Problème de collecte et de livraison avec fenêtre de temps et capacité unitaire : résolution de l'ordonnancement et de l'affectation par intelligence collective

Description du problème :

//Décrire le problème : véhicule à capacité unitaire, point de collecte, point de livraison, fenêtre de temps.

L'objectif est de minimiser la distance totale parcourue par les véhicules (ressources) tout en minimisant la durée de dépassement des fenêtres de temps (contraintes temporelles sur les tâches).

Le problème est modélisé sous forme de graphe afin de pouvoir appliquer l'algorithme d'intelligence collective. Le but est de pré-résoudre la partie ordonnancement du problème en permettant uniquement de lier des missions dont l'enchaînement ne provoque pas de dépassement de leurs fenêtre de temps (du moins dans le cas statique). Deux missions sont enchaînnables si la seconde ne débute pas avant la fin de la première et que le temps de parcours entre le point de livraison de la première mission et point de collecte de la seconde mission est suffisament court pour ne pas provoquer le dépassement de la fenêtre de temps de collecte de la seconde mission.

Exemple :

Taches : T = { mission1, mission2, mission3 }
Ressources R = { véhicule1, véhicule2 }

Mission	Conteneur	Point de collecte	Fenêtre de temps de collecte	Point de livraison	Fenêtre de temps de livraison
mission1	JGAU 905811 4	pavé J , travée J-11 , slot 4	00:01:09 – 00:03:17	pavé K , travée K-19 , slot 1	00:03:52 – 00:06:00
mission2	OMEU 487849 6	pavé C , travée C-21 , slot 12	00:01:32 – 00:03:58	pavé G , travée G-22 , slot 5	00:04:21 – 00:06:47
mission3	HZIU 618058 5	pavé C , travée C-21, slot 12	00:06:56 - 00:09:52	pavé E, travée E-17, slot 0	00:09:14 – 00:06:47

Temps de parcours :

	Point de collecte → Point de livraison		Dépot → Point de collecte		Point de livraison → Dépot	
Mission	Véhicule 1	Véhicule 2	Véhicule 1	Véhicule 2	Véhicule 1	Véhicule 2
mission1	00:01:38	00:01:38	00:00:54	00:00:55	00:01:34	00:01:32
mission2	00:01:49	00:01:49	00:01:25	00:01:27	00:01:29	00:01:31
mission3	00:01:38	00:01:38	00:01:22	00:01:24	00:01:30	00:01:31

Point de livraison → Point de collecte						
Mission destination	Mission 1		Mission 2		Mission 3	
Mission origine	Véhicule 1	Véhicule 2	Véhicule 1	Véhicule 2	Véhicule 1	Véhicule 2
mission1			00:02:42	00:02:42	00:02:39	00:02:39
mission2	00:01:40	00:01:40			00:01:46	00:01:46
mission3	00:01:50	00:01:50	00:01:46	00:01:46		

Distances de parcours :

	Point de collecte → Point de livraison		Dépot → Point de collecte		Point de livraison → Dépot	
Mission	Véhicule 1	Véhicule 2	Véhicule 1	Véhicule 2	Véhicule 1	Véhicule 2
mission1	306	306	173	180	347	340
mission2	413	413	334	341	344	351
mission3	396	396	328	335	348	355

Point de livraison → Point de collecte			
Mission destination / Mission origine	Mission 1	Mission 2	Mission 3
mission1		642	636
mission2	317		407
mission3	399	399	

Dans cet exemple les missions 1 et 2 peuvent-être suivies de la mission 3 (00:03:52 + 00:02:39 = 00 :06 :31 < 00:06:56 et 00 :04:21 + 00:01:46 = 00 :06 :07 < 00:06:31). En revanche, il ne peut y avoir d'enchaînnement entre les missions 1 et 2 (00:03:52 > 00:01:32 et 00:04:21 > 00:01:09).

Les solutions possibles sont :

Solution ID	**véhicule 1**	**véhicule 2**
s1	mission 1, mission 3	mission 2
s2	mission 1	mission 2, mission 3
s3	mission 2	mission 1, mission 3
s4	mission 2, mission 3	mission 1

Distances totales parcourues :

	Véhicule 1		Véhicule 2		Total
s1	173+306+636+348	= **1463**	341+413+351	= **1105**	**2568**
s2	173+306+347	= **826**	341+413+407+396+355	= **1912**	**2738**
s3	334+413+344	= **1091**	180+306+636+396+355	= **1873**	**2964**
s4	334+413+407+396+348	= **1898**	180+106+340	= **626**	**2524**

La meilleure solution est la solution 4 (2524 mètres).

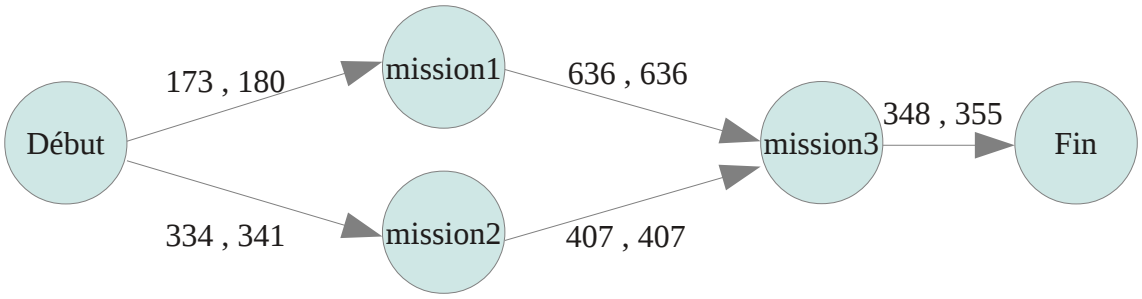
Construction du graphe :

Le graphe est construit comme suit. Chaque mission est représentée par un nœud. Si deux missions sont enchaînnable (voir Description du problème) alors un arc est créé entre les deux nœuds respectifs. Deux autres nœuds sont également créés : un nœud de départ et un nœud de fin. Le nœud de départ est relié à chaque nœud du graphe qui a un degré entrant nul. Les nœuds du graphe qui ont un degré sortant nul sont quant à eux reliés au nœud de fin.

Les arcs sont pondérés par les distances à parcourrir depuis le nœud d'origine de l'arc vers le nœud de destination de l'arc.
Pour un arc reliant le nœud de départ à un nœud de mission, le poid représente la distance entre la position du véhicule à la fin de son activité courante et le point de collecte de la mission du nœud d'arrivée. Si le véhicule est en train d'effectuer une mission, alors le poid correspondra à la distance entre le point de livraison de cette mission et le point de collecte de la mission de destination de l'arc. Dans le cas où le véhicule est inactif, alors le poid de l'arc correspondra à la distance entre la position courante du véhicule et le point de collecte de la mission de destination.

Pour un arc reliant un nœud de mission au nœud de fin, le poid représente la distance à parcourrir entre le point de livraison de la mission et le dépôt. Concernant les arcs reliant deux missions, le poid représente la distance à parcourrir entre le point de livraison de la mission d'origine et le point de collecte de la mission de destination.

Il faut garder à l'esprit que le poid d'un arc peut différer d'un véhicule à un autre, en fonction de leur activité et de leur position. Ainsi, chaque arc contient les pondérations pour chaque véhicule. Le poid des arcs est donc dépendant du temps, même dans le cas statique du problème.



Graphe de mission de l'exemple

Le graphe représente donc les différentes possibilités d'ordonnancement des tâches. Il reste à modéliser la partie affectation du problème.

Fonctionnement de l'algorithme de colonies de fourmis :

Pour résoudre le problème d'affectation des tâches, un algorithme de colonies de fourmis est utilisé. Dans cette version multi-colonies de l'algorithme fourmis, les fourmis sont attirées par la phéromone de leur colonie et repoussées par la phéromone des autres colonies. Il en résulte un comportement de compétition entre les colonies pour parcourir le graphe.

Dans notre modèle, une colonie de fourmis représente un véhicule. Le véhicule doit choisir le meilleur enchainnement de missions pour lui, c'est-à-dire l'enchaînnement qui minimise sa distance parcourue et qui minimise également le temps de dépassement des fenêtres de temps des missions qu'il réalise. Les fourmis de la colonie vont donc devoir coloniser le graphe des missions (voir ci-dessus) afin de trouver le chemin le plus court allant du nœud de départ au nœud de fin (solution locale). Lorsqu'une fourmi choisit un nœud à coloniser elle dépose de la phéromone sur ce nœud en fonction de la qualité de celui-ci. Cette phéromone servira à guider les autres fourmis de la colonie vers ce nœud et à repousser les fourmis des autres colonies vers d'autres nœuds. Or, chaque colonie procède de la même façon, ce qui provoque la répartition des missions entre les colonies (véhicules) tout en convergeant vers une solution minimisant la distance totale parcourue par les véhicule (solution globale).

Chaque colonie est modélisée par une couleur. Ainsi, chaque nœud du graphe prend la couleur de la phéromone la plus présente. La solution est obtenue en construisant les meilleurs chemins pour chaque couleur. Ainsi, le chemin est construit en commencant du nœud de début et en regardant pour chaque nœud accessible s'il est de la couleur du chemin à construire et en prennant celui qui contient la plus grande quantité de phéromone, jusqu'à atteindre le nœud de fin ou de ne plus trouver de nœud accessible de la couleur du chemin à construire.

Les missions de la solution obtenue sont ensuite insérées dans le plan de charge du véhicule représenté par la colonie de fourmis de cette couleur.

Paramètres de l'algorithme :

Il est possible d'influencer le comportement de l'algorithme grâce à plusieurs paramètres :

- α : importance de la trace de phéromone dans le choix d'une destination ;
- β : importance du poid de l'arc dans le choix d'une destination ;
- γ : importance de la quantité de phéromone étrangère sur le choix d'une destination ;
- Δ : seuil de pression de l'environment. Lors du choix d'un nœud par une fourmis, si la part de phéromone de la colonie de cette fourmie dans la quantité totale de phéromone présente sur le nœud est inférieure à D, alors la fourmi meurt (recommence depuis le nœud de départ) ;
- η : nombre de fourmis par colonie ;
- Θ : facteur de synchronisation avec la simulation (nombre d'itération de l'algorithme pour chaque itération de la simulation). Si $\Theta = 0$ alors l'algorithme est désynchronisé et est éxécuté parallèlement à la simulation ;
- λ : quantité de base de phéromone déposée sur un noeux choisit (quantité déposée = λ*(1+(1/w)) (avec λ

> 1) ;

- Λ : quantité de renforcement : quantité de phéromone déposé sur tout le chemin quand la première mission du chemin débute ;
- ς : taux de conservation de phéromone après chaque évaporation (ς ∈ [0 , 1]).

Déroulement de l'algorithme :

```
début
  |       Pour chaque fourmi f de chaque colonie c Faire
  |         |       Noeud destination = choisir_destination() ;
  |         |       Si destination == nul Alors
  |         |       |       retour au nœud de depart ;
  |         |       Sinon
  |         |       |       se_deplacer(destination) ;
  |         |       |       Deposer Pheromone ;
  |         |       Fin Si
  |       Fin Pour
  |       Pour chaque colonie c Faire
  |         |       calculer_chemin(c) ;
  |       Fin Pour
  |       Pour Chaque nœud n Faire
  |         |       Evaporation ;
  |       Fin Pour
fin
```

Le choix de la destination se fait de la façon suivante :

```
Entrée :
        Noeud position
        Colonie couleur
Sortie Noeud
début
  |   double[ ] ρ ← [ ]
  |   int i ← 0
  |   double somme ← 0
  |
  |   Si position.degréSortant == 0 Alors
  |   |   retourne null
  |   Fin Si
  |
  |   Pour chaque Arc a sortant de position Faire
  |   |   Noeud destination ← a.destination ;
  |   |   ρ[i] = destination.pheromone(couleur)^α * 1/a.poid(couleur)^β * ((destination.pheromoneTotale –
  |   |           destination.pheromone(couleur))/destination.pheromoneTotale) ^ γ ;
  |   |   somme ← somme + ρ[i] ;
  |   |   i ← i + 1 ;
  |   Fin Pour
  |
  |   Pour j allant de 0 à ρ.taille Faire
  |   |   ρ[j] ← ρ[j] / somme ;
  |   Fin Pour
  |
  |   Noeud choix ← choixAleatoireBiaisé(ρ,position) ;
  |
  |   Si choix.pheromone(couleur) < Δ Alors
  |   |   choix ← null ;
  |   Fin Si
  |
  |   retourne choix ;
fin
```

La fonction choixAléatoireBiaisé(double[ ] probabilités , Noeud position) retourne un Noeud de destination choisi parmis les destinations accessibles depuis le Noeud position. Le choix s'effectue en fonction des probabilités calculées précédement et placées en entrée de fonction. Plus un arc a une probabilité élevée, plus il a de chance d'être choisi par la fourmi. Toutefois, n'importe quel arc peut être choisi même avec une très faible probabilité.

La quantité de phéromone déposé sur un nœud est calculé selon l'algorithme suivant :

```
Entrée :
    Arc arcEmprunté
    Colonie couleur
début
 |  Noeud destination ← arcEmprunté.destination ;
 |  double q ←λ * (1 + (1 / arcEmprunté.poid(couleur))) ;
 |  destination.pheromone(couleur) ← destination.pheromone(couleur) + q ;
fin
```

Dynamique du graphe :

Dans le cas dynamique du problème, des missions peuvent être ajoutées ou supprimées du pool de missions. D'autres missions peuvent être modifiées également (changement des fenêtres de temps, du point de livraison...). Le graphe de missions doit donc permettre ces modifications.

Ainsi, lors de l'ajout d'une mission sur le graphe les arcs sont générés comme décris précédemment. Si l'ajout de cette mission provoque la création d'un arc vers un nœud relié au nœud de début alors l'arc reliant le nœud de début à l'autre nœud est supprimé. De la même façon, si l'ajout d'un nœud provoque la création d'un arc entre un nœud relié au nœud de fin et ce nouveau nœud, alors, l'arc reliant le nœud au nœud de fin est supprimé.
Ce procéssus est nécessaire afin d'imposer aux véhicules d'effectuer toutes les missions, autrement, la meilleure solution de l'algorithme consisterait à n'effectuer aucune mission car la distance parcourue serait ainsi nulle.

Une mission peut-être supprimée du graphe pour deux raisons. D'une part si elle a été annulée et d'autre part si elle a été réalisée. Le nœud est alors supprimé du graphe et les arcs des nœuds adjacents sont reconstruit en tenant compte des critères évoqués dans le paragraphe précédent.

Lorsqu'une mission est modifiée, le nœud du graphe est alors supprimé puis ré-inséré sur le graphe, ce qui permet de prendre en compte les nouvelles caractéristiques de la mission.

D'autre part, comme vu précédemment, les poids des arcs sont mis-à-jour en fonction de l'activité des véhicules.

Lorsqu'une mission débute, alors le véhicule qui l'éxécute doit la mener à terme sauf s'il tombe en panne. Dans ce cas, il s'agit d'une modification de la mission car le point de collecte peut avoir été modifié si le véhicule a commencé à transporter la marchandise. D'autre part, si le véhicule devient inutilisable, alors les fourmis de la colonie de ce véhicule sont placées sur le nœud de début et doivent y rester jusqu'à ce que le véhicule redevienne utilisable. Pendant ce temps, le processus d'évaporation de phéromone aura fait disparaître l'affectation précédement calculée. Dans le cas où le véhicule ne tombe pas en panne il doit obligatoirement terminer la mission. Afin de représenter cette contrainte dans l'algorithme, les fourmis de la colonie du véhicule débutent obligatoirement leur parcours par le nœud de la mission courante. La phéromone des autres colonies sur ce nœud est également évaporée afin de rendre le nœud inaccessible aux autres colonies de fourmis.

Possibilités :

Le modèle n'est pas encore fixé. Il est possible de modifier différents aspects :

- Synchronisation temporelle avec le simulateur : pour le moment, le facteur de synchronisation est fixe tout au long de la simulation. Mais on pourrait imaginer un processus de gestion du facteur de synchronisation basé sur la dynamique du simulateur. Ainsi, en début de simulation effectuer un grand nombre d'itération de l'algorithme d'optimisation, puis arrêter l'algorithme, puis lors d'un événement ayant un impact sur le graphe relancer l'algorithme un grand nombre d'itérations... Il peut aussi être utile de borner le temps d'éxécution de l'algo.;
- Pression de la population : pour controler le nombre d'individus par colonie il peut s'avérer utile de borner le nombre total de fourmis en fonction du nombre de nœuds du graphe. Il est aussi possible de limiter le nombre de fourmi autorisées à se trouver sur un nœud en même temps. Si cette taille limite est atteinte alors les fourmis ne peuvent plus se rendre sur ce nœud. Si elles n'ont plus de nœud atteignables elles

retournent au nœud de départ...

- Dépot de phéromones : il peut s'avérer utile de sur-marquer un chemin en phéromone afin d'établir une sorte de réservation. Si un chemin complet (du nœud de départ au nœud de fin) est trouvé alors il peut être utile de marquer l'intérêt de ce chemin (utilise une chaine de mission qui rempli le plan de charge du véhicule) par rapport à un chemin incomplet. D'autre part pour ne pas « oublier » les résultat du calcul d'une solution pour un véhicule lorsque la première mission du chemin est retirée du graphe, il peut également être utile de sur-marquer le reste du chemin.