CONTACTS

Dimitri DUVAL
dduval@synora.fr

Alexandre Coubetergues
acoubetergues@synora.fr

ADRESSE

Atout Majeur Concept
11, Bis rue du Château de Ribaute
- ZA de Ribaute
31130 QUINT FONSEGRIVES

Atout Majeur Concept

SAS au capital de 120 000 €
SIRET : 482 114 386 00048
APE : 6202 A
Agrément Formation
73310456931

PROJET REGULATION BRANCARDAGES

Python

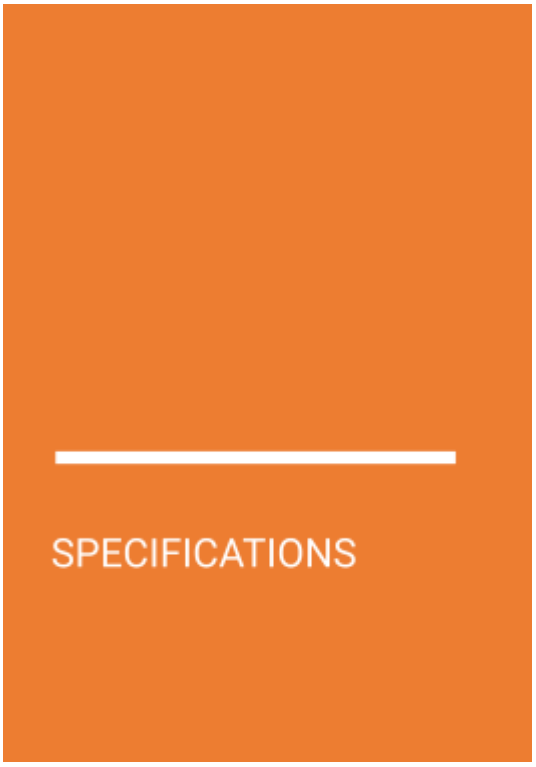


Table des matières

PRÉSENTATION

Le brancardage est l'un des éléments clés de maîtrise des flux de patients et des phénomènes d'engorgement.

Le brancardage recouvre des transports de natures différentes : transport interne dans le service (transports entre boxes et salles d'attente), transport en lien avec les plateaux médico-techniques (bloc opératoire, imagerie, …), transport vers les services cliniques d'hospitalisation.

Afin de fluidifier ces interfaces, différentes pistes d'amélioration méritent d'être explorées : régulation du brancardage, optimisation des mises en brancard/fauteuil, gestion du parc de brancards, etc. Nous nous concentrons ici sur l'automatisation de cette régulation, dans le but de réduire le temps d'attente patient, tout en maximisant la répartition des tâches des brancardiers.

Un brancardage implique plusieurs paramètres, voici un tableau regroupant les éléments caractéristiques d'un brancardage :

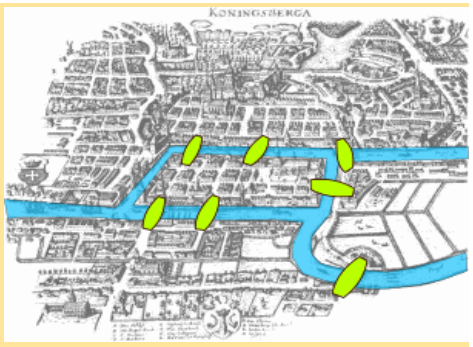
Nom	Définition	Fonction	Acteurs
Service demandeur ou Service d'hospitalisation	Service dont émane le patient	Les agents autorisés de ces services : 1. Réalisent les demandes de rendez-vous préalables 2. Complètent les bons avec la prise en charge des patients afin de générer les bons de transport	Les agents des services désignés et paramétrés comme ayant droit (ex : cadres, responsables d'unité/de pôle, etc.) *Paramétrable par service*
Service receveur	Service où doit se rendre un patient	Les agents autorisés, de ces services, se concentrent sur : 1. La planification des rendez-vous 2. La génération des bons de retour	Les agents des services désignés et paramétrés comme ayant droit (ex : cadres, responsables d'unité/de pôle, secrétaires médicales, etc.) *Paramétrable par service*
Régulateur	Régulateur du service de brancardage	Il gère l'équipe de brancardiers, attribue les bons de transport, gère les plannings.	Le ou les régulateurs/ référents du service de brancardage
Brancardier	Agent réalisant les transports des patients en interne	Les brancardiers déplacent les patients.	L'ensemble des brancardiers du service de brancardage

APPROCHES DU PROBLÈME

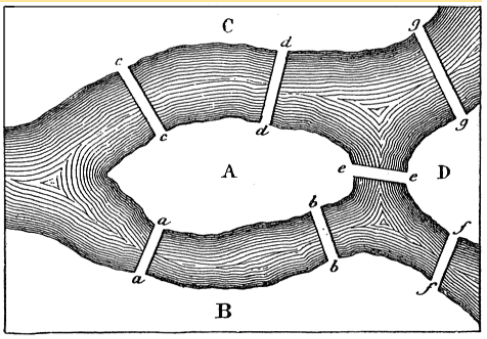
Pour atteindre une solution à notre problème, plusieurs approches et modélisations sont possibles. En effet, nous pouvons choisir de considérer les brancardiers de manière individuelle, ou de les regrouper par équipe, ou encore de considérer que c'est une entité regroupant l'ensemble… il existe ainsi un nombre indéterminé de solutions pour pouvoir répondre à notre problématique.

Parmi les problèmes déjà connus, notre situation peut se retrouver dans le problème du voyageur de commerce (Travelling Salesman Problem, TSP), où le but est de déterminer le plus petit chemin possible d'un ensemble de points en passant 1 seule fois par chacun d'eux.

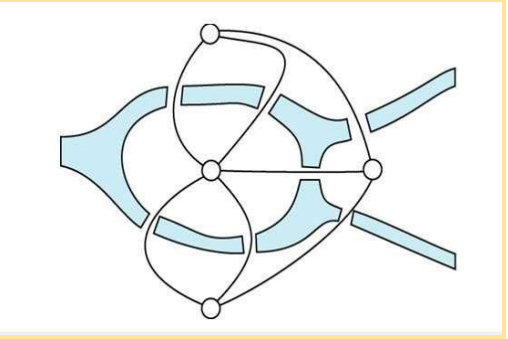
- TSP consiste à modéliser un espace avec les graphes pour en déterminer le plus court chemin. Voici un exemple d'une modélisation d'un espace géographique selon la théorie des graphes



Ville de Konisberg



Plan de la ville de Konisberg



Graphe de la ville de konisberg

A partir de ce(s) graphe(s) trouver le chemin le plus court possible d'un point A vers un point B parmi l'ensemble des chemins possibles de sommets et d'arcs permettant le trajet de A vers B.

SIMULATION PYTHON

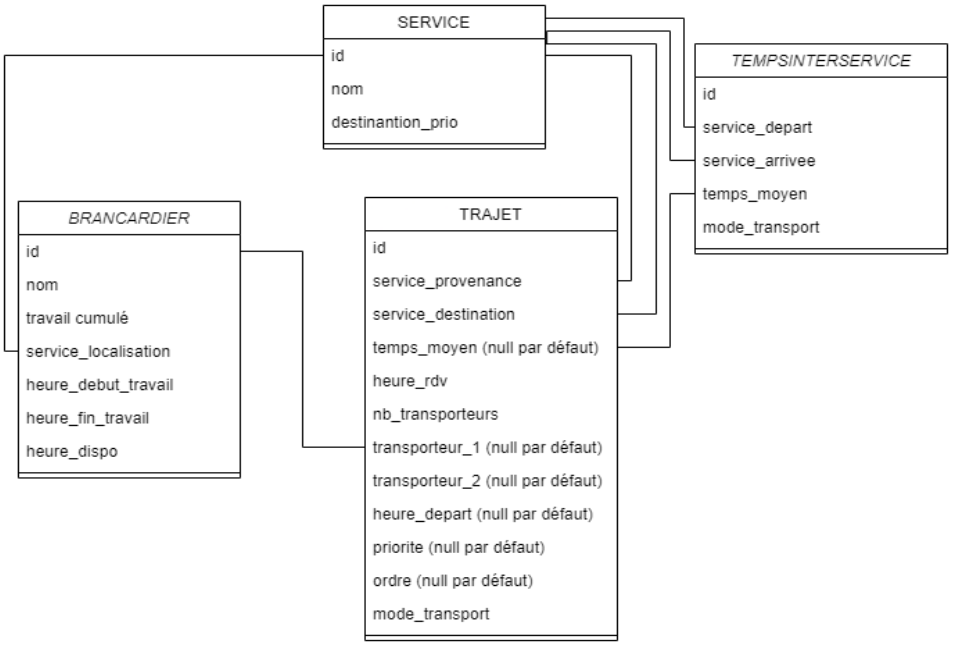
I. SYSTÈME IMAGINÉ

Dans le cadre de cette automatisation, nous avons commencé par imaginer un système simple de classe qui suit la logique suivante :

Un trajet est caractérisé par :

- L'heure de rendez-vous du patient,
- Les services de départ et d'arrivée,
- Le temps de trajet qu'il implique (calculé à partir d'une matrice temps moyen)
- Le(s) brancardier(s) qui va réaliser ce transport

Afin de simuler cette situation nous avons imaginé les objets informatiques suivants :



La classe SERVICE correspond à l'ensemble des services de l'hôpital :

id_service	Attribut numérique unique, fait office d'identifiant de l'objet informatique
nom	Libelle du service
destination_prio	Ordre de priorité lorsque ce service est en destination d'un trajet

La classe TEMPSINTERSERVICE liste les temps de trajets moyens en fonction du service d'origine et du service d'arrivée :

id_tempsMoy	Attribut numérique unique, fait office d'identifiant de l'objet informatique
serviceProvenance	Correspond au service de départ d'un trajet
serviceDestination	Correspond au service d'arrivée d'un trajet
temps_moyen	Correspond au temps de trajet entre les deux services précédents

La classe TRAJET correspond aux demandes de transport de l'hôpital :

ID	Attribut numérique unique, fait office d'identifiant de l'objet informatique
ID service provenance	Correspond au service de départ du trajet
ID service receveur	Correspond au service d'arrivée du trajet
Temps moy	Temps moyen associé au trajet, null par défaut puis calculé à l'aide de TEMPSINTERSERVICES.temps_moyen
Heure de RDV	Heure de rendez-vous du trajet
Nb Brancardiers	Nombre de brancardiers requis pour le transport
Mode de transport	Matériel utilisé pour le transport (fauteuil, lit…)
ID brancardiers	Liste composée de l'ID du ou des brancardiers en charge du transport
Nom brancardiers	Liste composée du nom du ou des brancardiers en charge du transport

heure_depart	Heure pour partir du service de provenance pour arriver à l'heure de rdv dans le service de destination, calculé null par défaut puis calculé à l'aide de la différence entre 'heure_rdv' et TEMPSINTERSERVICES.temps_moyen
priorite	Score de priorité, null par défaut puis calculé à l'aide de 'service_destination' avec SERVICE.destination_prio
ordre	Correspond à l'ordre de passage des différents trajets de la liste, null par défaut puis calculés en fonction des règles de priorité

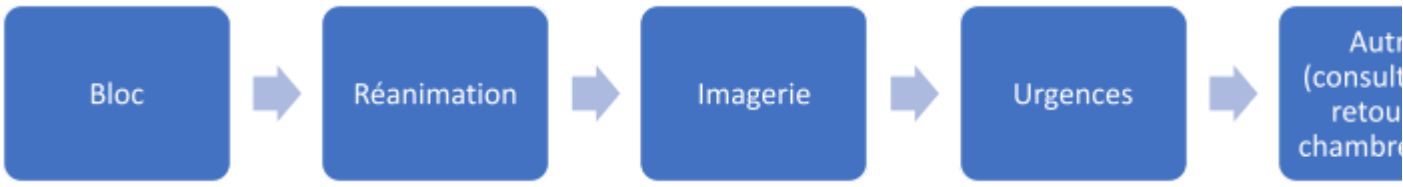
La classe BRANCARDIER liste l'ensemble des brancardiers de l'hôpital :

id	Attribut numérique unique, fait office d'identifiant de l'objet informatique
nom	Nom du brancardier
travail_cumule	Correspond au temps consacré à ses taches de brancardage
service localisation	Id du service où se trouve le brancardier
heure_debut_travail	Heure à laquelle le brancardier commence sa journée
heure_fin_travail	Heure à laquelle le brancardier termine sa journée
heure dispo	Heure à laquelle le brancardier peut accepter une nouvelle demande

II. PRIORISER LES BRANCARDAGES

01 // Règles de priorité en fonction de la destination du transport

En complément à ceux-ci, nous savons qu'il y a un ordre de priorité suivant la destination du transport :



Dans notre cas cette priorisation sera effective sur les demandes se trouvant à 10 min d'intervalle (selon l'heure de départ calculée).

02 // Prendre en compte le nombre de demandes (à programmer après un 1er essai)

Par ailleurs, il est aussi important de prendre en compte le nombre de demandes :

- 3 rdv pour imagerie : 8h, 8h15, 8h30
- 1 rdv pour consultation : entre 8h et 8h30

Dans ce cas notre algorithme va chaîner de sorte à réaliser tous les rdv imagerie avant la consultation

- Imagerie : rdv 8h
- Imagerie : rdv 8h15
- Imagerie : rdv 8h30
- Consultation : rdv entre 8h et 8h30

Le but serait d'obtenir un équilibrage également au niveau des destinations de sorte que tous les services « aient du travail » obtenir l'enchainement suivant :

- Imagerie : rdv 8h
- Imagerie : rdv 8h15
- Consultation : rdv entre 8h et 8h30
- Imagerie : rdv 8h30

03 // Mise en place dans python

Afin d'intégrer ces contraintes, nous avons créés plusieurs fonctions, notamment pour calculer et affecter les temps de parcours ou encore comment ordonner notre liste de trajets :

Affecter les attributs / colonnes vides d'un trajet

Un trajet comme expliqué précédemment est caractérisé par un ensemble d'attributs / colonnes. Parmi eux, certains sont attribués lors de la création de la demande :

Id	Attribut numérique unique, fait office d'identifiant de l'objet informatique
service_provenance	Correspond au service de départ du trajet
service_destination	Correspond au service d'arrivée du trajet
Nb_transporteurs	Nombre de brancardiers requis pour le transport
heure_rdv	Heure de rendez-vous du trajet
mode de transport	Matériel utilisé pour le transport (fauteuil, lit…)

D'autres sont à déterminer à partir des données de bases, et autres données d'autres tables (SERVICE ou encore TEMPSINTERSERVICES) :

temps_moyen	Temps moyen associé au trajet, null par défaut puis calculé à l'aide de TEMPSINTERSERVICES.temps_moyen
transporteurs	Liste des ID des brancardiers réalisant le trajet, null par défaut, déterminé par le programme
heure_depart	Heure pour partir du service de provenance pour arriver à l'heure de rdv dans le service de destination, calculé null par défaut puis calculé à l'aide de la différence entre 'heure_rdv' et TEMPSINTERSERVICES.temps_moyen
priorite	Score de priorité, null par défaut puis calculé à l'aide de 'service_destination' avec SERVICE.destination_prio
ordre	Correspond à l'ordre de passage des différents trajets de la liste, null par défaut puis calculés en fonction des règles de priorité

Pour attribuer leurs valeurs, nous avons créée plusieurs fonctions python de la forme : `nomFonction(parametres)`

temps_moyen	`affecterTempsMoyen(trajet)`
transporteurs	`affecterBrancardierTrajet(trajet)`
heure_depart	`affecterHeureDepart(trajet)`
priorite	`affecterPriorite(trajet)`
ordre	`triTrajets(listTrajets, tempsApresRdv, tempsAvantRdv)`

Également nous avons défini des getters pour récupérer les valeurs d'un objet selon sa classe.

On peut diviser ici les fonctions selon un degré de complexité :

- Degré 1 : récupération de valeurs depuis une autre classe associé, impliquant peu de calcul tel qu'une somme
- Degré 2 : Détermination de valeurs par enchainements de calculs et de boucles, implique plusieurs calculs conditionnels, considérés plus complexe que les fonctions de Degré 1

FONCTION DE COMPLEXITÉ 1

temps_moyen	`affecterTempsMoyen(trajet)`
heure_depart	`affecterHeureDepart(trajet)`
priorite	`affecterPriorite(trajet)`

Ces fonctions permettent principalement de récupérer des informations des autres classes. En suivant vous trouverez des diagrammes décrivant le comportement de nos fonctions.

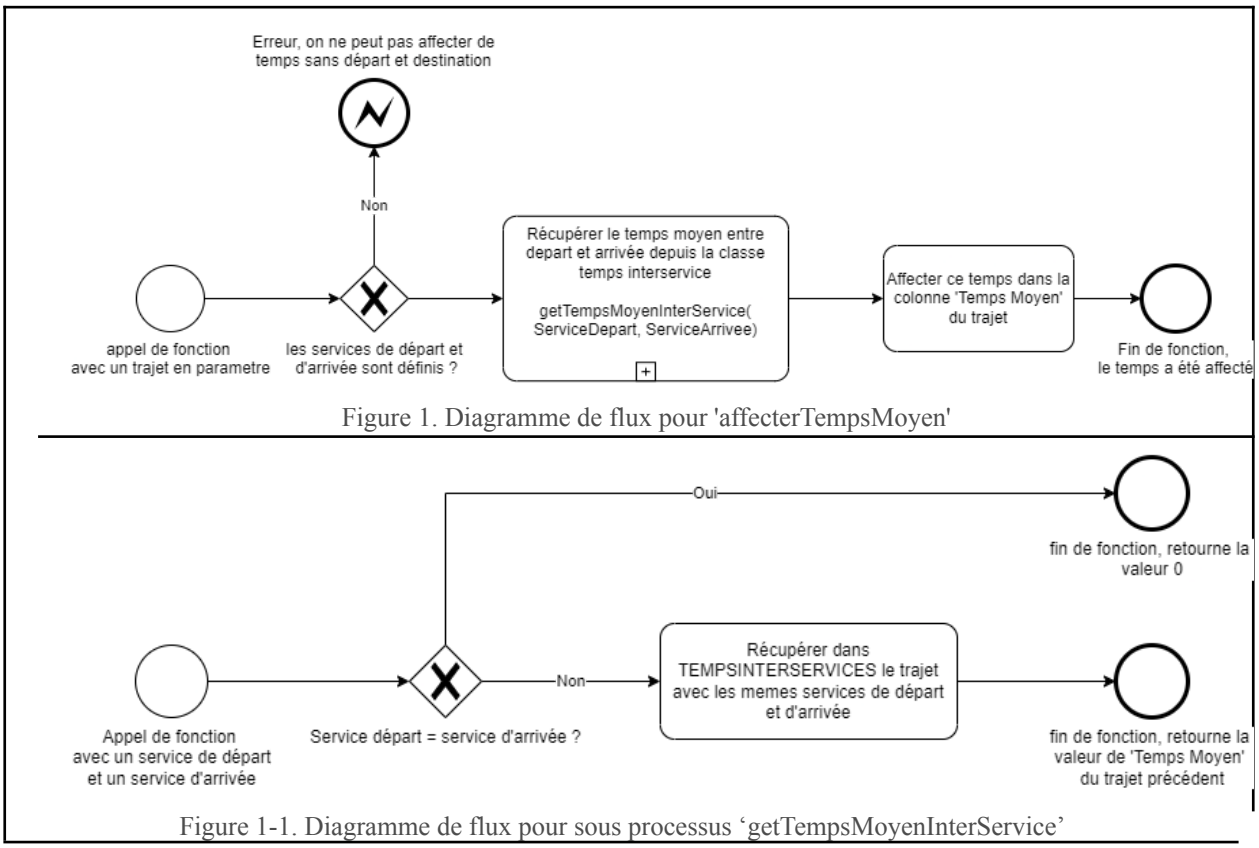


Figure 1. Diagramme de flux pour 'affecterTempsMoyen'

Figure 1-1. Diagramme de flux pour sous processus ‘getTempsMoyenInterService’

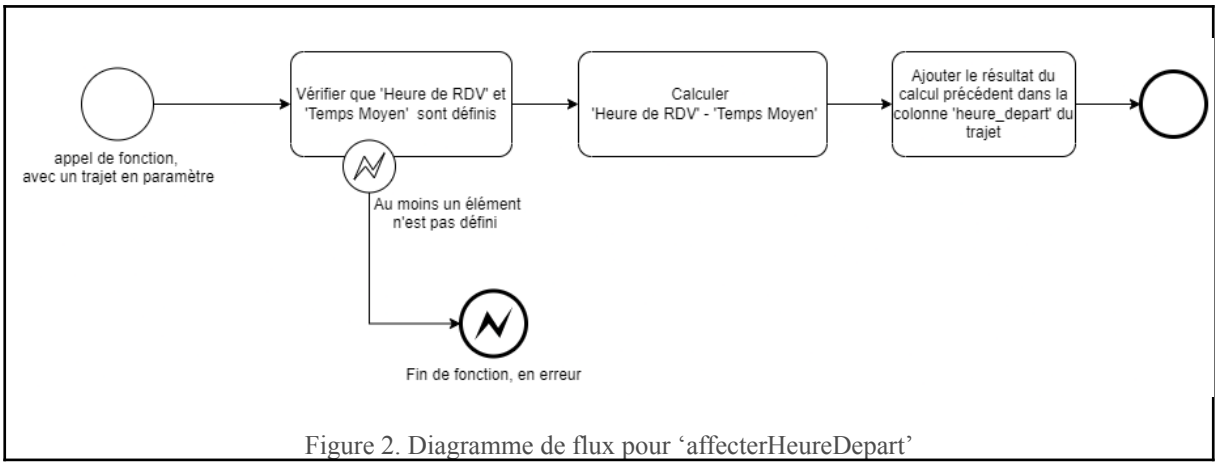


Figure 2. Diagramme de flux pour ‘affecterHeureDepart’

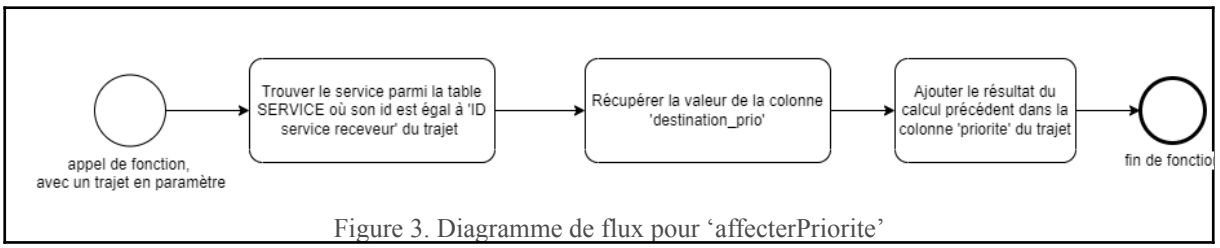


Figure 3. Diagramme de flux pour ‘affecterPriorite’

FONCTION DE COMPLEXITÉ 2

Ces fonctions permettent de réaliser les tâches principales de notre projet : trier les trajets par règles de priorité et affecter les brancardiers selon leur disponibilité et proximité. En suivant vous trouverez des diagrammes décrivant le comportement de ces fonctions.

transporteurs	`affecterBrancardierTrajet(trajet)`
ordre	`triTrajets(listTrajets, tempsApresRdv, tempsAvantRdv)`

`affecterBrancardierTrajet(trajet)` sera détaillé dans la partie suivante, partie qui se concentre sur les propriétés et les fonctions des brancardiers.
Nous verrons ici, seulement la fonction **`triTrajets.`**

Nous avons pour chaque trajet :

- Un nombre décrivant un indice de priorité contenu dans la colonne ‘priorite’ (déterminé par **`affecterPriorite`**). Un trajet est prioritaire à un autre si son indice est inférieur à celui de l’autre trajet.
- Un nombre décrivant le temps moyen en secondes pour faire ce trajet (déterminé par **`affecterTempsMoyen)`**
- Une heure de départ donnant l’heure minimale pour partir du service de provenance afin d’arriver à l’heure de rendez-vous dans le service d’arrivée (déterminée par **`affecterHeureDepart`**).

Exemple de trajets après différentes affectations :

id_traj	serv_pro	serv_dest	tps_moy	heure_rdv	nb_transporteurs	transporteurs	heure_depart	priorite	ordre
1	0	1	394	08h00	2		07h53	1	
2	0	2	300	08h03	1		07h58	2	
3	0	3	120	08h06	2		08h04	0	
4	0	1	394	08h09	1		08h02	1	
5	5	1	60	08h12	1		08h11	1	
6	1	4	340	08h15	1		08h09	4	
7	2	3	292	08h18	2		08h13	0	
8	0	3	120	08h21	1		08h19	0	
9	3	2	146	08h24	1		08h21	2	
10	4	3	10	08h27	2		08h26	0	
11	5	1	60	08h30	1		08h29	1	
12	1	2	75	08h33	1		08h31	2	
13	0	2	300	08h36	2		08h31	2	
14	3	1	180	08h39	2		08h36	1	
15	3	4	20	08h42	1		08h41	4	

Le but va être de compléter la colonne ‘ordre’ pour déterminer l’enchaînement des trajets selon l’ordre de priorité et une ”fenêtre de temps” par rapport à l’heure de départ. Dans un premier temps, on vient trier notre liste selon les heures de départ :

id_traj	serv_pro	serv_dest	tps_moy	heure_rdv	nb_transporteurs	transporteurs	heure_depart	priorite	ordre
1	0	1	394	08h00	2		07h53	1	0
2	0	2	300	08h03	1		07h58	2	3
4	0	1	394	08h09	1		08h02	1	2
3	0	3	120	08h06	2		08h04	0	1
6	1	4	340	08h15	1		08h09	4	7
5	5	1	60	08h12	1		08h11	1	6
7	2	3	292	08h18	2		08h13	0	4
8	0	3	120	08h21	1		08h19	0	5
9	3	2	146	08h24	1		08h21	2	10
10	4	3	10	08h27	2		08h26	0	8
11	5	1	60	08h30	1		08h29	1	9
12	1	2	75	08h33	1		08h31	2	12
13	0	2	300	08h36	2		08h31	2	13
14	3	1	180	08h39	2		08h36	1	11
15	3	4	20	08h42	1		08h41	4	14

A partir de cette liste triée, nous allons regarder les trajets qui se trouvent moins de 10 min plus tard que notre trajet sélectionné. Par exemple pour le trajet d'ID 1, nous avons les trajets d'ID 2 et 4.

id_traj	serv_pro	serv_dest	tps_moy	heure_rdv	nb_transporteurs	transporteurs	heure_depart	priorite	ordre
1	0	1	394	08h00	2		07h53	1	
2	0	2	300	08h03	1		07h58	2	
4	0	1	394	08h09	1		08h02	1	
3	0	3	120	08h06	2		08h04	0	
6	1	4	340	08h15	1		08h09	4	

On remarque que selon la priorité des trajets et l'heure de départ prévu, la bonne séquence est 1 -> 4 -> 2

On refait cette comparaison avec le trajet suivant : pour le trajet d'ID 2, nous avons les trajets d'ID 4 et 3 qui se trouvent à moins de 10 min après.

id_traj	serv_pro	serv_dest	tps_moy	heure_rdv	nb_transporteurs	transporteurs	heure_depart	priorite	ordre
1	0	1	394	08h00	2		07h53	1	
2	0	2	300	08h03	1		07h58	2	
4	0	1	394	08h09	1		08h02	1	
3	0	3	120	08h06	2		08h04	0	
6	1	4	340	08h15	1		08h09	4	

Maintenant, on remarque que selon la priorité des trajets et l'heure de départ prévu, la bonne séquence est 3 -> 4 -> 2.

Donc notre séquence actuelle combinée avec le précédent (1 -> 4 -> 2) devrait ressembler à 1 -> 3 -> 4 -> 2.

Puis, on refait la même chose pour le suivant : pour le trajet d'ID 4, nous avons les trajets d'ID 3, 6 et 5 qui se trouvent à moins de 10 min après.

id_traj	serv_pro	serv_dest	tps_moy	heure_rdv	nb_transporteurs	transporteurs	heure_depart	priorite	ordre
1	0	1	394	08h00	2		07h53	1	
2	0	2	300	08h03	1		07h58	2	
4	0	1	394	08h09	1		08h02	1	
3	0	3	120	08h06	2		08h04	0	
6	1	4	340	08h15	1		08h09	4	
5	5	1	60	08h12	1		08h11	1	
7	2	3	292	08h18	2		08h13	0	

Ce qui nous donne comme nouvelle séquence : 3 -> 4 -> 5 -> 6. En combinant avec les précédents (1 -> 3 -> 4 -> 2), on voudrait obtenir la séquence : 1 -> 3 -> 4 -> 2 -> 5 -> 6.

Et on continue ces étapes jusqu'au dernier trajet de la liste, ce qui nous donnera une séquence finale avec des index correspondants à l'ordre d'agencement de nos trajets :

id_traj	serv_pro	serv_dest	tps_moy	heure_rdv	nb_transporteurs	transporteurs	heure_depart	priorite	ordre
1	0	1	394	08h00	2		07h53	1	0
2	0	2	300	08h03	1		07h58	2	3
4	0	1	394	08h09	1		08h02	1	2
3	0	3	120	08h06	2		08h04	0	1
6	1	4	340	08h15	1		08h09	4	7
5	5	1	60	08h12	1		08h11	1	6
7	2	3	292	08h18	2		08h13	0	4
8	0	3	120	08h21	1		08h19	0	5
9	3	2	146	08h24	1		08h21	2	10
10	4	3	10	08h27	2		08h26	0	8
11	5	1	60	08h30	1		08h29	1	9
12	1	2	75	08h33	1		08h31	2	12
13	0	2	300	08h36	2		08h31	2	13
14	3	1	180	08h39	2		08h36	1	11
15	3	4	20	08h42	1		08h41	4	14

Si on trie par 'ordre' cela nous donne la séquence ci-après :

id_traj	serv_pro	serv_dest	tps_moy	heure_rdv	nb_transporteurs	transporteurs	heure_depart	priorite	ordre
1	0	1	394	08h00	2	[1]	07h53	1	0
3	0	3	120	08h06	2	[4]	08h04	0	1
4	0	1	394	08h09	1	[3]	08h02	1	2
2	0	2	300	08h03	1	[2]	07h58	2	3
7	2	3	292	08h18	2	[4]	08h13	0	4
8	0	3	120	08h21	1	[6]	08h19	0	5
5	5	1	60	08h12	1	[6]	08h11	1	6
6	1	4	340	08h15	1	[5]	08h09	4	7
10	4	3	10	08h27	2	[6]	08h26	0	8
11	5	1	60	08h30	1	[5]	08h29	1	9
9	3	2	146	08h24	1	[2]	08h21	2	10
14	3	1	180	08h39	2	[3]	08h36	1	11
12	1	2	75	08h33	1	[6]	08h31	2	12
13	0	2	300	08h36	2	[1]	08h31	2	13
15	3	4	20	08h42	1	[6]	08h41	4	14

III. AFFECTER UN BRANCARDIER SELON DES RÈGLES

Dans le cadre de l'affectation d'un brancardier à un transport, il faut imaginer plusieurs critères de sélection. Comme expliqué en introduction, nous cherchons à minimiser l'attente de patient, tout en maximisant le travail des brancardiers.

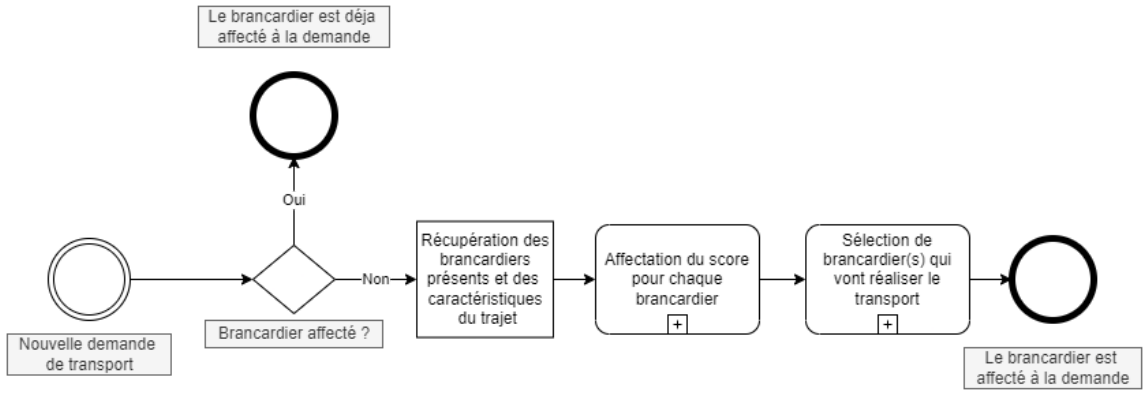


Figure BPM Général de l'algorithme d'affectation de brancardier

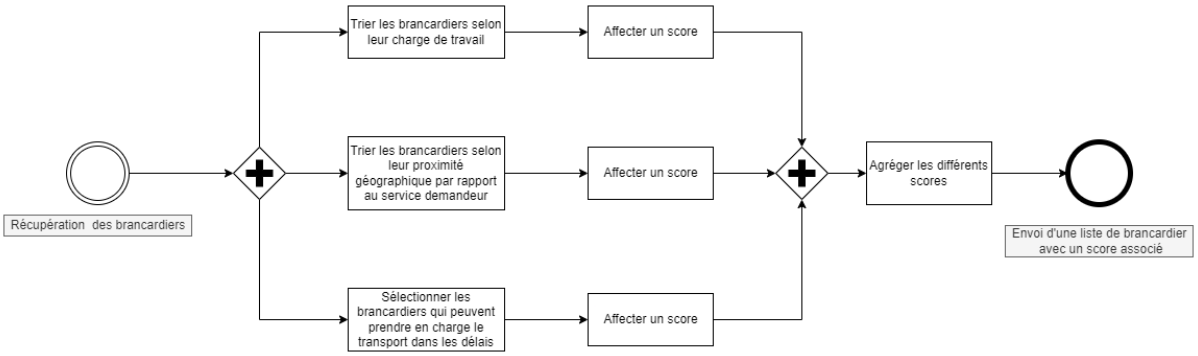


Figure Sous processus d'affectation de score

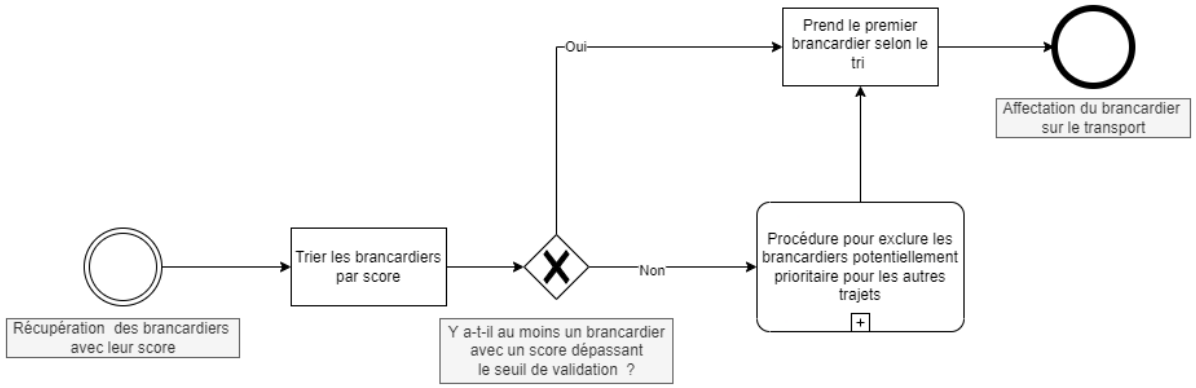


Figure Sous processus de sélection du brancardier à affecter

- Un brancardier peut prendre un brancardage si :
  - Il se trouve à moins de 10 min du service départ de la demande
    - Implique la connaissance de la proximité entre services (matrice de temps) selon le service de localisation du brancardier
    - Implique la connaissance de la disponibilité du brancardier pour être à l'heure de rdv
- Un brancardier est prioritaire pour prendre en charge la demande si :

    - Sa charge de travail est inférieure à celle d'un autre brancardier pouvant prendre en charge la demande
        - Implique la mise à jour automatique de sa charge de travail
- Si pas de brancardier prioritaire déterminé,
    - On regarde les brancardiers qui peuvent être potentiellement prioritaire sur trajets suivants et proches selon un temps déterminé (= est-ce qu'un brancardier est prioritaire sur un autre trajet prévu dans les 10 min)
        - On les exclut de la sélection en cours

Une fois les brancardiers affectés à chaque trajet, la fonction **AffecterTrajetBrancardier()** permet à l'inverse d'affecter à chaque brancardier une liste contenant tous les trajets qu'il a réalisés dans une journée et ainsi d'obtenir une vue différente par brancardier avec ses trajets, le nombre de transports qu'il a réalisé ainsi que la quantité de travail qu'il a cumulé dans la journée.

La fonction **AffecterTrajetBrancardier()** fonctionne de la manière suivante :

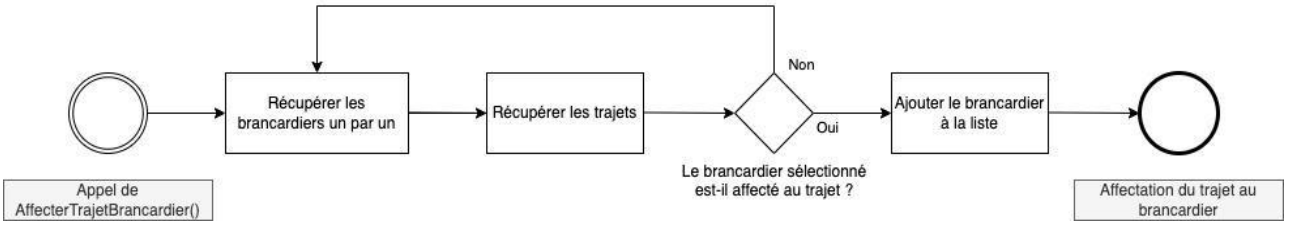


Figure BPM de la fonction d'affectation des trajets à un brancardier

La fonction **recupTravailBrancardier()** permet quant à elle de récupérer le temps de travail cumulé des brancardiers en cumulant les temps de trajets de leurs missions. Ce temps permet de vérifier l'égalité de temps de travail entre les brancardiers au cours de leur journée de travail.

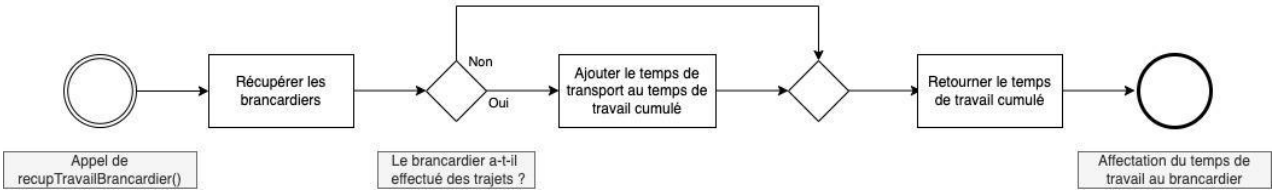


Figure BPM de la fonction de récupération du temps de travail d'un brancardier

Enfin, la fonction **exec()** a le rôle de fonction finale qui exécute une majeure partie des fonctions précédentes afin de réaliser les affectations et les tris nécessaires à l'affichage final de nos résultats.

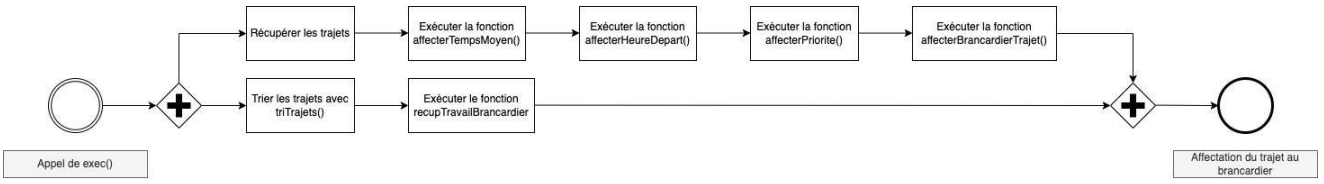


Figure BPM de la fonction d'exécution des fonctions

IV. RÉSULTATS FINAUX

La fonction **exec()** permet de centraliser les appels aux différentes fonctions, et d'ainsi générer par la suite le résultat final représenté par deux fichiers CSV :

- RES_trajets.csv
- RES_brancardiers.csv

Le fichier **RES_trajets.csv** dont un exemple se situe ci-dessous, contient les informations essentielles à chaque trajet, avec bien évidemment l'affectation du ou des brancardier(s) chargé(s) du transport mais également l'heure de départ à respecter, l'ordre des trajets à respecter en fonction des priorités etc...

ID	ID service provenance	ID service receveur	Temps moy	Heure de RDV	Nb Brancardiers	Retour	Mode de transport	ID brancardiers	Nom brancardiers	heure_depart	priorite	ordre
3682	62	60	2220	07h40	2	0	Lit	['10', '13']	['HULEUX Dominique', 'VAN NIEUWENHOVE Ludivine']	07h03	5	0
3683	62	60	2220	08h00	2	0	Brancard	['1', '2']	['COCHELARD Jonathan', 'BRANCARDIER01 Utilisateur']	07h23	5	1
3687	26	70	1860	08h10	1	0	Fauteuil	['3']	['BRANCARDIER02 Utilisateur']	07h39	6	2
3686	62	29	540	08h50	1	0	Fauteuil	['4']	['GRANDFILS Celine']	08h41	5	3
3696	18	60	420	09h20	2	0	Lit	['5', '6']	['DA SILVA Yohan', 'MOUCHERON Aurelien']	09h13	5	4
3689	60	51	900	09h33	1	1	Brancard	['7']	['PATYNA Guy-Christian']	09h18	6	5
3688	29	62	900	09h35	1	1	Fauteuil	['8']	['TAOUTAOU Alain']	09h20	6	6
3692	60	51	900	09h49	1	1	Brancard	['9']	['CAULLERY Fabrice']	09h34	6	7
3697	70	26	900	10h20	1	1	Fauteuil	['11']	['TERKI Tamine']	10h05	6	11
3693	26	157	180	10h10	1	0	Brancard	['12']	['MOUCHART Cyrille']	10h07	1	8
3694	26	157	180	10h10	2	0	Lit	['14', '15']	['RADENNE Gilles', 'DOUILLY Alexandre']	10h07	1	9
3695	26	157	180	10h13	2	0	Lit	['16', '17']	['DUPONT Laurent', 'FONTENELLE Theo']	10h10	1	10
3698	26	157	180	10h17	1	0	Brancard	['18']	['DA SILVA VILAR Benoit']	10h14	1	12
3701	26	70	1860	10h55	1	0	Fauteuil	['19']	['LEGRAND Arnaud']	10h24	6	14
3678	110	81	300	10h30	1	0	Brancard	['20']	['LARIVIERE Coralie']	10h25	0	13
3699	60	51	900	10h43	1	1	Brancard	['12']	['MOUCHART Cyrille']	10h28	6	15
3704	26	70	1860	11h10	1	0	Fauteuil	['14']	['RADENNE Gilles']	10h39	6	16
3700	157	26	900	10h55	1	1	Brancard	['15']	['DOUILLY Alexandre']	10h40	6	17
3703	81	110	900	11h00	1	1	Brancard	['16']	['DUPONT Laurent']	10h45	6	18
3705	60	51	900	11h12	1	1	Brancard	['17']	['FONTENELLE Theo']	10h57	6	20
3711	62	60	2220	11h35	2	0	Lit	['18', '20']	['DA SILVA VILAR Benoit', 'LARIVIERE Coralie']	10h58	5	19
3707	60	62	900	11h16	2	1	Lit	['5', '6']	['DA SILVA Yohan', 'MOUCHERON Aurelien']	11h01	6	21
3706	90	74	300	11h15	2	0	Lit	['4', '7']	['GRANDFILS Celine', 'PATYNA Guy-Christian']	11h10	2	22
3709	60	62	900	11h30	1	1	Brancard	['8']	['TAOUTAOU Alain']	11h15	6	23
3690	18	301	180	11h25	1	0	Brancard	['9']	['CAULLERY Fabrice']	11h22	6	24
3712	60	51	900	11h46	1	1	Brancard	['4']	['GRANDFILS Celine']	11h31	6	25
3713	157	26	900	11h50	2	1	Lit	['11', '12']	['TERKI Tamine', 'MOUCHART Cyrille']	11h35	6	26
3714	70	26	900	11h55	1	1	Fauteuil	['15']	['DOUILLY Alexandre']	11h40	6	27
3715	301	18	900	12h00	1	1	Brancard	['9']	['CAULLERY Fabrice']	11h45	6	28
3710	86	828	300	12h00	1	0	Brancard	['16']	['DUPONT Laurent']	11h55	6	29
3716	60	90	900	12h10	1	1	Brancard	['17']	['FONTENELLE Theo']	11h55	6	30
3717	60	51	900	12h12	1	1	Brancard	['7']	['PATYNA Guy-Christian']	11h57	6	31
3718	60	18	900	12h21	2	1	Lit	['5', '6']	['DA SILVA Yohan', 'MOUCHERON Aurelien']	12h06	6	32
3720	828	86	900	12h40	1	1	Brancard	['16']	['DUPONT Laurent']	12h25	6	33
3722	70	26	900	12h40	1	1	Fauteuil	['4']	['GRANDFILS Celine']	12h25	6	34
3724	60	51	900	12h46	1	1	Brancard	['8']	['TAOUTAOU Alain']	12h31	6	36
3685	26	81	240	12h40	1	0	Fauteuil	['11']	['TERKI Tamine']	12h36	0	35

Exemple du fichier RES_trajets.csv pour une journée type à l'hôpital de Maubeuge

Le fichier **RES_brancardiers.csv** également en exemple, permet une vue par brancardier. On peut ainsi facilement savoir quels trajets sont réalisés par un brancardier en particulier, et ainsi connaître leur temps de travail cumulé afin de respecter l'égalité du temps de travail.

ID	Nom brancardier	Trajets	Nombre de transports	Travail Cumulé
1	COCHELARD Jonathan	['3683', '3730', '3744']	3	01h07
2	BRANCARDIER01 Utilisateur	['3683', '3731', '3742', '3747', '3755']	5	01h15
3	BRANCARDIER02 Utilisateur	['3687', '3725', '3741']	3	01h01
4	GRANDFILS Celine	['3686', '3706', '3712', '3722', '3727', '3749']	6	01h02
5	DA SILVA Yohan	['3696', '3707', '3718', '3731', '3676']	5	01h01
6	MOUCHERON Aurelien	['3696', '3707', '3718', '3729', '3676']	5	01h01
7	PATYNA Guy-Christian	['3689', '3706', '3717', '3680', '3708', '3721', '3739', '3751']	8	01h09
8	TAOUTAOU Alain	['3688', '3709', '3724', '3681', '3750']	5	01h09
9	CAULLERY Fabrice	['3692', '3690', '3715', '3723', '3733', '3745']	6	01h08
10	HULEUX Dominique	['3682', '3729', '3746']	3	01h07
11	TERKI Tamine	['3697', '3713', '3685', '3669', '3736', '3753']	6	01h27
12	MOUCHART Cyrille	['3693', '3699', '3713', '3679', '3702', '3719', '3681', '3751']	8	01h09
13	VAN NIEUWENHOVE Ludivine	['3682', '3732', '3740']	3	01h01
14	RADENNE Gilles	['3694', '3704', '3680', '3735', '3743']	5	01h01
15	DOUILLY Alexandre	['3694', '3700', '3714', '3679', '3732', '3738']	6	01h01
16	DUPONT Laurent	['3695', '3703', '3710', '3720', '3736', '3748']	6	01h08
17	FONTENELLE Theo	['3695', '3705', '3716', '3726', '3749']	5	01h03
18	DA SILVA VILAR Benoit	['3698', '3711', '3719', '3737', '3754']	5	01h14
19	LEGRAND Arnaud	['3701', '3684', '3728', '3677', '3747', '3755']	6	01h15
20	LARIVIERE Coralie	['3678', '3711', '3734', '3752']	4	01h12

Exemple du fichier RES_brancardiers.csv pour une journée type à l'hôpital de Maubeuge

SOURCES/BIBLIOGRAPHIE

Présentation

ANAP _ Optimiser le brancardage dans le service d'urgence :
https://ressources.anap.fr/urgences/publication/1762-optimiser-le-brancardage-dans-le-service-d-urgence