proposition et intégrer le fait qu'une proposition est aussi un nœud de l'arborescence de l'agent.

J'ai également voulu optimiser certaines parties, et **clarifier le fonctionnement des méthodes** de résolution (couplage trop fort entre certaines fonctionnalités, par exemple le processus d'influence et la mise à jour des propositions filles d'un nœud). **Ces modifications ont entrainé des différences de résultat visibles dans les logs et très difficile à localiser.**

Donc je n'aurais **pas du directement essayer d'adapter mes évolutions à l'ancienne version sans avoir fait au préalable une phase de modélisation de quelques jours.** Cela aurait permis d'évaluer les risques à priori plutôt que de supposer que les erreurs pourraient être corrigées à la fin.

Enfin cela **met en lumière l'intérêt d'intégrer des modifications de manière itérative en testant chaque fois séparément chaque modification et de la valider avant de passer à la suivante.**

Cela met également en lumière la **faiblesse de l'approche qui consiste à rechercher à partir des résultats les possibles causes de problèmes après avoir tout développé**. Cette méthode ne garantit pas que l'erreur est correctement corrigée, mais simplement que le bon résultat est atteint dans un cas précis. De plus la correction d'une erreur peut en faire apparaitre une autre. Enfin il n'y a aucune maitrise des risques puisque l'on ne sait pas combien il y a d'erreurs en tout.

Au final cette approche a fait que **j'ai rapidement pu intégrer mes modifications à l'ancienne version, mais que la durée de la phase de correction des erreurs était non maitrisée**. J'ai décidé d'arrêter de chercher à résoudre les erreurs de cette manière car celles-ci devenaient de plus en plus difficiles à localiser et corriger.

Pour corriger mon approche j'ai décidé de **repartir sur une nouvelle base et de valider itérativement les changements effectués.** L'objectif est de construire un programme plus maintenable et d'avoir une nouvelle version générique réutilisant tout le fonctionnement du processus de résolution, pas seulement d'intégrer une évolution. Donc plutôt que de prendre mon code et de l'intégrer à l'ancienne version, je vais prendre l'ancienne version pour l'intégrer en mon code en testant itérativement.

Je peux donc procéder par itération en partant de la base et en validant les évolutions progressivement, chaque évolution dans chaque phase étant également validée progressivement.

:

1) Evolution de la "couche basse" = organisation entre les différents composants (nœuds, agents, environnement)

2) Evolutions pour la gestion des liens entre les nœuds, leur ajout et suppression, leur ajout dans l'environnement

3) Evolutions concernant le processus de résolution et la détection des conflits

**Chaque phase prenant environ deux jours je devrais avoir un programme fonctionnel en milieu de semaine prochaine.**

**3 – Article à rédiger**

- Partie rédaction et positionnement du travail gérée par Véronique et Salima, avec les deux autres articles rédigés sur le sujet comme base de départ.

- Je m'occupe de la partie benchmarks avec la comparaison des résultats, les preuves de validité et visualisation (basée sur les graphiques développés avant avec macros excel).

- L'article doit avoir été fini de rédigé fin juin.

**Actions :**

KES

- Avancé des développements.

- Etude de benchmarks DVRP pour voir s'ils peuvent correspondre à notre problème, et alimentation du wiki.

- Voir la visualisation des résultats réalisés par macros.

VDE / SHA

- Alimentation du wiki avec des benchmarks proches de notre problématique.