GESTION DES MOYENS AU SOL D'UNE PLATEFORME AÉROPORTUAIRE

PROJET TECHNIQUE

Laura CORMERAIS
Anna LLORT ESCOTÉ
IENAC 08T

ECOLE NATIONALE DE L'AVIATION CIVILE

REMERCIEMENTS

Nous remercions toutes les personnes qui ont participé à la réalisation de notre projet. Particulièrement à **Catherine Mancel** qui, grâce à sa disponibilité, nous a guidées dans la conception de ce sujet ces derniers mois et nous a donné tout type de conseil pour arriver à perfectionner la mise en place de notre travail ainsi que **Félix Mora Camino.**

GLOSSAIRE

- **Ground Handling :**
  Le mouvement des passagers, des bagages et du cargo à travers des terminaux, ainsi que le déplacement des avions sur la plate-forme sont effectués grâce au « Ground Handling » (en français : Services d'assistance au sol).
  En fonction de la taille de l'aéroport ainsi que la politique de sa direction, ses activités peuvent être effectuées en collaboration entre l'autorité de l'aéroport, les compagnies aériennes et les agences spéciales de services d'assistance au sol.

- **Quick turnaround weight :**
  Masse maximale à l'atterrissage (MLW) pour laquelle les pneus du train atterrissage n'éclatent pas. Cette masse est spécifique pour chaque avion.

- **Quick turnaround time :**
  Temps qui doit s'écouler avant que l'avion ne reparte, quand sa masse à l'atterrissage antérieure a dépassé sa « Quick turnaround weight ». Le temps d'attente doit être environ 67 minutes et, un contrôle de la température des pneumatiques du train d'atterrissage doit être effectué avant le redécollage.

- **Big M**

  C'est une méthode de résolution des problèmes d'optimisation. Cette méthode consiste à choisir un M suffisamment grand pour que toutes les variables artificielles sortent de la base. On obtient ainsi la solution optimale au problème si une telle solution existe.

- **LP Solve**

  Logiciel qui permet de résoudre des problèmes de programmation linéaire. Ce sont des problèmes d'optimisation où la fonction objectif et les contraintes sont toutes linéaires.

TABLE DES MATIERES

INTRODUCTION GÉNERALE

Le sujet de notre projet, **Gestion des moyens au sol d'une plate-forme aéroportuaire**, se focalise sur les opérations d'escale d'un avion. Il existe une **problématique** en relation avec le temps des **escales d'aéroports**, qui affecte directement les passagers, les compagnies aériennes et les exploitants du même aéroport. Chaque fois qu'une tâche, qui est dans le plan d'escale, met plus de temps que celui planifié, son retard aura une répercussion sur la tâche suivante. L'accumulation de temps en plus peut être traduite par un retard global de l'opération.

Concernant **retard** dans le domaine du **trafic aérien,** c'est vu comme un des problèmes les plus gênants actuellement. Personne n'est sans savoir que le temps c'est de l'argent. C'est pour cela qu'une perte de temps ici implique une perte en efficacité de l'aéroport où s'effectue l'escale et ainsi, une perte de recettes.

Le **but** de notre projet est d'aborder tous les facteurs liés à la planification qui génèrent du retard (assignation des ressources et ordre d'exécution des différents tâches), les étudier séparément puis conjointement et, en partant de certaines hypothèses, proposer un **outil** qui aide à améliorer la gestion des opérations d'escale en **réduisant le retard**.

Étant donné qu'il y a toujours des facteurs externes imprévisibles qui pourraient induire un tel retard, nous ne pouvons pas garantir un retard nul dans tous les cas sinon une diminution.

I. ACTIVITES DE GROUND HANDLING

Les activités effectuées dans l'Aire de Trafic pour une opération d'escale pour un vol commercial sont présentées sur la figure0.

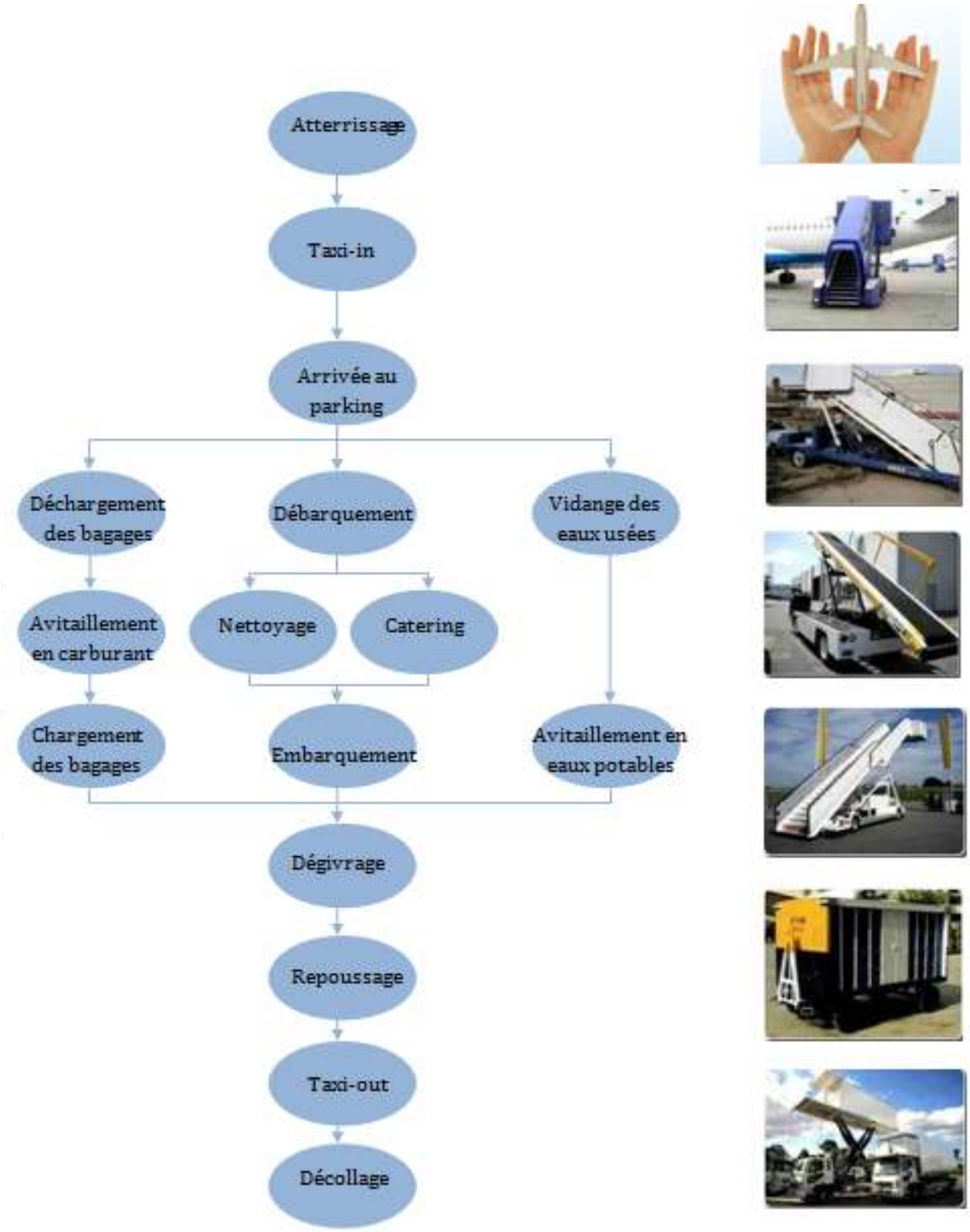

Figure0 : Activités d'escale (**Source :** *Integrating optimization and simulation to gain more efficient airport logistics)*

II. PRESENTATION DU PROBLEME

INTRODUCTION

La mondialisation de l'espace économique repose en bonne partie sur le transport de fret, mais également sur le mouvement de passagers. La croissance du trafic aérien est d'ailleurs hautement corrélée à la croissance du commerce international. Comme conséquence, l'étude des **capacités aéroportuaires** et **prévisions du trafic** ont eu lieu. Pour garantir que l'activité future estimée d'un aéroport pourra être supporte sans causer des retards, on peut amplifier les infrastructures de l'aéroport à long terme et optimiser la gestion au sol plus à court terme. Dans ce projet nous nous focalisons principalement sur la gestion au sol.

Dans la gestion des services d'assistance au sol d'une plate-forme aéroportuaire ou « Ground Handling[1] », le temps joue un rôle très important. Il existe un nombre d'activités à coordonner en une opération d'escale, qui sont liées à une série des contraintes en ordre d'exécution et en ressources disponibles. En plus, il peuvent apparaitre des aléas qui font qu'au niveau opérationnel le plan initialement prévu n'est plus réalisable.

Les différentes tâches sont effectuées de préférence simultanément pour diminuer le temps au sol et ainsi augmenter la productivité et rentabilité des avions. Les compagnies aériennes payent très cher les retards que subissent leurs avions en temps d'escales. Pour cette raison, qui fournit les opérations au sol (soit la même compagnie, soit l'autorité aéroportuaire ou une compagnie indépendant) est pressé pour réaliser un bon rapport temps/efficacité. Ce qui rend cette mission dure, est la coordination de tous les facteurs qui sont impliqués.

La planification des opérations d'escales permet de déterminer ce qui doit être réalisé et comment aboutir à cette réalisation. Il s'agit donc de séquencer les traitements dans un temps défini au préalable. Pendant le processus de planification nous devons nous pencher sur trois aspects différents. Le premier, fait référence au choix des activités qui doivent être réalisées ; le deuxième, correspond à quand réaliser l'exécution des tâches planifiées et le dernier aspect est lié à l'affectation des ressources disponibles.

Dans le problème que l'on présente ci-dessous, nous partons de l'hypothèse que le choix des activités lors de l'escale a déjà été effectué. Notre sujet nous impose que le problème d'ordonnancement des tâches et le problème de la tourné de véhicules ne doivent pas être traités séparément. Nous présenterons alors, par la suite, un modèle considérant ces deux problèmes intégrés. Ce traitement n'est pas courant en Recherche Opérationnelle, donc sa complexité est considérable.

A. METHODES DE RECHERCHE OPERATIONNELLE

La **Recherche Opérationnelle** (RO) peut être définie comme l'ensemble des méthodes et techniques rationnelles d'analyse et de synthèse des phénomènes d'organisation, utilisables pour élaborer de meilleures décisions. Cette *« aide à la décision »,* propose des modèles conceptuels pour analyser des situations complexes et permet choisir les plus efficaces.

Dans la pratique, les problèmes que la R.O. peut aider à résoudre sont : stratégiques (ex : dimensionnement d'une flotte de véhicules) ou opérationnelles (ex : ordonnancement).

Les étapes classiques de la recherche opérationnelle sont :

1. Définition du problème et rassemblement des données appropriées.
2. Formulation d'un modèle mathématique pour représenter le problème.
3. Élaboration d'une procédure informatique pour dériver des solutions au problème modélisé.
4. Validation du modèle et de ses hypothèses. S'il y a besoin on réalise un raffinement.
5. Préparation de l'application continue du modèle comme prescrite par la gestion.
6. Mise en application.

Les principales méthodes utilisées pour résoudre les problèmes d'optimisation sont :

- Algorithmes polynomiaux
- Programmation dynamique
- Processus stochastiques
- Simulation informatique
- Programmation linéaire et non linéaire
- Méthodes arborescentes
- Heuristiques et métaheuristiques

Nous appliquerons les trois premières étapes de la méthode de RO afin de trouver une « bonne » solution à la problématique exposée dans la partie antérieure.

Choix de modélisation :

Nous commencerons par déterminer les données dont nous aurons besoin dans le problème d'une plateforme aéroportuaire générique. Ensuite, nous définirons comment les paramétrer.
Il est nécessaire d'introduire une **notion temporelle** pour tenir compte de la séquence d'exécution pendant le « Ground Handling ». Nous utiliserons des paramètres de temps qui permettront le lien entre le problème d'ordonnancement et le problème de la tournée de véhicules.

B. PROBLEME D'ORDONNANCEMENT DES TACHES

1) Présentation

Le trafic des avions et les services d'assistance à l'escale qui ont lieu sur une plate-forme aéroportuaire doivent être planifiés pour obtenir sa bonne réalisation, tout en tenant en compte des ressources disponibles de manière optimale, accomplissant ainsi les objectifs de l'aéroport. Ce sous-problème de planification fait référence à l'ordonnancement.

Le problème envisagé consiste à décider de l'ordre d'exécution des activités en tenant compte de diverses priorités. Par exemple, dans le cas des procédures du « Ground Handling », la tâche « Débarquement passagers » doit être effectuée avant la tâche « Ravitaillement en combustible ». Ensuite, il faut définir le séquencement des tâches, leur commencement, et traduire le fait que parfois une tâche ne peut débuter avant la fin de l'exécution d'une autre.

2) Modélisation graphique du problème d'ordonnancement

Nous représentons par un graphique potentiel tâche dans la figure1 un exemple de suite de tâches pour illustrer le problème d'ordonnancement.

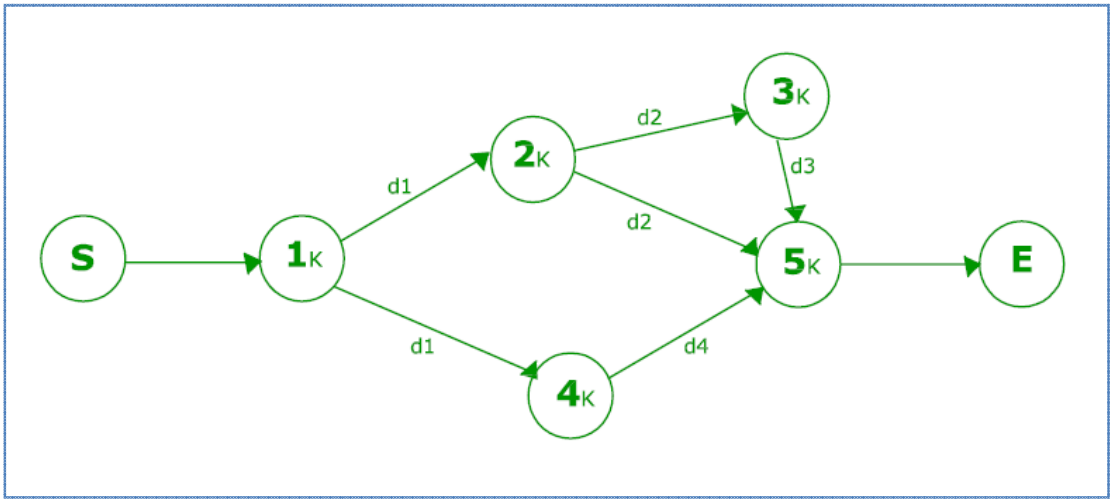

Figure 1-Graphe potentiel tâche
Exemple de contraintes de précédences pour les opérations d'escale pour un avion donné

Notons que:

- Ce graphique englobe les opérations d'escale à réaliser sur un même avion.
- Chaque nœud du graphe fait référence à une tâche de l'opération d'escale.
- Chaque arc du graphe représente une contrainte de précédence, avec pour (i, j) appartenant au graphe, i précède j.
- S symbolise « Start » (début de séquence) et E symbolise « End » (fin de séquence). Ces tâches ne sont pas à réaliser pour les véhicules de service. Ce sont des tâches fictives représentées seulement pour faciliter la compréhension d'un cycle d'escale d'un avion.

C. PROBLEME DE TOURNEE DE VEHICULES

1) Présentation

L'une des préoccupations des entreprises industrielles est d'améliorer l'efficacité de leur chaine logistique pour pouvoir organiser, au moindre coût, un meilleur service et fluidifier l'écoulement de leurs marchandises. Ainsi un élément fondamental de tout système logistique est la planification des réseaux de distribution par des flottes de véhicules.

Les problèmes de tournées de véhicules surviennent dans les situations où un ceux-ci sont utilisés pour servir un ensemble de demandes : ici ce sont les tâches à réaliser sur les avions. Ces tâches peuvent être effectuées par différents véhicules et nous devrons connaître quelle est la route optimale que doit prendre un véhicule bien choisi, pour minimiser le temps d'exécution, en supposant une quasi-homogénéité de la flotte de véhicules.

2) Modélisation graphique du problème de tournée de véhicules:

Nous représentons par un graphique en figure2, un exemple de tournée de véhicule.

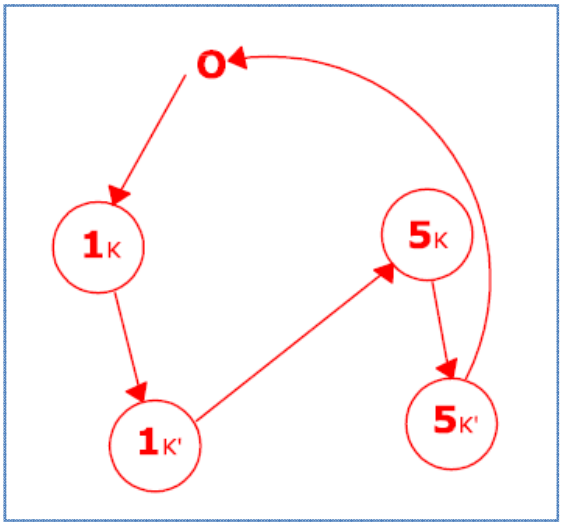

Figure 2-Exemple de tournée pour un véhicule donné

Notons que:

- Cette tournée convient à la capacité du véhicule
- Sa durée vaut : $D = d_{1k} + d_{1k'} + d_{5k} + d_{5k'} + temps_{attente} + temps_{transfert}$

- Le graphique représente une solution proposée pour un véhicule donné.
- O symbolise la base d'où le véhicule part pour commencer sa tournée et où il revient après exécution de sa dernière tâche.
- L'indice qui accompagne le numéro de la tâche fait référence à l'avion sur lequel la tâche est effectuée. (k et k' sont deux avions différents)

Le graphe de la figure3 comprend les deux problèmes intégrés pour montrer leur lien et permet d'illustrer la complexité du problème.
En effet, sur la figure 2 nous ne pouvons pas voir le fait que les différentes tâches sont soumises à des contraintes de précédence. C'est pourquoi nous représentons dans la figure 3 une solution de tournée d'un véhicule avec en parallèle les contraintes d'ordonnancement des tâches (ici effectuées sur deux avions).

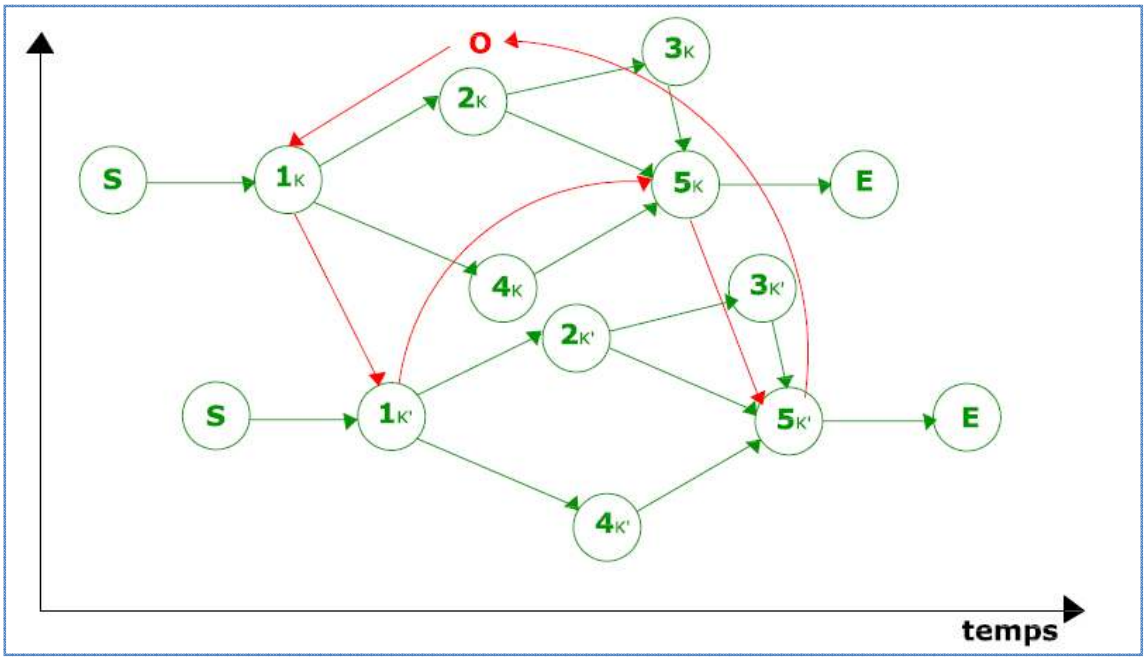

Figure 3-Exemple du modèle intégré.

Remarque : Les tâches 2,3 et 4 pour les avions k et k' seront effectuées par d'autres véhicules non représentés sur ce graphe par soucis de clarté.

III. MODELISATION MATHEMATIQUE DU PROBLEME

Nous commencerons par déterminer les données dont nous aurons besoin dans le problème d'une plateforme aéroportuaire générique. Ensuite, nous définirons comment les paramétrer.
Il est nécessaire d'introduire une **notion temporelle** pour tenir compte de la séquence d'exécution pendant le « Ground Handling ». Nous utiliserons des paramètres de temps qui permettront le lien entre le problème d'ordonnancement et le problème de la tournée de véhicules.

A. HYPOTHESES

1) Concernant la réalisation des taches

H1 : L'ensemble des tâches d'escale est le même pour tous les avions.
H2 : Chaque tâche est effectuée par un seul véhicule.
H3 : La durée d'une tâche ne dépend pas du type de véhicule qui l'effectue.

2) Concernant les véhicules

H4 : Chaque véhicule effectue une tâche à la fois.
H5 : Un même véhicule peut effectuer plusieurs types de taches.
H6 : Tous les véhicules ne sont pas compatibles avec tous les avions.
H7 : Un véhicule effectue un nombre limité de tâches avant de revenir à sa base.

B. ENSEMBLES ET PARAMETRES

Ensembles de variables	Description
K	Ensemble des avions indicé par k
T	Ensemble des tâches indicé par t
V	Ensemble des véhicules indicé par v

Tableau 1 - Ensembles utilisés dans le Modèle Mathématique

Paramètres	Description
HFE_K	Heure de fin d'escale prévue pour l'avion k
HDE_K	Heure de début d'escale de l'avion k au poste de stationnement
$d_{t,k}$	Durée de la réalisation de la tâche t sur l'avion k (H3)
$tr_{tk,t'k',v}$	Temps que le véhicule v met pour aller de la tâche t effectuée sur l'avion k à la tâche t' à effectuer sue l'avion k'
Nt(v)	Nombre de tâches que peut réaliser v avant de revenir à sa base

Tableau 2 - Paramètres du Modèle Mathématique

Soit m le nombre total des tâches à réaliser, nous posons les tâche 0 et m+1, des tâches fictives de début et de fin d'escale, de durée nulle.
On aura ainsi T = {0,1,….,m+1}.

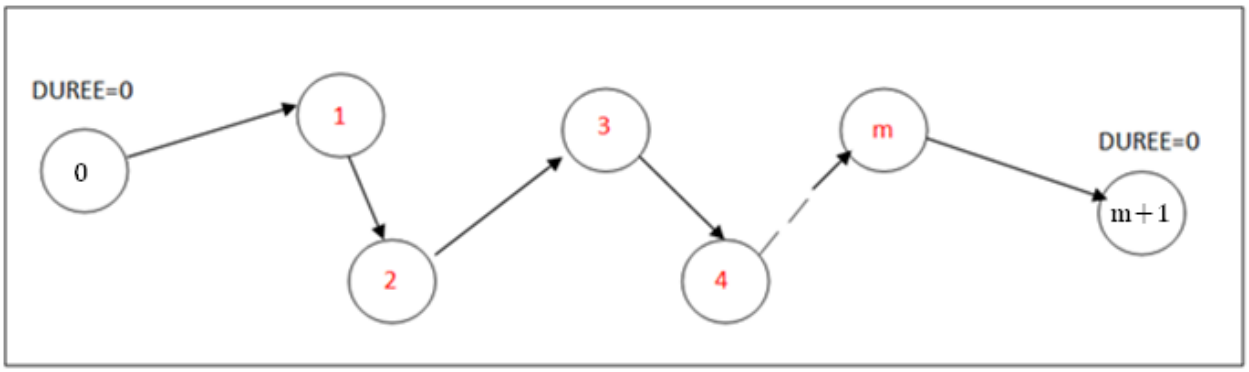

Figure 4 – Schéma du nombre de tâches que doivent s'effectuer sur l'avion qui fait l'escale.

C. MODELE MATHEMATIQUE

1) Variables de décision

On pose

- $c_{t,k}$ la date de commencement de la tâche t effectuée sur l'avion k.
- $\varepsilon_{t,k,v}$ = { 1 si le véhicule v effectue la tâche t sur l'avion k ; 0 sinon
- $X_{tk,t'k',v}$ = { 1 si le véhicule v utilise l'arc tk → t'k', i.e si v effectue la tâche t' sur l'avion k' juste après la tâche t sur l'avion k. ; 0 sinon
- $r_{t,v,k}$ = { 1 si le véhicule v effectue la tâche t sur l'avion k ; 0 sinon

Nous pouvons remarquer que $\varepsilon_{t,k,v}$ n'est pas une ''vraie'' variable de décision, en effet nous pouvons l'expliciter à l'aide de différents $X_{tk,t'k',v}$:

$$\varepsilon_{t\prime,k\prime,v} = \sum_{\substack{t\in T(v)\\ k\in K}} X_{tk,t'k',v}$$

Il en est de même pour $r_{t,v,k}$ que nous pourrions exprimer en fonction des X. car :

$$r_{t,v,k} = 1 \text{ et } r_{t\prime,v,k\prime} = 1 \rightarrow X_{tk,t'k',v}=1$$
$$X_{tk,t'k',v}=1 \rightarrow r_{t,v,k} = 1 \text{ et } r_{t\prime,v,k\prime} = 1$$

Cependant, nous les conservons toutes deux dans notre modèle à des fins de clarté des équations.

2) Contraintes

a) Contraintes de réalisation

C1 : On pose F(t), l'ensemble de véhicules pouvant effectuer la tâche t et Fk(t) l'ensemble des véhicules pouvant effectuer la tâche t sur l'avion k.
On note que Fk(t)C F(t).

Puis $F_k(t) = \{v \in V \mid v \text{ compatible avec l'avion k et } t \in T(v)\}$ C F(t)
Toutes les tâches sont réalisées pour chaque avion et par un seul véhicule (H1 et H2) se traduit par :

$$\sum_{v \in F_k(t)} \varepsilon_{t,k,v} = 1 \quad \forall k \in K, \forall t \in T$$

C2 : On pose V(k) l'ensemble des véhicules compatibles avec l'avion k.
Chaque véhicule effectue une tâche à la fois (H4) se traduit par :

$$\sum_{\substack{t' \in T(v) \\ k' \in K \\ v \in V(k) \cap V(k')}} X_{jk,j'k',v} = 1 \quad \text{et} \quad \sum_{\substack{t'' \in T(v) \\ k'' \in K \\ v \in V(k) \cap V(k'')}} X_{t'k'',jk,v} = 1 \qquad \forall k \in K, \forall v \in V(k), \forall t \in T(v)$$

b) Contraintes d'ordonnancement des tâches

C3 : Avec T = {0,1,…,m+1} on a

$$c_{0,k} \geq HA_k \quad \forall k \in K$$

C4 : <u>Contrainte de précédence :</u>

On définit $C=\{t, t') \in T^2 | t \text{ précède } t'\}$ l'ensemble des paires de tâches tel que, quel que soit (t,t') dans C, la tâche t' ne peut pas débuter avant la fin de la réalisation de la tâche t.
On pose alors la contrainte suivante :

$$c_{t',k} \geq c_{t,k} + \sum_{v \in F_k(t)} (d_{t,k} * \varepsilon_{t,k,v}) \quad \forall (t, t') \in T^2, \forall k \in K$$

c) Contraintes de tournée de véhicules

C5 : Chaque véhicule peut effectuer un nombre limité de tâches avant de retourner à sa base (H7).
Ainsi, nous avons, pour une mission du véhicule v :

$$\sum_{t \in T(v)} r_{t,v,k} = N_t(v)$$

Nous observons que l'adition de la variable r, en fixant le véhicule v, nous donne le nombre de tâches effectuées par v depuis son départ de la base jusqu'à son retour.

C6 : Contrainte temporelle

Pour pouvoir lier les tâches dans l'espace temporel, nous utilisons le paramètre $tr_{tk,t'k',v}$. Cela nous indique le temps qu'un véhicule met pour aller de la tâche t sur l'avion k à la tâche t' sur l'avion k'.

$$c_{t',k'} \geq \sum_{\substack{t \neq t' \\ k \neq k' \\ v \text{ compatible} \\ \text{avec } k,k' t,t'}} (c_{t,k} + d_{t,k,v} + tr_{tk,t'k',v}) * x_{t,k,t',k',v} \quad \forall (t,t') \in T^2, \forall (k,k') \in K^2$$

Notons que les contraintes C5 (faisant référence au séquencement) et C7 (faisant référence à la disponibilité de véhicules de service) nous donnent des bornes inferieures de commencement de réalisation de la tâche.

Parfois il existe une contrainte qui s'ajoute aux contraintes que nous avons considérait. Cette contrainte est connue comme ***Quick turnaround time*** et fait référence au temps que doit s'écouler avant que l'avion décolle de nouveau. Ce temps vient marquait pour l'intensité du freinage et la masse à l'atterrissage (Landing Weight). Si la LW est supérieur à la ***Quick turnaround weight*** (valeur spécifique pour chaque avion) l'avion devra rester au sol 67 minutes environ.

3) Fonction objectif

Le but de notre gestion de la plateforme aéroportuaire est de minimiser les retards des avions lors de l'escale, c'est-à-dire faire en sorte que l'écart entre l'heure de fin d'escale prévue pour l'avion et son heure de fin d'escale réelle sur la plateforme soit le plus petit possible, l'idéal étant qu'il soit nul.

Nous allons considérer l'ensemble des mouvements de la plateforme sur une journée, en supposant que chaque avion fait une seule escale.

On peut distinguer différentes façons de formuler la minimisation du retard des avions. Nous choisirons par la suite, celle qui nous semble la plus appropriée.

a) Minimisation du retard maximal

On pose alors :

$$Z1 = \text{Min}\left\{\max_{k \in K}(HFE_k - c_{m+1,k})\right\}$$

b) Minimisation du retard global

Cela revient à minimiser la somme des retards de tous les avions. On pose alors :

$$Z2 = \text{Min}\left\{\sum_{k \in K}(HFE_k - c_{m+1,k})\right\}$$

c) Minimisation du retard moyen

On pose alors :

$$Z3 = \text{Min}\left\{\frac{\sum_{k \in K}(HFE_k - c_{m+1,k})}{\text{Card}(K)}\right\}$$

IV. PROPOSITION D'UNE HEURISTIQUE

A. EXPLICATIONS

Dans l'algorithme qui suit, nous commençons par trier les couples (tâche,avion) dans l'ordre croissant dans le temps. Ensuite, pour chaque couple (t,a), nous cherchons parmi les véhicules capables d'effectuer la tâche t, celui qui peut être là le plus tôt afin de l'affecter à la position de l'avion a pour faire la tâche t. Ainsi, à chaque itération, nous enregistrons la ''vraie'' valeur (ie celle en considérant les contraintes de tournée de véhicules donc après d'éventuels retards) de date de début au plus tôt de la tâche t pour l'avion a.

Après avoir fait cela pour toutes les tâches de tous les avions sur la période choisie, nous calculons la somme des retards. C'est-à-dire que nous accumulons le temps de retard de chaque avion.

B. DONNEES

S = [[1] ; … ; [5]] — *%S[t]=liste des successeurs de la tâche t.*

t0 t1 tm
Ft = [[] ; [… ; v ; …] ; … [… ; …]] — *% Flottes de véhicules pouvant effectuer les tâches : F[t]=véhicules pouvant effectuer la tâche t.*

0 = base n
Tr = [[0 ; .. ; ….. ; …] ; … ; [… ; 0]] — *% Distance (temps) entre chaque position (parking de 1 à n et base)*

D(0,0) D(0,n) D(n,n)

OtaT = [(,) ; … ; (t,a) ; …] — *% Tableau d'ordonnancement au plus tôt des couples (tâche,avion) ordonnés par ordre croissant.*

Avions = [0 ; 1 ; … ; a ; …] — *%Avion[a] = renvoie à la structure de l'avion numéro a La structure avions[0] est vide pour simplifier l'utilisation des indices dans l'algorithme.*

Véhicules = [0 ; 1 ; … ; v ; …] — *%Véhicules[v] = renvoie à la structure du véhicule numéro v. La structure véhicules[0] est vide pour simplifier l'utilisation des indices dans l'algorithme.*

C. STRUCTURES

Type_avion
{
num_avion ;
position ; *% Numéro du parking*
HDE ; *% Heure de Début d'Escale*
HFE ; *% Heure de Fin d'Escale PREVUE*
Tdt ; *% Tableau de durée des tâches pour l'avion a*
Tct = [0 ; c_1 ; c_2 ; .. ; c_{m+1}=$HFE_{réelle}$] *% Tableau d'heure de commencement des tâches pour l'avion à initialiser avec les valeurs au plus tôt*
}

Type_véhicule
{
num_véhicule ;
position ; *%Initialisée là où il effectuera sa première tâche*
tâche ; *% Dernière tâche que le véhicule fait ou a fait*
h =0; *%Heure de disponibilité du véhicule initialisée à 0*
Nbt ; *% Nombre de tâches effectuées depuis la position 0*
Nt ; *% Nombre maximum de tâches que v peut faire avant de revenir à la base*
}

D. FONCTION SECONDAIRE

Cette fonction secondaire est une fonction de mise à jour des valeurs de commencement de tâches au plus tôt. Elle sera utilisée dans la fonction principale lors de l'actualisation de toutes nos valeurs. Elle permet de calculer, après mémorisation de la valeur définitive de date de début d'une tâche t, les nouvelles dates au plus tôt de toutes les tâches successeurs de t.

On n'actualise pas les dates de début des tâches successeurs, successeurs car elles seront calculées aux itérations suivantes dans la fonction principale.

```
Actualisation_c (a,t,S,avions,vehicules)=   %entrée de la fonction= avion et tâche considérés
{                                           matrice des successeurs, tableaux des données des
                                            avions et des véhicules
Int s ;
        Pour tout successeur s de t         %pour toute tâche dans S[t]
        {
        si Avions[a].Tct[s]< Avions[a].Tct[t]+ Avions[a].Tdt[t]
        Avions[a].Tct[s] =  Avions[a].Tct[t]+ Avions[a].Tdt[t];
        }
}
```

E. FONCTION PRINCIPALE

```
Int a, t, v ;                          %Avec a=num_avion,t=tâche, v=num_véhicule,
Int vd ;                               %vd : num_véhicule_disponible
Int Retard=0 ;                         % Retard= retard_total initialisé à 0

Int c,k,i,j ;                          % variables utilisées pour trouver les nouvelles dates de
                                       commencement de tâches

pour j=0 à taille de OtaT              %Pour chaque tâche prise dans l'ordre croissant de
{                                      temps

    (t,a) = OtaT[j] ;                  %Couple (tâche,avion) courant
    c=1000 ;                           %permet d'améliorer la date de début dès la 1ère boucle
    i=0 ;                              %indice du tableau de F[t]


    pour tout v dans Ft(t)             %Pour tout véhicule pouvant effectuer la tâche t
    {

    v=F[t][i] ;                        %véhicule courant

    k =
    max(Avions[a].Tct[t] ;                                    %contrainte de précédence
    Véhicules[v].h + Tr[Véhicules[v].position][Avions[a] .position]   %contrainte de disponibilité
                                                                de v


    si k < c                           %si v permet d'améliorer (de réduire) la date de
      {                                début au plus tôt de la tâche t

      c = k ;                          %mémorisation de date de début au plus tôt de t
      vd=v ;                           %mémorisation du numéro du véhicule disponible au
                                       plus tôt
      }

    i++ ;

    }
```

%A ce stade de l'algorithme, notre véhicule qui va effectuer la tâche t sur l'avion a est déterminé. Il est appelé vd pour véhicule disponible.
Nous allons maintenant procédé aux actualisations nécessaires.

Actualisations

```
Si Véhicules[vd].Nbt < Nt-1                    %le véhicule peut encore effectuer au moins une
        {                                      tâche avant de repasser à sa base

        Véhicules[vd].position = Avions[a].position ;
        Véhicules[vd].tâche = t ;
        Véhicules[vd].Nbt = Véhicules[vd].Nbt + 1 ;
        Véhicules[vd].h = c + Avions[a].Tdt[t] ;

        }

Sinon                                  %le véhicule doit aller à la
        {                              base à la fin de cette tâche

        Véhicules[vd].position = 0 ;
        Véhicules[vd].tâche = 0 ;
        Véhicules[vd].Nbt = 0 ;
        Véhicules[vd].h = c + Avions[a].Tdt[t] + Tr[0][Avions[a] .position] ;

        }

Avions[a].Tct[t] = c ;

Actualisation_c(a,t,S,avions,vehicules) ;

 j++ ;

}
```

Calcul du retard cumulé

```
pour a=1 au nombre total d'avion effectuant une escale sur la période donnée
{

Retard = Retard +(Avions[a].Tct[m+1] - Avions[a].HFE) ;     %addition des retards de
a++ ;                                                       chaque avion

}
```

F. ANNALYSE

Cet algorithme est une heuristique. Il n'est peut-être pas optimal, mais nous verrons à l'aide de jeux de données (voir partie suivante, page 20) qu'il est malgré tout puissant.

V. APPLICATIONS DE L'ALGORITHME

A. PREMIER JEU DE DONNEES

Nous présentons ci-dessous le premier jeu données que nous avons créé pour résoudre à petite échelle le problème de gestion d'une plate-forme aéroportuaire présenté dans le rapport. Nous considérons l'aéroport de la *figure 1.*

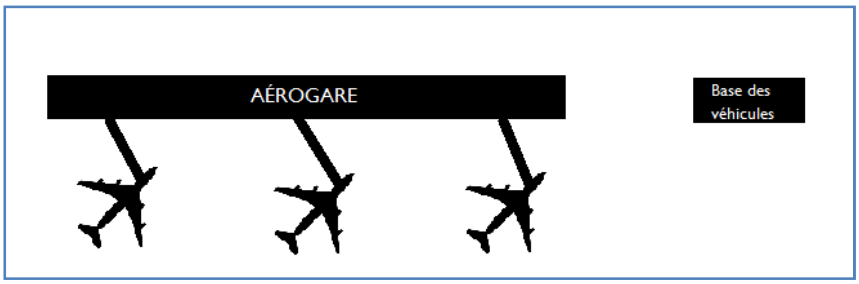

Figure 5 – Représentation schématique de l'aéroport

Les avions décriront l'escale suivante qui compte quatre tâches. Les tâches « *refueling* »(2) et « *catering* »(3) peuvent être réalisées en même temps.

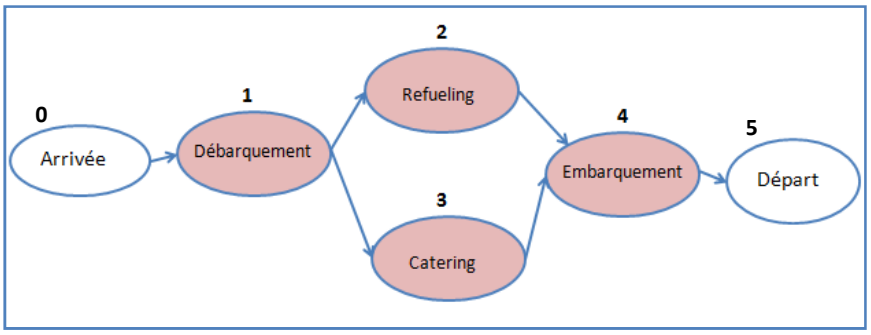

Figure 6 – Escale technique

Remarque : Nous supposons que l'avion repart directement quand les opérations de l'escale sont finies, il n'y a alors pas besoin des tâches: déchargement/chargement cargo.

Nous considérons que cinq avions feront escale dans cet aéroport sur une période donnée d'environ trois heures. Dans ce tableau nous avons les heures de début et de fin d'escale (en minutes) pour chaque avion et aussi, quel parking celui-ci occupera.

Avion	1	2	3	4	5
HDE	0	20	35	43	64
HFE	32	52	67	75	96
Position (nºparking)	1	2	3	1	2

Tableau 3 – Heures prévues de début et fin d'escale pour chaque avion et, son assignation au parking

Remarque 2: Nous supposons que le chemin critique pour effectuer toutes les opérations d'escale est de 32 minutes. Comme on peut observer sur le tableau ci-dessus.

Nous avons attribué à chaque tâche considérée une durée, en s'appuyant sur des données réelles.

Avion\Tsk	1	2	3	4
1	7	9	10	15
2	7	9	10	15
3	7	9	10	15
4	7	9	10	15
5	7	9	10	15

Tableau 4 – Durée de réalisation de chaque tâche pour chaque avion (**en minutes**)

Véhicule\Tsk	1	2	3	4
1	OK			OK
2			OK	
3		OK		

Tableau 5 – Véhicules qui peuvent réaliser les différentes tâches sur les avions

T_r (v)	0	1	2	3
0	0	3	3	3
1	3	0	3	3
2	3	3	0	3
3	3	3	3	0

Tableau 6 – Temps que les véhicules mettent pour aller d'une position à une autre (parking : 1, 2,3 ; base : 0)

Remarque : Pour simplifier la résolution du jeu de données nous considérons que les temps de transition sont tous égaux.

TACHE PRECENDENTE	TACHE
	1
1	2
1	3
2	4
3	

Tableau 7– Tableau des contraintes de précédence

Nous avons réalisé le **diagramme de gant** du jeu de données présenté sans introduire les contraintes de ressources. Sur l'axe des ordonnées nous avons représenté les cinq avions qui arrivent et sur l'axe des abscisses, nous avons représenté le temps en minutes.

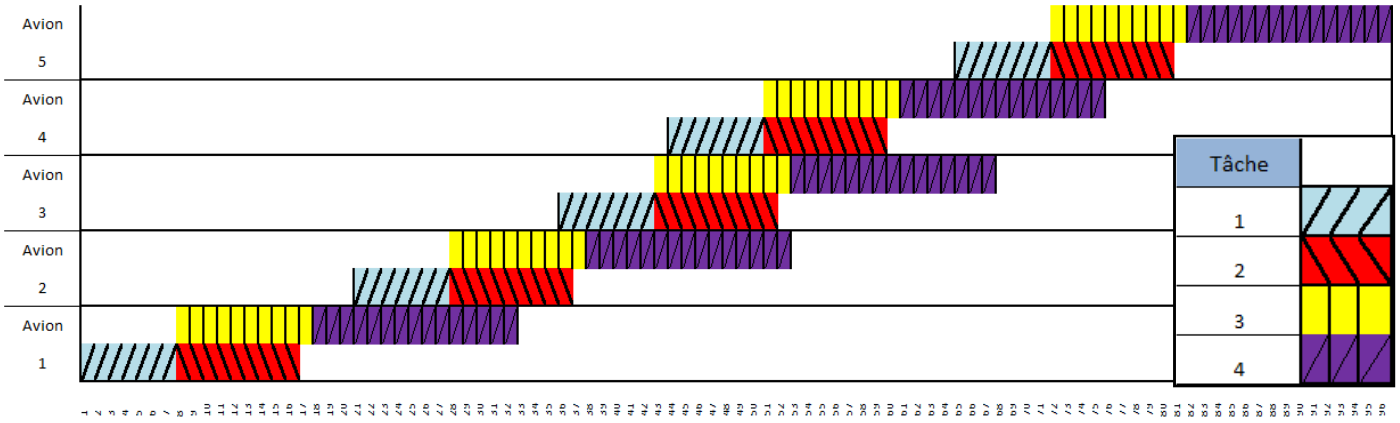

Figure 7 – Représentation du diagramme de gant **sans** considérer les **contraintes** de ressources

Ensuite, nous appliquons l'algorithme que nous avons proposé (voir ANNEXE) à ce premier jeu de données. Nous obtenons ainsi un tableau considérant le commencement de chaque tâche sur chaque avion, qui nous permettra de réaliser un diagramme de gant qui tiendra compte des contraintes de précédence et des ressources. Nous obtiendrons ainsi le retard total.

Application de l'algorithme:

Données :

S= [[1] ;[2 ;3] ;[4] ;[4] ;[5]] ;

F_t= [[] ;[1] ;[3] ;[2] ;[1]] ;

T_r= [[0 ;3 ;3 ;3] ;[3 ;0 ;3 ;3] ;[3 ;3 ;0 ;3] ;[3 ;3 ;3 ;0]] ;

OtaT= [(1,1) ; (2,1) ; (3,1) ; (4,1) ; (1,2) ; (2,2) ; (3,2) ; (1,3) ; (4,2) ; (2,3) ; (3,3) ; (1,4) ; (2,4) ; (3,4) ; (4,3) ; (4,4) ; (1,5) ; (2,5) ; (3,5) ; (4,5)] ;

Structures Avions:

Dans un tableau, qui est défini comme Avions [… ;…] (voir ALGORITHME page 16), nous avons entrée les données des cinq avions qui font l'escale. Ci-dessous nous présentons les valeurs pour chaque avion.

Remarque3 : Nous laissons la première case [0] du tableau vide pour facilité la manipulation des indices de tableau.

Avions [1]=
[
1 ;
1 ;
0 ;
32 ;
[0 ; 7 ; 9 ; 10 ; 15] ;
[0 ; 0 ; 7 ;7 ;17 ;32]
]

```
Type_avion
        {
        num_avion ;
        position ;
        HDE ;
        HFE ;
        Tdt ;
        Tct = [ 0 ; c1 ; c2 ; .. ; cm+1=HFEréelle]
        }
```

Figure 8 – Structure de type avion

Avions [2]=

[
2 ;
2 ;
20 ;
52 ;
[0 ; 7 ; 9 ; 10 ; 15] ;
[0 ; 20 ; 27 ;27 ;37 ;52]
]

Avions [3]=

[
3 ;
3 ;
35 ;
67 ;
[0 ; 7 ; 9 ; 10 ; 15] ;
[0 ; 35 ; 42 ;42 ;52 ;67]
]

Avions [4]=

[
4 ;
1 ;
43 ;
75 ;
[0 ; 7 ; 9 ; 10 ; 15] ;
[0 ; 43 ; 50 ;50 ;60 ;75]
]

Avions [5]=

[
5 ;
2 ;
64 ;
96 ;
[0 ; 7 ; 9 ; 10 ; 15] ;
[0 ; 64 ; 71 ;71 ;81 ;96]
]

Structures Véhicules:

Vehicules[1] =

{
1 ;
0 ;
0 ;
0 ;
0 ;
10
}

```
Type_véhicule
        {
        num_véhicule ;
        position ;
        tâche ;
        h =0;
        Nbt ;
        Nt ;
        }
```

Figure 9 – Structure de type véhicule

Vehicules[2] =
{
2 ;
0 ;
0 ;
0 ;
0;
10
}

Vehicules[3] =
{
3 ;
0 ;
0 ;
0 ;
0;
10
}

Rappel: $c_{t,a}$ est la date de début de la tâche t sur l'avion a exprimée en minutes

Nous savons que :
- $c_{0,a}$=HDE_a pour tout avion a.
-dans l'algorithme, pour le véhicule vd choisi (disponible),
$c_{t,a}$ = max{date de début au plus tôt de la tâche t sur l'avion a ;
date de disponibilité du véhicule vd}

Application de l'algorithme au jeu de données :

$c_{0,1} = 0$

OtaT[0] = (1,1)

t=1	a1	vd=v1

$c_{1,1}$ = max{0 ; 0} = 0
Actualisation :
v1 = { 1 ; **1** ;**1** ;**7**; **1** ; 10} *%7=0+7=début_t1+durée_t1*
Avions [1].Tct[1] = 0
Avions [1].Tct[2] = 7 } Actualisation des tâches successeurs
Avions [1].Tct[3] = 7 } de la tâche 1

OtaT[1] = (2,1)

t=2	a1	vd=v3

$c_{2,1}$ = max{ 7 ; 0} =7
Actualisation :
v3 = { 3 ; **1** ;**2** ;**16** ; **1** ; 10} *%16=7+9=début_t2+durée_t2*
Avions [1].Tct[2] = 7
Avions [1].Tct[4] = 16

OtaT[2] = (3,1)

t=3	a1	vd=v2

$c_{3,1}$ = max{ 7 ; 0} =7
Actualisation :
v2 = { 2 ; **1 ;3 ;17** ; **1** ; 10} *%17=7+10=début_t3+durée_t3*
Avions [1].Tct[3] =**7**
Avions [1].Tct[4] = 17

OtaT[3] = (4,1)

t=4	a1	vd=v1

$c_{4,1}$ = max{17 ; 7} = 17
Actualisation :
V1 = { 1 ; **1 ;4 ;32 ; 2** ; 10} *%32=17+15=début_t4+durée_t4*
Avions [1].Tct[4]=17
Avions [1].Tct[5]=32

Pour l'avion 1 (Avions [1]) nous obtenons :

Avions [1].**HFE** = 32 (prévue)
Avions [1].**HFE** $_{\text{réelle}}$ = Avions [1].**Tct [5]=** $c_{5,1}$= 32

Retard_ Avions [1]= Avions [1].HFE $_{\text{réelle}}$ **- Avions [1]**.HFE = 32 – 32 = **0 minutes**

Cette procédure sera la même pour les tâches et avions suivants (dans OtaT).
Les résultats sont présentés dans les tableaux qui suivent.

Tableaux des résultats:

a	t	v	v.Nbt	$c_{\text{tâche,avion}}$	Retard (minutes)
1	1	1	1	0	0
1	2	3	1	7	0
1	3	2	1	7	0
1	4	1	2	17	0

Avions [1].**Tct**= [0 ;0 ; 7 ; 7 ; 17 ;32]
Retard_ Avions [1]= Avions [1].HFE$_{\text{réelle}}$ - Avions [1].HFE = 32 – 32 = **0 minutes**

a	t	v	v.Nbt	$c_{tâche,avion}$	Retard (minutes)
2	1	1	3	35	15 (contrainte de ressources)
2	2	3	2	42	15
2	3	2	2	42	15
2	4	1	5	55	18

Avions [2].**Tct**= [0 ;35 ; 42 ;42 ;55 ; 70]
Retard_ Avions [2]= Avions [2].$HFE_{réelle}$ – Avions [2].HFE = 70 – 52 = **18 minutes**

a	t	v	v.Nbt	$c_{tâche,avion}$	Retard (minutes)
3	1	1	4	45	10
3	2	3	3	54	12
3	3	2	3	55	13
3	4	1	7	83	31

Avions [3].**Tct**= [0 ;45; 54 ;55 ; 83 ;98]
Retard_ Avions [3]= Avions [3].$HFE_{réelle}$ – Avions [3].HFE = 98 – 67 = **31 minutes**

a	t	v	v.Nbt	$c_{tâche,avion}$	Retard (minutes)
4	1	1	6	73	30
4	2	3	4	80	30
4	3	2	4	80	30
4	4	1	8	101	41

Avions [4].**Tct**= [0; 73 ; 80 ; 80 ; 101 ; 116]
Retard_ Avions [4]= Avions [4].$HFE_{réelle}$ – Avions [4].HFE = 116 – 75 = **41 minutes**

a	t	v	v.Nbt	$c_{tâche,avion}$	Retard (minutes)
5	1	1	9	119	55
5	2	3	5	126	55
5	3	2	5	126	55
5	4	1	10	136	55

Avions [5].**Tct**= [0; 119 ; 126 ; 126 ; 136 ; 151]
Retard_ Avions [5]= Avions [5].$HFE_{réelle}$ – Avions [5].HFE = 151 – 96 = **55 minutes**

Nous réalisons le nouveau **diagramme de gant**, maintenant en considérant les contraintes de ressources et de précédence.

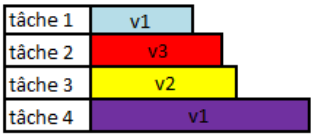

Figure 10 – Légende du diagramme

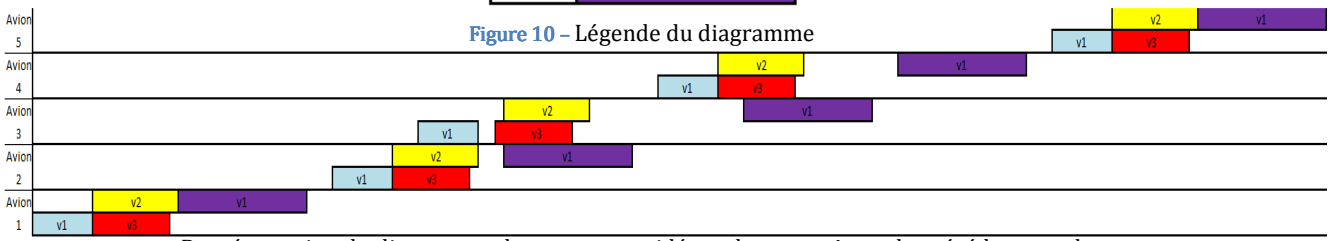

Figure 11 – Représentation du diagramme de gant **en** considérant les **contraintes** de précédence et de ressources

151

B. SECOND JEU DE DONNEES

Nous présentons un deuxième jeu données que nous avons créé pour montrer que quand nous augmentons la flotte de véhicules capables d'effectuer une même tâche, nous faisons face à un problème d'affectation de ressources. Dans ce cas nous avons aussi réduit la fenêtre de temps afin d'être confronté aux contraintes de tournée de véhicules. L'aéroport et l'escale que nous considérons sont ci-dessous.

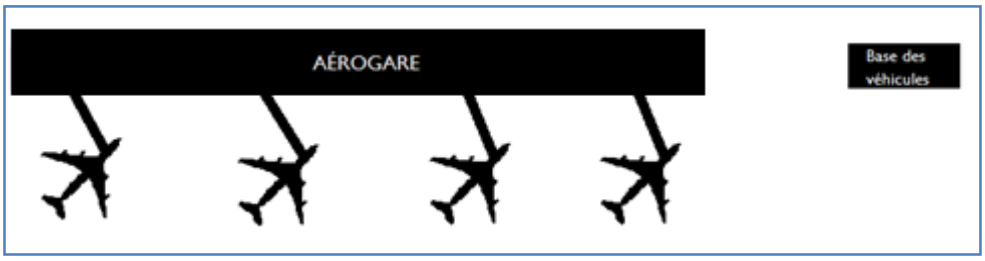

Figure 12 – Représentation schématique de l'aéroport

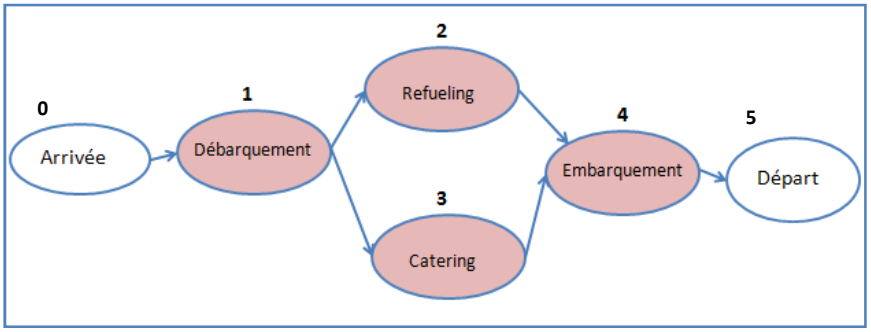

Figure 13 – Escale technique

Nous considérons ici aussi que cinq avions feront escale dans cet aéroport sur une période donnée d'environ une heure. Dans ce tableau nous avons les heures de début et de fin d'escale (en minutes) pour chaque avion et aussi, quel parking celui-ci occupera.

Avion	1	2	3	4	5
HDE	0	8	15	20	26
HFE	32	40	47	52	58
Position (nºparking)	1	2	3	4	1

Tableau 8 – Heures prévues de début et fin d'escale pour chaque avion et, son assignation au parking

Remarque 2: Nous supposons que le chemin critique pour effectuer toutes les opérations d'escale est de 32 minutes. Comme on peut observer sur le tableau ci-dessus.

Avion\Tsk	1	2	3	4
1	7	9	10	15
2	7	9	10	15
3	7	9	10	15
4	7	9	10	15
5	7	9	10	15

Tableau 9 – Durée de réalisation de chaque tâche pour chaque avion (**en minutes**)

Véhicule\Tsk	1	2	3	4
1	OK			OK
2	OK			OK
3			OK	
4		OK		

Tableau 10 – Véhicules qui peuvent réaliser les différentes tâches sur les avions

T_r (v)	0	1	2	3	4
0	0	3	3	3	3
1	3	0	3	3	3
2	3	3	0	3	3
3	3	3	3	0	3
4	3	3	3	3	0

Tableau 11 – Temps que les véhicules mettent pour aller d'une position à une autre (parking : 1, 2, 3, 4 ; base : 0)

Remarque : Pour simplifier la résolution du jeu de données nous considérons que les temps de transition sont tous égaux.

TACHE PRECENDENTE	TACHE
	1
1	2
1	3
2	4
3	

Tableau 12– Tableau des contraintes de précédence

Nous réalisons le **diagramme de gant** du jeu de données présenté sans introduire les contraintes de ressources et en considérant les contraintes de précédence. Sur l'axe des ordonnées nous avons représenté les cinq avions qui arrivent et sur l'axe des abscisses, nous avons représenté le temps en minutes.

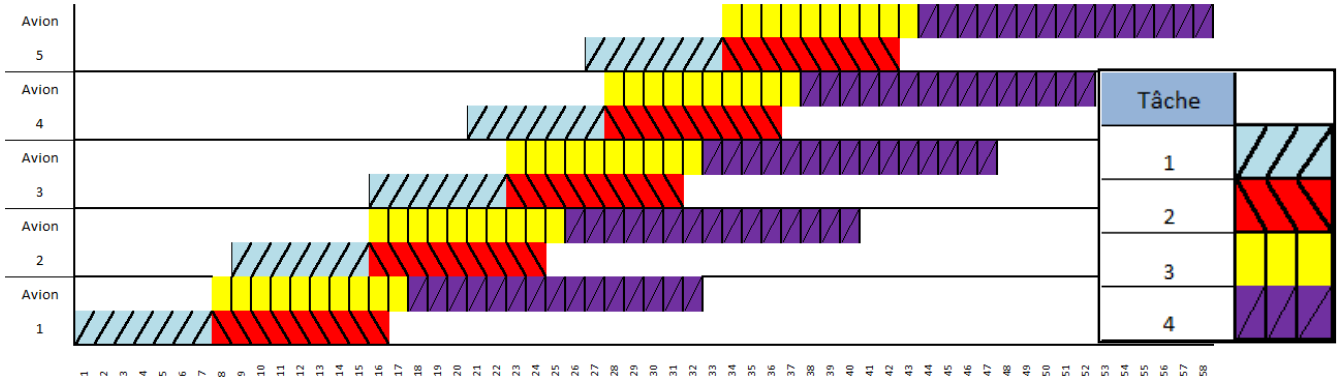

Figure 14 – Représentation du diagramme de gant **sans** considérer les **contraintes** des ressources

Ensuite, nous appliquons l'algorithme que nous avons proposé (voir ALGORITHME page 16) à ce deuxième jeu de données. Nous obtenons ainsi un tableau considérant le début de chaque tâche sur chaque avion, ce qui nous permettra de réaliser un diagramme de gant qui tiendra compte des contraintes de précédence et des ressources. Nous obtiendrons aussi le retard total.

Données :

S= [[1] ;[2 ;3] ;[4] ;[4] ;[5]] ;

F_t= [[] ;[1 ;2] ;[3] ;[4] ;[1 ;2]] ;

T_r= [[0 ;3 ;3 ;3 ;3] ;[3 ;0 ;3 ;3 ;3] ;[3 ;3 ;0 ;3 ;3] ;[3 ;3 ;3 ;0 ;3] 3 ;3 ;3 ;3 ;0]] ;

OtaT= [(1,1) ; (2,1) ; (3,1) ; (1,2) ; (2,2) ; (3,2) ; (1,3) ; (4,1) ; (1,4) ; (2,3) ; (3,3) ; (4,2) ; (1,5) ; (2,4) ; (3,4) ; (4,3) ; (2,5) ; (3,5) ; (4,4) ; (4,5)] ;

Structures Avions:

Dans un tableau, qui est défini comme Avions [… ;…] (voir ANNEXE), nous avons entrée les données des cinq avions qui font l'escale. Ci-dessous nous présentons les valeurs qu'on trouve case per case.

Remarque3 : Nous laissons la première case [0] du tableau vide pour suivre plus facilement la méthode algorithmique.

Avions [1]=

[
1 ;
1 ;
0 ;
32 ;
[0 ; 7 ; 9 ; 10 ; 15] ;
[0 ; 0 ; 7 ; 7 ; 17 ; 32] ;
]

Type_avion
{
num_avion ;
position ;
HDE ;
HFE ;
Tdt ;
Tct = [0 ; c_1 ; c_2 ; .. ; c_{m+1}=$HFE_{réelle}$]
}

Figure 15 – Structure de type avion

Avions [2] =

[
2 ;
2 ;
8 ;
40 ;
[0 ; 7 ; 9 ; 10 ; 15] ;
[0 ; 8 ; 15 ; 15 ; 25 ; 40]

]

Avions [3] =

[
3 ;
3 ;
15 ;
47 ;
[0 ; 7 ; 9 ; 10 ; 15] ;
[0 ; 15 ; 22 ; 22 ; 32 ; 47]
]

Avions [4] =

[
4 ;
4 ;
20 ;
52 ;
[0 ; 7 ; 9 ; 10 ; 15] ;
[0 ; 20 ; 27 ; 27 ; 37 ; 52]
]

Avions [5] =

[
5 ;
1 ;
26 ;
58 ;
[0 ; 7 ; 9 ; 10 ; 15] ;
[0 ; 26 ; 33 ; 33 ; 43 ; 58]
]

Structures Véhicules:

Vehicules[1] =

{
1 ;
0 ;
0 ;
0 ;
0 ;
10
}

Vehicules[2] =

{
2 ;
0 ;
0 ;
0 ;
0;
10
}

```
Type_véhicule
        {
        num_véhicule ;
        position ;
        tâche ;
        h =0;
        Nbt ;
        Nt ;
        }
```

Figure 16 – Structure de type véhicule

Vehicules[3] =
{
3 ;
0 ;
0 ;
0 ;
0;
10
}

Vehicules[4] =
{
4 ;
0 ;
0 ;
0 ;
0;
10
}

Rappel: $c_{t,a}$ est la date de début de la tâche t sur l'avion a exprimée en minutes

Nous savons que :
- $c_{0,a}$=HDE_a pour tout avion a.
-dans l'algorithme, pour le véhicule vd choisi (disponible) :
$c_{t,a}$ = max{date de début au plus tôt de la tâche t sur l'avion a ;
date de disponibilité du véhicule vd}

Application de l'algorithme au jeu de données :

$c_{0,1}$ = 0

OtaT[0]= (1,1)

t=1	**a**1	**v**1	**v1.**Nbt=1

$c_{1,1}$ = max{0 ; 0} = 0
Actualisation :
v1 = { 1 ; **1** ;**1** ;**7**; **1** ; 10} *%7=0+7=début_t1+durée_t1*
Avions [1].Tct[1] = 0
Avions [1].Tct[2] = 7 } Actualisation des tâches successeurs
Avions [1].Tct[3] = 7 } de la tâche 1

OtaT[1]= (2,1)

t=2	**a**1	**v**3	**v3.**Nbt=1

$c_{2,1}$ = max{7 ;0} = 7
Actualisation :
v3 = {3 ; **1** ; **2** ; **16** ; **1** ; 10}
Avions [1].Tct[2]= **7**
Avions [1].Tct[4]= 16

OtaT[2]= (3,1)

t=3	a1	v4	v4.Nbt=1

$c_{3,1}$ = max{7 ;0} = 7

Actualisation :

v4 = {4 ; **1 ; 3** ; **17** ; **1** ; 10}

Avions [1].Tct[3]= **7**

Avions [1].Tct[4]= 17

OtaT[3]= (1,2)

t=1	a2	v2	v2.Nbt=1

$c_{1,2}$= max{Avions[2].Tct[1]=8 ;0}=8

Actualisation :

v2 = {2 ; **2 ; 1** ; **15 ; 1** ; 10}

Avions [2].Tct[1]= 8

Avions [2].Tct[2] = 15 }

Avions [2].Tct[3] = 15 }

Actualisation des tâches successeurs de la tâche 1

OtaT[4]= (2,2)

t=2	a2	v3	v3.Nbt=2

$c_{2,2}$ = max{15 ;19}=19 *%19=16+3=heure_disponibilité_v3 + tr(pos_a1 ; pos_a2)*

Actualisation :

v3 = {2 ; **2 ; 2** ; **28** ; **2** ; 10}

Avions [2].Tct[2]=19

Avions [2].Tct[4]=28

OtaT[5]= (3,2)

t=3	a2	v4	v4.Nbt=2

$c_{3,2}$ = max{15 ;20}=20 *%20=17+3=heure_disponibilité_v3 + tr(pos_a1 ; pos_a2)*

Actualisation :

v4 = {2 ; **2 ; 3** ; **30** ; **2** ; 10}

Avions [2].Tct[3]=20

Avions [2].Tct[4]=30

OtaT[6]= (1,3)

t=1	a3	v2	v2.Nbt=2

$c_{1,3}$= max{15 ; 18} = 18

Actualisation :

v2 = {2 ; **3 ; 1** ; **25 ; 1** ; 10}

Avions [3].Tct[1]= 18

Avions [2].Tct[2] = 25

Avions [2].Tct[3] = 25

OtaT[7]= (4,1)

t=4	a1	v1	v1.Nbt=2

$c_{4,1}$ = max{17 ; 10} = 17

Actualisation :

V1 = {1 ; **1 ; 4** ; **32 ; 2** ; 10}

Avions [1].Tct[4]= 17

Avions [1].Tct[5] = 32

Pour l'avion 1 (Avions [1]) nous obtenons :

Avions [1].**HFE =** 32 (prévue)

Avions [1].**HFE $_{réelle}$ =** Avions [1].**Tct [5]=** $c_{5,1}$= 32

Retard_ Avions [1]= Avions [1].HFE $_{réelle}$ **- Avions [1].**HFE = 32 – 32 = **0 minutes**

A ce stade, l'escale du premier avion est finie. Nous avons donc le retard pour cet avion. Cette procédure sera la même pour les tâches et avions suivants (dans OtaT[]). Les résultats sont présentés dans les tableaux qui suivent.

Tableaux des résultats:

a	t	v	v.Nbt	$c_{tâche,avion}$	Retard (minutes)
1	1	1	1	0	0
1	2	3	1	7	0
1	3	4	1	7	0
1	4	1	2	18	1

Avions [1].**Tct**= [0 ; 0 ; 7 ; 7 ; 18 ; 33]

Retard_ Avions [1]= HFE réelle - Avions [1].HFE = 33 – 32 = **1 minute**

a	t	v	v.Nbt	$c_{tâche,avion}$	Retard (minutes)
2	1	2	1	8	0
2	2	3	2	19	4
2	3	4	2	20	5
2	4	1	3	35	10

Avions [2].**Tct**= [0 ; 8 ; 19 ; 20 ; 35 ; 50]

Retard_ Avions [2]= HFE réelle – Avions [2].HFE = 50 – 40 = **10 minutes**

a	t	v	v.Nbt	$c_{tâche,avion}$	Retard (minutes)
3	1	2	2	15	0
3	2	3	3	31	9
3	3	4	3	33	11
3	4	2	5	43	11

Avions [3].**Tct**= [0 ; 15; 31 ; 33 ; 48 ; 58]

Retard_ Avions [3]= HFE réelle – Avions [3].HFE = 63 – 47 = **11 minutes**

a	t	v	v.Nbt	$c_{tâche,avion}$	Retard (minutes)
4	1	2	3	25	5
4	2	3	4	43	16
4	3	4	4	46	19
4	4	1	4	56	19

Avions [4].**Tct**= [0 ; 25 ; 43 ; 46 ; 56 ; 71]
Retard_ Avions [4]= HFE réelle – Avions [4].HFE = 71 – 52 = **19 minutes**

a	t	v	v.Nbt	$c_{tâche,avion}$	Retard (minutes)
5	1	2	4	33	7
5	2	3	5	55	22
5	3	4	5	59	26
5	4	2	6	69	26

Avions [5].**Tct**= [0 ; 33 ; 55 ; 59 ; 69 ; 84]
Retard_ Avions [5]= HFE réelle – Avions [5].HFE = 84 – 58 = **26 minutes**

Nous réalisons le nouveau **diagramme de gant**, résultant de l'application de notre algorithme, en considérant les contraintes de ressources et de précédence.
La préférence est cédée à l'avion qu'arrive, afin de raccourcir le retard global de la journée.

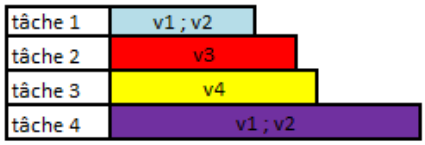

Figure 17 – Légende du diagramme

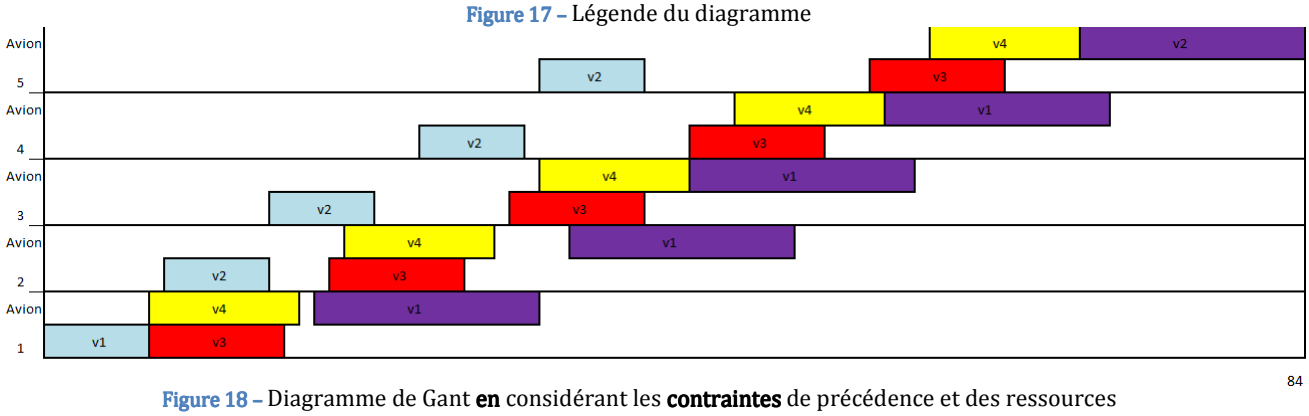

Figure 18 – Diagramme de Gant **en** considérant les **contraintes** de précédence et des ressources

C. ANALYSE DES RÉSULTATS

Pour voir la puissance de l'algorithme que nous proposons et pour analyser les résultats obtenus, nous présentons d'autres solutions possibles.

Référent au Jeu de données 1 :

Solution alternative :

Nous proposons une autre alternative d'ordonnancement et nous nous rendons compte que la solution résultant de notre algorithme n'est pas optimal pour réduire le retard total. L'alternative (diagramme de gant du dessus sur figure 19) fait la tâche 4 sur l'avion 3 avant la tâche 1 sur l'avion 4 et la tâche 1 sur l'avion 5 avant la tâche 4 sur l'avion 4. Cela représente un retard moindre sur les avions 3 et 5 et un retard plus grand sur l'avion 4. Cela nous fait au total un gain de 6 minutes.

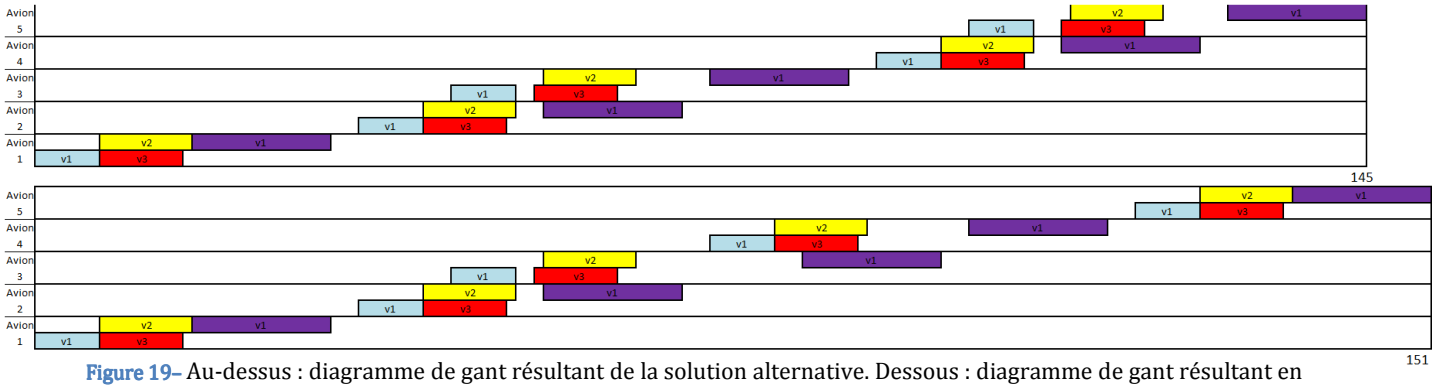

Figure 19– Au-dessus : diagramme de gant résultant de la solution alternative. Dessous : diagramme de gant résultant en appliquant notre algorithme.

Référent au Jeu de données 2 :

Première solution alternative (premier diagramme de gant sur la figure 20)

Ici, l'aéroport donne la priorité à l'avion qui est en place et non pas à celui qui arrive. Cela fait que l'ensemble de tâches sur les avions qui font l'escale à cet aéroport se réalise avec 19 minutes de plus par rapport au résultat de notre algorithme (dernier diagramme de gant sur la figure 20).

Deuxième solution alternative (deuxième diagramme de gant sur la figure 20)

L'aéroport décide que les véhicules pouvant faire deux tâches différentes sont spécialisés dans l'une d'elles. Comme par exemple dans notre « jeu de données 2 », nous supposons que le véhicule 1 est spécialisé dans la tâche 1 et le 2 dans la tâche 4.
Cette alternative prend 21 minutes de plus que celle de notre algorithme.

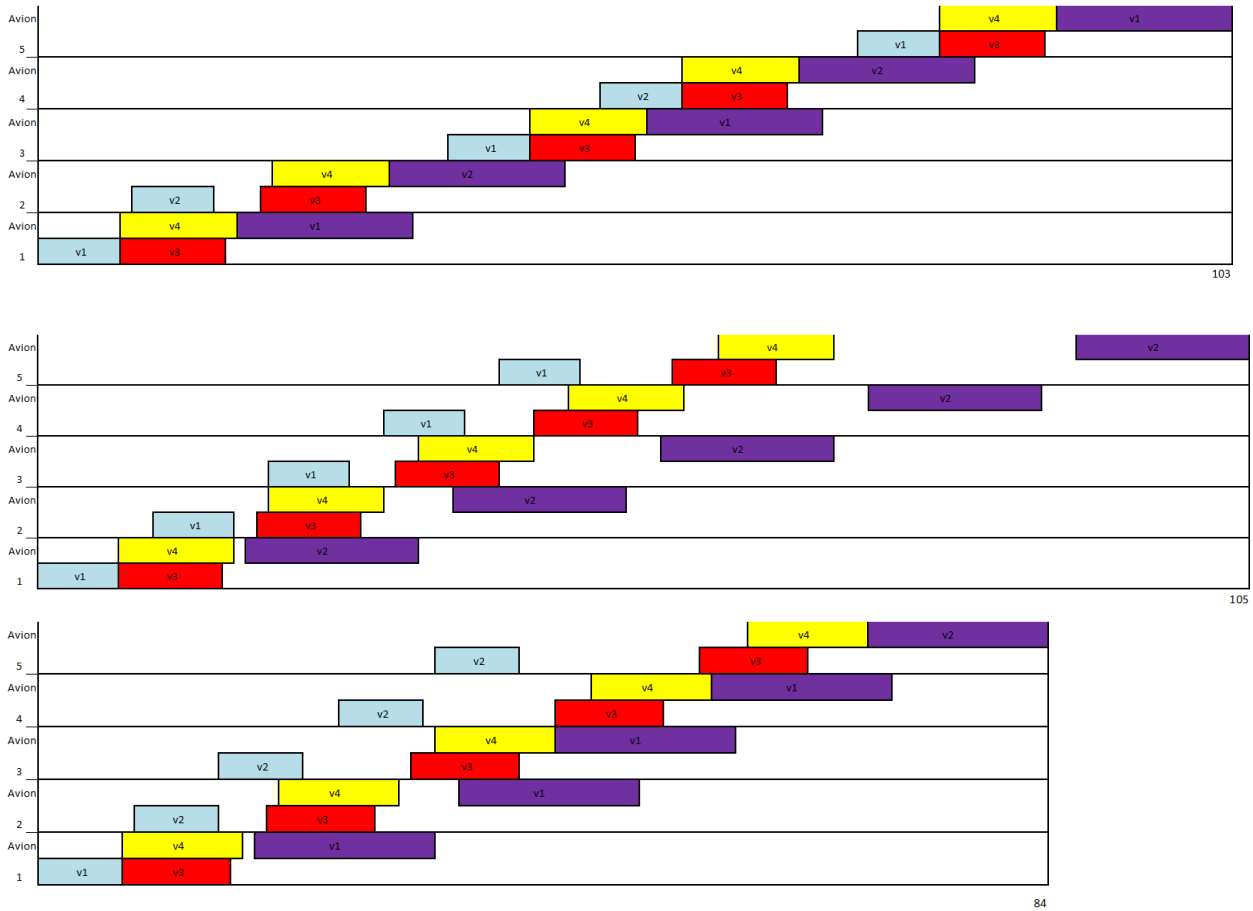

Figure 20 – Diagramme de Gant **en** considérant les **contraintes** de précédence et des ressources

Remarque : Notre jeu de données 2 nous donne un retard plus grand que pour le premier car nous avons fait en sorte d'ajouter des contraintes en resserrant la fenêtre de temps tout en gardant le même nombre d'avions. L'algorithme n'en devient pas moins efficace, il nous dit juste qu'avec tant de retard l'aéroport se doit d'avoir de nouveaux véhicules.

CONCLUSION

De nombreuses conditions de service doivent être satisfaites durant l'escale d'un avion. A partir du moment où il arrive à une porte terminale jusqu'au moment où il part sur son prochain vol. Minimiser le retard sur les opérations d'escale en considérant toutes les contraintes n'est pas un travail aisé. L'ordonnancement des tâches à réaliser conjointement avec l'allocation de ressources est un problème très complexe à gérer comme nous avons pu le vérifier grâce à ce projet.

En réalisant la modélisation mathématique du problème existant en relation avec le retard des opérations d'escale, nous avons trouvé avec un model non linéaire, qui traite en même temps la problématique liée à l'ordonnancement et celle liée à la tournée de véhicules. Le fait de considérer un modèle intégrant les deux problèmes au même temps nous a permis d'approcher de plus près la réalité dans le cadre d'escales d'aéroports. Cette méthode n'est pas courante en Recherche Opérationnelle, donc sa complexité est considérable.

Nous aurions pu générer un système de base de données aléatoire afin de faciliter la manipulation de l'algorithme.

Notre problème aurait pu également être linéarisé avec des méthodes mathématiques (« big M ») et nous aurions pu ainsi comparer nos résultats avec ceux optimisés (grâce à LPsolve et l'algorithme du simplexe). De cette façon, nous aurions pu affiner notre algorithme mais faute de temps nous laisserons au groupe de l'an prochain le soin de s'en occuper.

BIBLIOGRAPHIE

Bibliographie :

- Sylvina Calas, Eunice J. Trejo Bravo
  Gestion des moyens au sol d'une plate-forme aéroportuaire, 2009
- Antonin Combes, Beatriz Garrido
  Optimisation des moyens sols d'une plate-forme aéroportuaire, 2010
- David Bredström, (Linköping Institute of Technology), Mikael Rönnqvist, (Norwegian School of Economics and Business Administration)
  Combined vehicle routing and sheduling with temporal precedence and synchronization constraints
- Catherine Mancel
  Cours de Recherche Operationelle IENAC, 2009
- Johann Dréo, Alain Pétrowski, Patrick Siarry, Éric Taillard
  Métaheuristiques pour l'optimisation difficile, 2005
- Charles-Edmond Bichot
  Élaboration d'une nouvelle métaheuristique pour le partionnement de graphe, 2007
- Alain Dancel
  Support de cours ENAC, Langage C, 1995
- P. Lopez, P. Esquirol
  Ordonnancement, 1999

Web :

- http://www.jstor.org
- http://captainpilot.com
- http://www.geog.umontreal.ca/Geotrans/fr/ch3fr/conc3fr/ch3c4fr.html
- http://www.atmseminar.org
- Wikipedia

Logiciels utilisés :

- Dev C++
- Autocad

ANNNEXES : CODE DE L'ALGORITHME

I. FICHER .h

```
#ifndef _AL08T_H
#define _AL08T_H

#include <stdio.h>
#include <stdlib.h>


//définition des structures et pointeurs
typedef struct {
   int num_avion;
   int position;
   int HDE;
   int HFE;
   int tdt[5];
   int tct[6];
         }avion;

typedef avion * ptavion;

typedef struct {
   int num_vehicule;
   int position;
   int tache;
   int h;
   int nbt;
   int nt;
   }vehicule;

typedef vehicule * ptvehicule;

//fonction de remplissage des données des avions et des véhicules
void rempli_avions(ptavion avions[]);
void rempli_vehicules(ptvehicule vehicules[]);
```

```
//Fonction d'actualisation des dates de débuts des successeurs d'une tâche t
void actualisation_c(int a, int t, int S[][],
                     ptavion avions[],ptvehicule vehicules[]);

//Fonction maximum utilisée dans la recherche du véhicule disponible
int max(int a, int b);

#endif
```

II. FICHER .c

```
#include "al08t.h"

int main()
{

//Création des matrices et tableaux nécessaires au programme
int s[5][3];
int ft[5][3];
int tr[5][5];
int otat[20][2];

//pointeurs pour aller chercher les données des avions et des véhicules
ptavion avions[6];
ptvehicule vehicules[5];

//remplissage des données avions et véhicules
rempli_avions(avions);
rempli_vehicules(vehicules);

//Remplissage des matrices du programme:
// Matrice des successeurs
s[0][0]=1; s[0][1]=66; s[0][2]=66;
s[1][0]=2; s[1][1]=3; s[1][2]=66;
s[2][0]=4; s[2][1]=66; s[2][2]=66;
s[3][0]=4; s[3][1]=66; s[3][2]=66;
s[4][0]=5; s[4][1]=66; s[4][2]=66;
```

```
//Matrice des flottes de véhicules
ft[0][0]=66; ft[0][1]=66;ft[0][2]=66;
ft[1][0]=1; ft[1][1]=2;ft[1][2]=66;
ft[2][0]=3; ft[2][1]=66;ft[2][2]=66;
ft[3][0]=4; ft[3][1]=66;ft[3][2]=66;
ft[4][0]=1; ft[4][1]=2;ft[4][2]=66;

//Matrice des temps de transition
tr[0][0]=0; tr[0][1]=3; tr[0][2]=3; tr[0][3]=3; tr[0][4]=3;
tr[1][0]=3; tr[1][1]=0; tr[1][2]=3; tr[1][3]=3; tr[1][4]=3;
tr[2][0]=3; tr[2][1]=3; tr[2][2]=0; tr[2][3]=3; tr[2][4]=3;
tr[3][0]=3; tr[3][1]=3; tr[3][2]=3; tr[3][3]=0; tr[3][4]=3;
tr[4][0]=3; tr[4][1]=3; tr[4][2]=3; tr[4][3]=3; tr[4][4]=0;

//Matrice des couples (tache,avion) dans l'ordre croissant
otat[0][0]=1; otat[0][1]=1;
otat[1][0]=2; otat[1][1]=1;
otat[2][0]=3; otat[2][1]=1;
otat[3][0]=1; otat[3][1]=2;
otat[4][0]=2; otat[4][1]=2;
otat[5][0]=3; otat[5][1]=2;
otat[6][0]=1; otat[6][1]=3;
otat[7][0]=4; otat[7][1]=1;
otat[8][0]=1; otat[8][1]=4;
otat[9][0]=2; otat[9][1]=3;
otat[10][0]=3; otat[10][1]=3;
otat[11][0]=4; otat[11][1]=2;
otat[12][0]=1; otat[12][1]=5;
otat[13][0]=2; otat[13][1]=4;
otat[14][0]=3; otat[14][1]=4;
otat[15][0]=4; otat[15][1]=3;
otat[16][0]=2; otat[16][1]=5;
otat[17][0]=3; otat[17][1]=5;
otat[18][0]=4; otat[18][1]=4;
otat[19][0]=4; otat[19][1]=5;


//variables utilisées
int a,t,v,vd,retard=0;
int k,c;
int j,i;
```

```
//boucle pour chaque couple (tâche,avion) dans OtaT
   for (j=0;j<20;j++)
   {
      t=otat[j][0];
      a=otat[j][1];
      c=1000;
      i=0;

//boucle pour trouver le véhicule disponible au plus tôt
      do
      {
         k = max((avions[a]->tct)[t],
        (vehicules[ft[t][i]]->h +
        tr[vehicules[ft[t][i]]->position][avions[a]->position]));

        if (k<c)
        {
           c=k;
           vd=ft[t][i];
        }
        i++;

      }while (ft[t][i]<66);

//Actualisations en fonction des cas où le véhicule disponible doit
//retourner ou non à sa base
      if ((vehicules[vd]->nbt)<((vehicules[vd]->nt)-1))
      {
      vehicules[vd]->position=avions[a]->position;
      vehicules[vd]->tache=t;
      vehicules[vd]->nbt=vehicules[vd]->nbt + 1;
      vehicules[vd]->h = c + (avions[a]->tdt)[t];
      }

      if ((vehicules[vd]->nbt)>((vehicules[vd]->nt)-2))
      {
      vehicules[vd]->position=0;
      vehicules[vd]->tache=0;
      vehicules[vd]->nbt=0;
      vehicules[vd]->h = c + (avions[a]->tdt)[t] + tr[avions[a]->position][0];
      }
```

```
//Mémorisation de la nouvelle valeur de commencement au plus tôt de t
   (avions[a]->tct)[t]=c;

   actualisation_c(a,t,s,avions,vehicules);
   printf("\n");

int m;
for (m=0;m<6;m++)
{printf("\n avion %d", a);
   printf("tct %d", avions[a]->tct[m]);    }

   }

//Calcul de l'accumulation des retards pour chaque avion
for (a=1;a<6;a++)
{
printf("\n retard avion %d : %d", a, (avions[a]->tct)[5]- avions[a]->HFE);
retard = retard + (avions[a]->tct)[5]- avions[a]->HFE ;
}

//Affichage du retard
printf(" \n retard total : %d ", retard);
system("PAUSE");
return 0;

}
```

```
//fonction de remplissage des données des avions
void rempli_avions(ptavion avions[])
{
int k;

//Allocation de mémoire
   for (k=0; k<6; k++)
   {
   avions[k]=(avion*)malloc(sizeof(avion));
   }

(avions[0]->num_avion)=0;
(avions[0]->position)=0;
(avions[0]->HDE)=0;
(avions[0]->HFE)=0;
(avions[0]->tdt)[0]=0;(avions[0]->tdt)[1]=0;(avions[0]->tdt)[2]=0;
(avions[0]->tdt)[3]=0;(avions[0]->tdt)[4]=0;
(avions[0]->tct)[0]=0;(avions[0]->tct)[1]=0;(avions[0]->tct)[2]=0;
(avions[0]->tct)[3]=0;(avions[0]->tct)[4]=0;(avions[0]->tct)[5]=0;

(avions[1]->num_avion)=1;
(avions[1]->position)=1;
(avions[1]->HDE)=0;
(avions[1]->HFE)=32;
(avions[1]->tdt)[0]=0;(avions[1]->tdt)[1]=7;(avions[1]->tdt)[2]=9;
(avions[1]->tdt)[3]=10;(avions[1]->tdt)[4]=15;
(avions[1]->tct)[0]=0;(avions[1]->tct)[1]=0;(avions[1]->tct)[2]=7;
(avions[1]->tct)[3]=7;(avions[1]->tct)[4]=17;(avions[1]->tct)[5]=32;

(avions[2]->num_avion)=2;
(avions[2]->position)=2;
(avions[2]->HDE)=8;
(avions[2]->HFE)=40;
(avions[2]->tdt)[0]=0;(avions[2]->tdt)[1]=7;(avions[2]->tdt)[2]=9;
(avions[2]->tdt)[3]=10;(avions[2]->tdt)[4]=15;
(avions[2]->tct)[0]=0;(avions[2]->tct)[1]=8;(avions[2]->tct)[2]=15;
(avions[2]->tct)[3]=15;(avions[2]->tct)[4]=25;(avions[2]->tct)[5]=40;

(avions[3]->num_avion)=3;
(avions[3]->position)=3;
(avions[3]->HDE)=15;
```

```
(avions[3]->HFE)=47;
(avions[3]->tdt)[0]=0;(avions[3]->tdt)[1]=7;(avions[3]->tdt)[2]=9;
(avions[3]->tdt)[3]=10;(avions[3]->tdt)[4]=15;
(avions[3]->tct)[0]=0;(avions[3]->tct)[1]=15;(avions[3]->tct)[2]=22;
(avions[3]->tct)[3]=22;(avions[3]->tct)[4]=32;(avions[3]->tct)[5]=47;

(avions[4]->num_avion)=4;
(avions[4]->position)=4;
(avions[4]->HDE)=20;
(avions[4]->HFE)=52;
(avions[4]->tdt)[0]=0;(avions[4]->tdt)[1]=7;(avions[4]->tdt)[2]=9;
(avions[4]->tdt)[3]=10;(avions[4]->tdt)[4]=15;
(avions[4]->tct)[0]=0;(avions[4]->tct)[1]=20;(avions[4]->tct)[2]=27;
(avions[4]->tct)[3]=27;(avions[4]->tct)[4]=37;(avions[4]->tct)[5]=52;

(avions[5]->num_avion)=5;
(avions[5]->position)=1;
(avions[5]->HDE)=26;
(avions[5]->HFE)=58;
(avions[5]->tdt)[0]=0;(avions[5]->tdt)[1]=7;(avions[5]->tdt)[2]=9;
(avions[5]->tdt)[3]=10;(avions[5]->tdt)[4]=15;
(avions[5]->tct)[0]=0;(avions[5]->tct)[1]=26;(avions[5]->tct)[2]=33;
(avions[5]->tct)[3]=33;(avions[5]->tct)[4]=43;(avions[5]->tct)[5]=58;

}


//fonction de remplissage des données des véhicules
void rempli_vehicules(ptvehicule vehicules[])
{
int k;

//Allocation de mémoire
   for (k=0;k<5;k++)
   {
   vehicules[k]=(vehicule*)malloc(sizeof(vehicule));
   }

(vehicules[0]->num_vehicule)=0;
(vehicules[0]->position)=0;
(vehicules[0]->tache)=0;
```

```
(vehicules[0]->h)=0;
(vehicules[0]->nbt)=0;
(vehicules[0]->nt)=0;

(vehicules[1]->num_vehicule)=1;
(vehicules[1]->position)=1;
(vehicules[1]->tache)=0;
(vehicules[1]->h)=0;
(vehicules[1]->nbt)=0;
(vehicules[1]->nt)=10;

(vehicules[2]->num_vehicule)=2;
(vehicules[2]->position)=2;
(vehicules[2]->tache)=0;
(vehicules[2]->h)=0;
(vehicules[2]->nbt)=0;
(vehicules[2]->nt)=10;

(vehicules[3]->num_vehicule)=3;
(vehicules[3]->position)=1;
(vehicules[3]->tache)=0;
(vehicules[3]->h)=0;
(vehicules[3]->nbt)=0;
(vehicules[3]->nt)=10;

(vehicules[4]->num_vehicule)=4;
(vehicules[4]->position)=1;
(vehicules[4]->tache)=0;
(vehicules[4]->h)=0;
(vehicules[4]->nbt)=0;
(vehicules[4]->nt)=10;

}
```

```
//Fonction d'actualisation des dates de débuts des successeurs d'une tâche t
void actualisation_c(int a, int t, int s[5][3],
                     ptavion avions[],ptvehicule vehicules[])
{
int k=0;
int succ=s[t][0];
   do
   {
   if ((avions[a]->tct)[succ]<((avions[a]->tct)[t]+(avions[a]->tdt)[t]))
      {
      (avions[a]->tct)[succ]=(avions[a]->tct)[t]+(avions[a]->tdt)[t];
      }
   k++;
   succ=s[t][k];
   } while (succ<66);
}



//Fonction maximum utilisée dans la recherche du véhicule disponible
int max(int a, int b)
{

if (a<b) return b;
else return a;

}
```