Planification de mission pour un véhicule aérien autonome

Elodie Chanthery

▶ **To cite this version:**

Elodie Chanthery. Planification de mission pour un véhicule aérien autonome. Automatique / Robotique. Ecole nationale superieure de l'aeronautique et de l'espace, 2005. Français. NNT : . tel-00174631

HAL Id: tel-00174631
https://theses.hal.science/tel-00174631

Submitted on 24 Sep 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

présentée en vue de l'obtention du titre de

Docteur
de
l'École Nationale Supérieure
de l'Aéronautique et de l'Espace

Spécialité : **Systèmes**

Planification de Mission pour un Véhicule Aérien Autonome

soutenue par

Elodie CHANTHERY

Membres du Jury :

Jean-Henri Llareus	Professeur, École Nationale Supérieure de l'Aéronautique et de l'Espace Président
Dominique Luzeaux	Directeur du Centre Technique des Systèmes d'Information, DGA Rapporteur
Erik Sandewall	Professeur, Université de Linköping, Suède Rapporteur
Philippe Morignot	Responsable scientifique, Axlog Ingénierie Examinateur
Magali Barbier	Ingénieur de recherche, ONERA/DCSD Co–Directeur de thèse
Raja Chatila	Directeur de recherche, LAAS/CNRS Co–Directeur de thèse
Jean-Loup Farges	Ingénieur de recherche, ONERA/DCSD Membre invité

Remerciements

Et quand vient le temps de dire "merci", alors je sais que ça y est, j'ai fini. Je sais qu'il faut tourner la dernière page, écrire la dernière ligne. C'est un peu de moi qui part, car bientôt ce travail ne m'appartiendra plus. Ce sera "ma thèse". Je la regarderai de loin, comme on voit une personne qu'on a bien connue jadis, mais qui à présent n'est plus qu'un vague souvenir. Alors qu'elle a été ma vie pendant trois ans, maintenant c'est un peu fini entre nous. J'espère que ce n'est pas la même chose pour les gens que j'ai eu le plaisir de côtoyer lors de ces trois années. Et qu'il me restera non pas un manuscrit sur une étagère, que je regarderai parfois, mais bien plus encore : des rencontres enrichissantes, des personnes de confiance, des amis et même un compagnon de vie.

Ma thèse ne serait rien sans Magali Barbier, qui m'a si bien entourée lors de mon stage de PFE DEA d'abord, puis lors de ma thèse. J'essaierai de ne pas oublier tes conseils de rédaction si pertinents. Merci pour ta disponibilité et pour ta spontanéité. Tu as été une aide tellement précieuse pendant ces trois ans de travail sur le plan professionnel, mais aussi sur le plan personnel.

Un grand merci à Jean-Loup Farges pour s'être autant impliqué dans cette thèse. Merci pour tous tes conseils sur l'orientation de ma thèse et ton appui concernant ma formation d'enseignante.

Enfin merci à Raja Chatila d'avoir accepté d'être mon directeur de thèse.

Je tiens à remercier tout particulièrement Messieurs Dominique Luzeaux et Erik Sandewall pour avoir accepté de relire ce manuscrit et pour toutes les remarques enrichissantes qu'ils ont pu me faire. Merci à Messieurs Philippe Morignot et Jean-Henri Llareus pour avoir bien voulu participer à ce jury.

Pour pouvoir travailler sereinement, j'ai eu la chance d'être accueillie au Département Commande des Systèmes et Dynamique du vol du Centre de Toulouse de l'Office National d'Etudes et de Recherches Aérospatiales. Un grand merci à Claude Barrouil, directeur du département, pour l'intérêt qu'il a porté à mes travaux, son aide et ses encouragements. Merci à Jean-François Gabard, qui s'occupe si bien des doctorants de son unité de recherche.

De nombreuses personnes m'ont aidées lors de mon passage à l'ONERA. Je tiens à remercier (mais que les autres ne se fâchent pas parce qu'ils ne sont pas cités) Patrick Fabiani, Catherine Tessier, Michel Corrège, Michel LLibre, Michel Lemaître, Gérard Verfaillie, Jean-Pierre Chrétien, Carsten Doll.

Merci aussi à Liliane, pour toute l'aide qu'elle m'a apportée pendant ces trois ans.

L'ONERA ne serait rien sans la grande famille des thésards, qui sont pour beaucoup devenus de vrais amis.

Merci donc à Olivier V., qui de près comme de loin a été mon plus grand soutien pendant ces trois ans. Tu as été un guide, un appui, un mentor, (oserais-je dire "une échelle" !! :)) bref, tout ce qu'on pouvait souhaiter de mieux pour réussir une thèse. Merci à Sontsada pour m'avoir accompagnée et guidée pendant deux années. Merci pour nos grandes discussions d'hiver (et d'été !), nos rires, et aussi pour m'avoir donné un si bon exemple d'organisation ! Merci à Sophie pour m'avoir supportée dans tous les sens du terme lors des derniers mois de ma thèse. Merci aux voisins aussi, Stéphanie, Jean-Baptiste et Vincent, car ce n'est pas si facile d'avoir deux pipelettes à côté de son bureau. Plus loin dans le couloir, un grand merci à Nico le tortionnaire si gentil et aux deux compères Charles et Olivier qui comptent maintenant énormément dans ma vie. Encore plus loin, Sylvain et son humour. Bien sûr je n'oublie pas Elodie (vive les sorties moto !!), Yannick et Guilhem, de la grande salle. Merci aussi à Florent ; je n'oublierai pas notre fin de stage en musique, nos week-ends foireux sous la pluie, nos vacances si réussies, nos premiers de l'an chaotiques... A tous, merci pour nos midis si sympas, pour les coinches infinissables, les soirées chez les uns et autres et tout le reste. Et bien sûr, rendez-vous à Nanterre !

Merci aussi à tous les autres, thésards et stagiaires passés et présents, qui font vivre l'ONERA jour après jour.

Cette famille si sympa a ses conjoints et conjointes, merci aussi à eux pour toutes nos soirées. Un remerciement spécial à Raoudha pour nos grandes parlottes sur nos vie. Un autre aussi à l'élan joyeux qui se reconnaîtra !!! (vive l'optimisme et l'auto-satisfaction !!). J'ai gagné deux amies pour longtemps, je l'espère.

Merci aussi aux amis de Toulouse et d'ailleurs, Thierry, Michaël, Noelle, Xavier, Fanny, car ils ont toujours plus confiance en moi que moi même... et ça, c'est bien agréable !

A l'ONERA, c'est aussi des moments de détente le midi, alors merci à Nicole qui nous voit tous les jours. J'aimerais aussi dire un grand merci au groupe de danse de l'AS, car c'était aussi bien agréable de vous retrouver le mardi midi. Nicolas, Olivier, Mira, et les autres, merci.

Je n'aurai certainement aucune autre occasion pour leur dire, alors j'espère qu'ils comprendront bien. Un immense merci à ma famille. Maman, papa, merci pour m'avoir accordé tout votre amour et votre confiance depuis un certain 6 septembre 1979. Merci maman, car c'est toi qui m'a portée lors de mes études scientifiques, merci pour tes encouragements et ton aide quand j'en avais besoin. Merci papa, car la confiance aveugle que tu me portes me donne de la force quand je doute. J'espère que tu es fier de ta fille ! Merci à mon grand frère, Sébastien, qui même de loin m'a toujours donné son soutien. Merci aussi à mamie et papi. Et j'en profite ici, car je n'ai pas souvent l'occasion de vous le dire, je vous aime.

Enfin, Fabien, merci à toi. Tu me demandes souvent pourquoi nous avons fait une thèse. Voilà une raison non scientifique, mais tellement valable à mes yeux : à faire de nous un couple, à créer une famille je l'espère. L'avenir le dira ! ...

Table des matières

Remerciements **iii**

Notations **ix**

Introduction **1**

1 Le problème de planification de mission **5**
1.1 Introduction 5
1.2 Littérature sur les modèles et méthodes de planification 7
1.2.1 Problèmes présentant des similitudes avec le problème de planification de mission 8
1.2.2 Les problèmes de sélection 11
1.2.3 Traitement de l'incertitude : les processus markoviens 16
1.2.4 Conclusions 17
1.3 Hypothèses et variables de la planification 19
1.3.1 Planification déterministe en ligne 19
1.3.2 Les variables d'état du problème 19
1.3.3 Les variables d'action du problème 20
1.3.4 Relation entre deux états successifs 20
1.4 Formalisation du problème 21
1.4.1 Description haut niveau : réalisation d'un objectif 22
1.4.2 Description bas niveau 23
1.4.3 Synthèse : entrées et sorties du problème 25
1.4.4 Quelques hypothèses et propriétés associées 26
1.4.5 Complexité 27
1.5 Application à une mission d'observation 28
1.5.1 Présentation du problème 28
1.5.2 Discrétisation de l'espace 30
1.5.3 Description haut niveau 30
1.5.4 Description bas niveau 33
1.6 Conclusion 40

2 Les algorithmes de planification **43**
2.1 Introduction 43
2.2 Rappel sur l'algorithme A^* et ses dérivés 44
2.3 Le cadre algorithmique 46

2.3.1 L'algorithme de base : vue d'ensemble . . . 46
2.3.2 Recherche d'une première solution admissible . . . 48
2.3.3 Traitement des contraintes et définition du critère . . . 50
2.3.4 Optimisation du critère . . . 52
2.4 Les méthodes de l'algorithme . . . 54
2.4.1 Les méthodes d'évaluation du coût du nœud courant à un nœud fin 54
2.4.2 Les méthodes d'élagage . . . 65
2.4.3 Les méthodes de rangement de la liste des nœuds pendants . . . 65
2.5 Quelques interprétations sur l'exemple applicatif . . . 66
2.6 Conclusion . . . 68

3 Intégration de la planification dans une architecture embarquée 69
3.1 Introduction . . . 69
3.2 Littérature sur les liens entre planification et exécution . . . 70
3.2.1 Les niveaux d'autonomie . . . 70
3.2.2 Calcul de plans hors ligne ou en ligne . . . 72
3.2.3 Les architectures des systèmes autonomes . . . 74
3.2.4 L'environnement logiciel ProCoSA . . . 83
3.2.5 Conclusions . . . 85
3.3 Fonctionnalités de l'architecture de contrôle . . . 86
3.3.1 Effectuer la mission . . . 86
3.3.2 Réagir aux aléas . . . 86
3.4 Niveaux de l'architecture hiérarchique . . . 87
3.4.1 Niveau 3 : gestion de la mission et de son environnement . . . 89
3.4.2 Niveau 2 : gestion du plan . . . 91
3.4.3 Niveau 1 : gestion de trajectoire . . . 94
3.4.4 Niveau 0 : guidage . . . 96
3.5 Utiliser ProCoSA pour coder et activer l'architecture . . . 98
3.5.1 L'organisation des tâches de raisonnement . . . 98
3.5.2 Le suivi des tâches physiques . . . 103
3.5.3 Conclusions . . . 104
3.6 Criticité temporelle et critères pour le calcul de replanification . . . 106
3.6.1 Evaluation du plan courant dans un nouveau contexte . . . 106
3.6.2 Classification des problèmes de replanification . . . 108
3.6.3 Critères d'évaluation pertinents selon la classe du problème . . . 109
3.6.4 Application à l'exemple . . . 111
3.6.5 Discussion . . . 112
3.7 Conclusion . . . 113

4 Les tests 115
4.1 Présentation . . . 115
4.2 Les scénarios . . . 116
4.2.1 Le système de drone . . . 116
4.2.2 Les missions et les environnements . . . 117
4.2.3 Planification globale et replanifications . . . 120

4.3 Résultats et Analyses 126
4.3.1 Mise en œuvre des tests 126
4.3.2 But de l'analyse 128
4.3.3 Moyens employés 128
4.3.4 Choix d'une méthode pour chaque contexte de replanification . . 129
4.3.5 Choix d'une méthode unique pour tous les cas de replanification 144
4.4 Conclusions 146

5 Conclusions et Perspectives **149**

ANNEXES **161**

A Rappels sur la construction et l'optimisation dans les graphes **161**
A.1 Construction de graphe 161
A.2 Optimisation dans les graphes 162

B Algorithme du simplexe **169**

C Principaux projets de recherche utilisant les drones **177**

D Scénarios tests et Résultats **179**
D.1 Récapitulatif des scénarios 179
D.2 Mission 1 180
D.3 Mission 2 190
D.4 Mission 3 200
D.5 Mission 4 210

Notations

Le problème de planification de mission

N		ensemble de nœuds
W		partition de N
$W_1, \ldots, W_e$		ensemble constituant W
S		relation de succession sur les W_i, étendue à la relation de succession sur les nœuds
W_1		sous-ensemble correspondant à l'état initial
W_e		sous-ensemble correspondant à l'état final
$\overline{W_e}$		nombre maximum de nœuds dans l'ensemble W_e
P		ensemble des objectifs à essayer de réaliser au cours de la mission
$\overline{P}$		cardinal de P
o		un objectif
$W_{s(o)}$		sous-ensemble indiquant le début du traitement de o
$W_{e(o)}$		sous-ensemble indiquant la fin du traitement de o
$W_{r(o)}$		sous-ensemble indiquant la fin de la réalisation de o
n_k		un nœud du sous-ensemble W_k
n_1		premier nœud du plan
$t_k, t_{k_{min}}, t_{k_{max}}$		date d'arrivée dans le sous-ensemble W_k, date au plus tôt, date au plus tard
t_1		date du début de la planification
r_k		vecteur ressource à l'entrée dans le sous-ensemble W_k
r_1		vecteur ressource au début de la planification
R_o		récompense obtenue en réalisant l'objectif o
R_e		fonction des coûts sur les ressources à valeurs dans $\mathbb{R}^+$
Q	=	séquence de sous-ensembles $W_{\pi(i)}$, résultant de la planification
$W_{\pi(1)}, \ldots, W_{\pi(q)}$		
$W_{\pi(i)}$		un sous-ensemble appartenant au plan
J		critère
E_o		ensemble des objectifs réalisés par le plan
A		ensemble des actions possibles
$A_{i,j}$		sous-ensemble de A indiquant les actions autorisées entre W_i et W_j
a_k		action mise en œuvre pour atteindre n_k
(n_k, n_{k+1})		arc
$L(n_k, n_{k+1}, a_{k+1})$		longueur de l'arc (n_k, n_{k+1}) s'il est parcouru avec l'action a_{k+1}
Δ		vecteur des durées pour parcourir chaque arc de l'itinéraire

$\Delta_{n_k,n_{k+1}}^{a_{k+1}}$	composante du vecteur Δ : durée par parcourir l'arc (n_k, n_{k+1}) avec l'action a_{k+1}
v, v_{min}, v_{max}	vitesse, vitesse minimale, vitesse maximale
r^s	$s^{ième}$ composante du vecteur ressources r
r_{min}^s	valeur minimale de r^s
$\underline{n}$	vecteur de nœuds
$\underline{a}$	vecteur d'actions
$\Delta_{\underline{n}}^{\underline{a}}$	version condensée des trois données : vecteurs des durées, des nœuds et des actions pour un itinéraire
$\sigma_{i,e}$	$\in \mathbb{R}^+$, coût résiduel d'un nœud courant à un nœud fin
K	nombre maximum de nœuds dans chacun des ensembles $W_{s(o)}$ ou $W_{e(o)}$
T	nombre maximum de nœuds dans $W_{r(o)}$

L'application : mission d'observation militaire pour un véhicule aérien autonome

La mission

TOW	point de décollage ; $W_1 = \{TOW\}$
LW	ensemble des points d'atterrissage ; $W_e = LW$
ENU	ensemble des points d'entrée en zone ennemie
EXU	ensemble des points de sortie de la zone ennemie
ENO_o	ensemble des points d'entrée de la zone objectif o
EXO_o	ensemble des points de sortie de la zone objectif o
TW	ensemble des points de transmission des données
DG	zone de danger

Les fonctions : ressources, récompenses, coûts

$A \backslash B$	A privé de B
$\sum_{k/...}$	somme sur les k tels que ...
$\lfloor x \rfloor$	partie entière de x
r^1	probabilité d'être vivant au temps courant
r^2	masse du véhicule au temps courant
pm	probabilité d'existence de la menace
pt	probabilité de détection du drone par la menace
D_{min}	durée avant laquelle le drone a moins de 1 % de risque d'être repéré par une menace existante
D_{max}	durée après laquelle le drone a plus de 99 % de risque d'être repéré par une menace existante
pa	la probabilité de touche d'un drone repéré
m	menace
S_M	ensemble des menaces

E_m	ensemble des expositions dues à la traversée d'une menace m
e_m	exposition à la menace m
I_m	l'ensemble des arcs (n_k, n_{k+1}) de l'itinéraire interceptant le menace m pendant l'exposition e_m
$L(n_k, n_{k+1}\|m)$	longueur de l'arc à l'intérieur de la menace m
$h(k, k+1)$, h	altitude
ρ	densité de l'air
S	surface du plan aérodynamique principal du drone
Cd_0	coefficient de traînée à portance nulle
K	coefficient de traînée induite
m_a	masse du drone supposée constante sur un arc
m_0	masse du drone au décollage
m_1	masse du drone au début de la planification
g	gravité supposée constante et égale à g_0
C_{SR}	consommation spécifique (en $kg.s^{-1}.N^{-1}$)
G_o	gain maximum possible pour l'objectif o
P_{Obs_o}	probabilité d'observation de l'objectif o
P_{sys}	prix du drone et de sa charge utile
P_{carbu}	prix du carburant par unité de masse consommée
P_{vol}	prix d'une unité de temps vol
$\mathbf{C}$	vecteur de même dimension que r, ici $\mathbb{R}^2$, de composante P_{sys} et P_{carbu}
L_p distance de parcours σ	la pente entre l'origine et l'extrémité de l'arc
Cl	coefficient de portance de valeur maximum Cl_{max}
T_{max}	poussée maximum
VNE	vitesse (Never Exceed) à ne jamais dépasser

Les algorithmes de planification

$\mathbf{X}^\top$	transposée de $\mathbf{X}$
$G(N, A)$	un graphe
N	ensemble des nœuds
A	ensemble des arcs reliant les nœuds
f_i	potentiel associé au nœud n_i ou potentiel ajusté
g_i	potentiel du nœud
h_i	estimation du coût du nœud courant n_i à un nœud destination
μ	constante donnant l'importance de h_i
$listeP$	liste des nœuds pendants ou frontière de recherche
$listeQ$	liste des nœuds développés
$\hat{u}$	premier élément dans la $listeP$
$S(\hat{u})$	ensemble des successeurs de $\hat{u}$
E_o^i	ensemble des objectifs o réalisés avant n_i
I	critère partiel de n_1 à n_i, de valeur optimale $\hat{I}$
I_Δ	gradient de I

$BORNE$	valeur optimale du critère pour l'itinéraire de n_1 à un nœud fin
n_i^a	nœud n_i atteint avec l'action a
Ψ^s	fonction de pénalisation pour la ressource r^s
H	fonction Heavyside
α	entier positif pair
ϵ	facteur de précision
$\mathbf{A}$, $\mathbf{b}$	matrices définissant les contraintes dans le simplexe
h	évaluation du critère pour un itinéraire du nœud courant à un nœud de fin
H_1	méthode de calcul de h ne prenant pas en compte l'utilisation des ressources
H_i	méthode de calcul de h prenant en compte l'utilisation des ressources jusqu'au nœud courant
H_r	méthode de calcul de h basée sur la résolution d'un problème relaxé
H_W	méthode de calcul de h utilisant la structure particulière à deux niveaux du graphe
$A^{v,u}$	vecteur A pour le superarc (v, u)
V^v	vecteur V pour le supernœud v
$V[i]$	ième composante de V
$P(\hat{u})$	ensemble des prédécesseurs du supernœud $\hat{u}$
E_1	méthode d'élagage utilisée si h est un minorant du critère
E_2	méthode d'élagage utilisée si h n'est pas un minorant du critère
R_1	méthode de rangement en profondeur ordonnée guidée par g
R_2	méthode de rangement en profondeur ordonnée guidée par $g + h$
R_3	méthode de rangement au meilleur g d'abord
R_4	méthode de rangement au meilleur $g + h$ d'abord

Intégration de la planification dans une architecture embarquée

C , C'	contexte de mission
$J_{C'}$	évaluation du critère dans le nouveau contexte
J_{MAX}	valeur de $\mathbb{R}$ telle que le risque d'échec de la mission est considéré comme trop élevé
δ_r	durée de calcul au-delà de laquelle le module de planification doit fournir un plan
δ_1	durée accordée pour améliorer le plan dans le cas de replanification critique
$J^1 \mathcal{I} J^2$	J^1 et J^2 sont indifférents
$J^2 \mathcal{P} J^1$	J^2 est préféré à J^1
δ_2	durée accordée pour améliorer le plan dans le cas de replanification nécessaire
δ_3	durée accordée pour améliorer le plan dans le cas de replanification éventuelle

Les tests

Evt_1	événement “fuite de carburant”
Evt_2	événement “changement dans la carte des zones de danger”
Evt_3	événement “changement dans la carte des zones objectif”
D, M, F	les trois instants de replanification : début, milieu et fin de mission
J^i	critère généralisé
f	fréquence d’occurrence

Introduction

Autonomie : Capacité à agir par soi-même en se donnant sa propre loi.

Actuellement, les missions effectuées par les systèmes de drone sont prédéfinies. Les drones suivent un plan de vol tout comme le ferait un avion commercial équipé d'un système de gestion de vol. Le plan de vol est une séquence de points de passage définis par leurs coordonnées et associés à des actions à effectuer sur la charge utile. Le suivi de cette séquence permet de réaliser des opérations qui sont les objectifs de la mission. Dans ces appareils, il n'y a aucune autonomie au sens de la prise de décision.

Le développement de méthodes de conduite et de décision pour l'exécution de missions par des véhicules inhabités est devenu un défi important en termes d'autonomie et de planification, notamment pour des missions dangereuses ou de longue durée. En particulier, les missions avec des communications limitées entre les opérateurs et le véhicule requièrent de celui-ci une autonomie décisionnelle. En effet, le véhicule doit non seulement suivre le plan courant, mais aussi réagir de façon autonome à des événements survenant en cours de mission et invalidant le plan. A l'occurrence d'un tel événement, une fonction de planification embarquée doit être capable de générer un nouveau plan en résolvant en ligne un problème de planification de mission.

Le problème de planification de mission consiste à sélectionner et ordonner le meilleur sous-ensemble d'objectifs parmi l'ensemble des objectifs à réaliser et à déterminer les dates de début et de fin de réalisation de chaque objectif, en maximisant les récompenses obtenues lors de la réalisation des objectifs et en minimisant des critères correspondant à la consommation des ressources, tout en respectant les contraintes sur les ressources et la mission.

Comme beaucoup de problèmes de planification, la planification de mission peut être considérée comme un problème de sélection. Les objectifs sont associés à des récompenses dont les valeurs varient selon l'importance de chaque objectif. La planification doit choisir un sous-ensemble d'objectifs à réaliser en temps et ressources limités. Les systèmes de planification existants sont pour la plupart inadaptés pour résoudre de tels problèmes : ils traitent un problème où le but est une conjonction d'objectifs et échouent si le but n'est pas atteint.
Par ailleurs, la sélection d'objectifs ne résout pas entièrement le problème de planification de mission. En effet, la sélection se base souvent sur un modèle simplifié pour les ressources du problème et ne tient pas compte des différentes manières de réaliser

un même objectif. La plupart du temps, une solution pratique est obtenue en combinant la sélection d'objectifs, la planification conjonctive classique et l'ordonnancement de tâches dans une approche multi-niveaux : chaque niveau définit le problème pour l'algorithme de planification du niveau inférieur. Cette approche présente deux inconvénients : l'utilisation de différents modèles à différents niveaux pose le problème de l'optimalité du plan global et la détection de problèmes sans solution admissible au niveau le plus bas induit une nouvelle planification au niveau le plus haut, ce qui peut entraîner des itérations coûteuses en temps de calcul.
Contrairement à la planification réalisée lors de la préparation de mission, la planification en ligne est caractérisée par le fait que le temps mis pour trouver un plan est un des principaux critères de jugement de la qualité d'une méthode.

L'objectif de ce travail est de formaliser le problème de planification de mission, puis de proposer et d'évaluer une méthode de planification en ligne qui puisse à la fois réaliser une sélection d'objectifs, prendre en compte les ressources de manière réaliste et s'intégrer dans une architecture de gestion de mission embarquée.

Ce manuscrit suit l'organisation suivante :

Des travaux ayant des objectifs semblables à ceux de cette thèse ont été menés depuis une cinquantaine d'années. Les modèles et méthodes de planification s'y référant ainsi que les liens entre planification et exécution seront exposés au fur et à mesure du document.

Le premier chapitre présente le cadre développé pour pouvoir **décrire formellement une large classe de problèmes de planification de mission**. Le but de ces problèmes est d'obtenir des récompenses en réalisant un certain nombre d'objectifs. Les récompenses doivent compenser le coût de l'utilisation des ressources lors de la réalisation des objectifs. L'idée principale est d'utiliser la notion d'abstraction pour décrire à un haut niveau la réalisation des objectifs. Ce haut niveau est commun à tous les problèmes et n'est donc pas spécifique au domaine d'application du problème. Le formalisme s'applique à un cadre large de contraintes sur le temps ou sur les ressources. Les ressources traitées ne sont pas nécessairement décomposables sur les actions effectuées pour réaliser la mission. Les ressources et les contraintes sont spécifiques au domaine d'application. Elles sont entièrement explicitées dans un second niveau de description. La capacité du cadre à s'appliquer sur une mission réaliste est démontrée en considérant une mission militaire d'observation pour un véhicule aérien dans un environnement tridimensionnel, dynamique, incertain et dangereux. L'environnement inclut une zone ennemie où le véhicule effectue des opérations qui sont les objectifs de la mission. Les contraintes de la mission sont induites par les objectifs, l'environnement et le véhicule.

Le deuxième chapitre propose plusieurs **méthodes de planification en ligne pour résoudre le problème de planification de mission** décrit dans le chapitre 1. Basées sur des outils connus de l'intelligence artificielle, notamment des algorithmes de recherche arborescente et des heuristiques d'évaluation, ces méthodes permettent de sélectionner un ensemble pertinent d'objectifs à réaliser et de prendre en compte les contraintes de réalisation et l'utilisation des ressources de manière réaliste.

Le troisième chapitre propose une **intégration de la fonction de planification dans une architecture embarquée**. Après une étude des liens entre planification et exécution, une architecture hybride hiérarchique dédiée au problème de la réalisation d'une mission par un véhicule autonome est décrite. La dernière partie du chapitre propose d'évaluer la criticité temporelle du calcul de planification en vue de déterminer les critères de performance utilisés pour évaluer les méthodes de planification lors des tests. Cette étude montre que les méthodes algorithmiques doivent être évaluées selon le contexte de replanification dans lequel elles sont mises en œuvre.

Dans le quatrième chapitre, les **algorithmes sont testés pour différents contextes de replanification de mission** de façon à mettre en valeur les avantages et les limites de la méthode générale de recherche présentée dans ce travail. Des replanifications menées dans des contextes variés illustrent la pertinence de l'utilisation de telle ou telle méthode dans chaque contexte.

Le dernier chapitre dégage les **apports de cette thèse et des conclusions**, une **discussion sur les hypothèses** formulées pour ce travail et enfin diverses **perspectives** concernant les différentes parties développées dans cette thèse.

1 Le problème de planification de mission

Résumé : Ce chapitre définit et formalise le concept de "mission". Les particularités du problème de planification par rapport à un problème de planification dit "classique" sont mises en avant. Après une étude des modèles et méthodes les plus pertinents dans le cadre de la planification de mission, les hypothèses et les variables du problème de planification sont posées. On développe ensuite un cadre permettant de décrire formellement une large classe de problèmes de planification de mission. L'idée principale est d'utiliser la notion d'abstraction pour décrire à un haut niveau la réalisation des objectifs. Ce haut niveau est commun à tous les problèmes et n'est donc pas spécifique au domaine d'application du problème. La capacité du cadre à s'appliquer sur une mission réaliste est démontrée en considérant une mission militaire d'observation pour un véhicule aérien.

1.1 Introduction

Une *mission* est effectuée en réalisant des *actions* : des actions de mouvements, des actions sur l'environnement, des collectes d'informations... Les ressources utilisées pendant la réalisation des actions sont disponibles en quantités limitées. Pour une grande partie des problèmes réels, les ressources sont consommables : leur niveau baisse à mesure que la mission se déroule.

Une mission a un *début* et une *fin* entre lesquels toutes les actions sont réalisées. Une mission a également un *objet* qui est l'ensemble des objectifs à essayer de réaliser.

Chaque objectif peut être réalisé de différentes manières. Chaque manière correspond au choix et au séquencement d'un ensemble d'actions. Le but de la planification de mission est de sélectionner les objectifs à réaliser et de trouver la manière de les réaliser en tenant compte de l'environnement. Parmi les solutions possibles, le planificateur doit choisir celle qui optimise un critère prenant en compte des récompenses pour chaque objectif et des coûts pour les réaliser, et qui respecte des contraintes de temps et de ressources. Les récompenses et les contraintes sont des fonctions non linéaires du temps et des ressources aux différents moments où le véhicule effectue les actions qui conduisent à la réalisation des objectifs.

La réalisation d'un objectif est rarement ponctuelle dans le temps et dans l'espace : il y a un début de traitement, une fin de traitement et un moment où la récompense associée à l'objectif est obtenue. Comme on va le voir dans les sections suivantes, pour

de nombreux problèmes de la littérature sur l'optimisation dans les graphes, ces trois événements sont confondus (Figure 1.1).

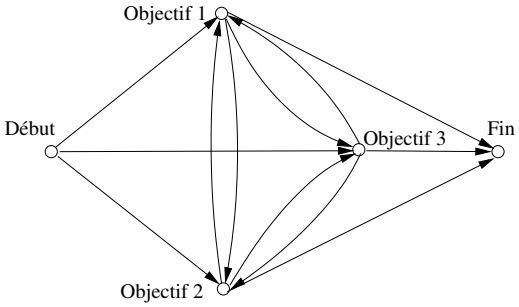


FIG. 1.1 – Approche par l'optimisation dans les graphes en recherche opérationnelle : le début, la fin de la réalisation et l'obtention des gains sont confondus

Pour des problèmes réels de planification de mission, la distinction entre les différentes étapes de la réalisation est nécessaire (Figure 1.2). Elle implique que les réalisations de différents objectifs peuvent s'entrelacer (Figure 1.3).

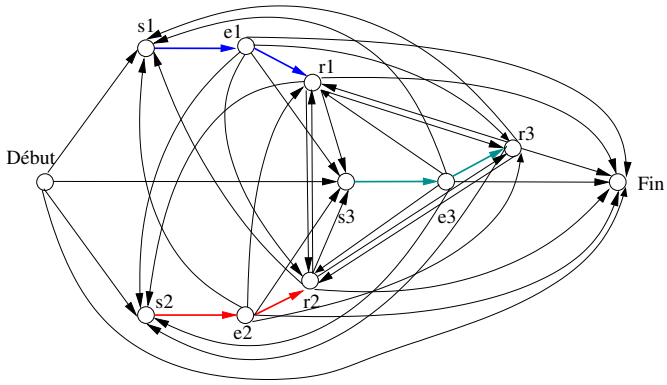


FIG. 1.2 – Graphe de planification de mission : le début (s_i), la fin (e_i) du traitement et le moment de l'obtention des gains (r_i) sont distincts pour les trois objectifs

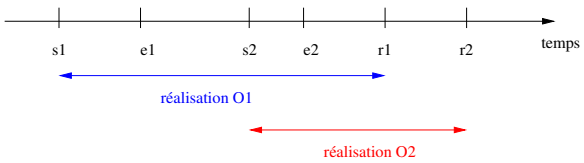


FIG. 1.3 – Entrelacement lors de la réalisation de deux objectifs O_1 et O_2

Le choix des actions pour réaliser un objectif a un impact sur le calcul des récompenses et des contraintes.

1.2 Littérature sur les modèles et méthodes de planification

Les modèles pour formaliser un problème de planification et les méthodes associées peuvent être classés en trois grands types [Thayse 1990] :
- les représentations de type logique ;
- les représentations de type graphe ;
- les représentations de type objet.

Les approches de type logique constituent une référence à laquelle la plupart des autres approches se rattachent, notamment en ce qui concerne la formalisation du raisonnement. Cependant, la concision du cadre de représentation logique est à la fois un avantage et un inconvénient dans la mesure où les informations ne sont pas structurées : ainsi, la représentation logique ne donne aucune indication de précédence.

Les représentations de type graphe offrent un cadre de représentation mieux structuré. Parmi les approches utilisant les graphes, on peut citer les réseaux de Petri [Murata 1989] et les réseaux Bayésiens [Pearl 1988].

Les représentations de type objet sont répandues du fait de leur adéquation avec les langages de programmation orientée-objet.

Une manière de structurer l'ensemble des connaissances sur le monde est de le replacer dans un espace d'état correspondant à l'ensemble des mondes possibles. Dans les approches logiques, chaque monde possible est déterminé par un ensemble de propriétés décrites par des formules logiques vraies. Dans la terminologie de l'Automatique, un état possible est déterminé par l'instanciation des variables d'état dans leurs domaines de valeurs respectifs.

L'approche logique fait généralement référence à une description statique éventuellement révisable, alors que l'approche par variables d'état implique souvent une représentation dynamique, et donc la prise en compte du temps et de l'évolution. Si la dénomination état correspond à un "instantané" du monde à un instant donné, cela ouvre la perspective de la prise en compte de l'aspect dynamique de l'environnement ou du système étudié. En effet, la dynamique d'un système se traduit alors par une succession d'états consécutifs indexés par un paramètre temps. Le formalisme état-action permet d'utiliser l'approche logique dans un cadre dynamique. Les états sont décrits par un ensemble de propositions logiques et une action est un opérateur permettant de passer d'un état à un autre.

Le but de la planification est de synthétiser une trajectoire abstraite ou non dans un espace de recherche, appelé aussi espace d'état, prédire les récompenses acquises pour cette trajectoire, choisir et organiser les actions de différents types pour atteindre un but et/ou optimiser des fonctions de récompense. Le résultat de la planification est un plan.

La suite de cette section présentera tout d'abord des **problèmes présentant des similitudes avec le problème de planification de mission** (Section 1.2.1). On dégagera les formalismes et les méthodes permettant de les traiter et on s'efforcera de

déterminer s'ils restent pertinents dans le cadre du problème de planification de mission.

La section 1.2.2 s'intéresse plus particulièrement aux **problèmes de sélection**, souvent décrits par le formalisme état-action.

Comme l'environnement est connu de manière incertaine dans le cadre de notre travail, il est intéressant d'étudier les formalismes permettant de prendre en compte les **incertitudes**, aussi bien sur l'état du système que sur les observations faites de l'environnement, comme par exemple les Processus Décisionnels de Markov (Section 1.2.3).

Enfin, dans le contexte de la planification de mission pour un véhicule autonome, le plan est nécessairement **relatif à un déplacement**, à l'exception des véhicules pour lesquels le déplacement est totalement subi, comme les satellites non agiles. La planification comprend la recherche d'un chemin de la configuration courante dans un espace géométrique à une configuration finale admissible en tenant compte du système et de son environnement. Le plan contient alors une séquence de points de passage dans l'espace géométrique considéré. La planification de déplacement peut être vue comme une recherche de chemin possible de la position courante du véhicule à des destinations données dans l'espace géométrique en évitant les obstacles de la meilleure façon [Latombe 1991]. Les méthodes naïves consistent à tracer une ligne droite entre le point de départ et le point d'arrivée et à suivre cette ligne. Si le véhicule rencontre un obstacle, il le contourne par la gauche ou par la droite. Des méthodes plus élaborées se basent sur la construction d'un graphe des configurations suivi d'une optimisation dans ce graphe. Ces méthodes, classiques, sont rappelées dans l'annexe A.

On pourra se reporter à l'ouvrage "Automated planning : theory and practice" [Ghallab, Nau, & Traverso 2004] pour une synthèse des méthodes de planification actuelles.

1.2.1 Problèmes présentant des similitudes avec le problème de planification de mission

Le Voyageur de Commerce et ses dérivés

Le problème du voyageur de commerce (Travelling Salesman Problem, TSP) est formulé de la manière suivante :
Etant donné un nombre fini de "villes" et le coût du trajet entre chaque paire de “villes”, on cherche le chemin le moins coûteux pour visiter toutes les villes et revenir à son point de départ.
L'analogie existante entre la tournée d'un voyageur de commerce et la visite d'une série de points d'intérêt par un véhicule aérien autonome est flagrante.

La modélisation mathématique du problème prend la forme d'un graphe où chaque ville est représentée par un nœud et où les arcs, orientés ou non, connectent chaque paire de nœuds. Chaque arc est pondéré par la distance entre ses deux extrémités. Quand le voyageur de commerce peut aller dans toutes les villes à partir de toutes les autres, le

problème est dit complet.

Les premières approches mathématiques exposées pour ce problème ont été traitées au $19^{ème}$ siècle par les mathématiciens Sir William Rowan Hamilton et Thomas Penyngton Kirkman. Hamilton en a fait un jeu : Hamilton's Icosian game [Herschel 1862]. Les joueurs devaient réaliser une tournée passant par 20 points en utilisant uniquement les connections prédéfinies. Le problème du voyageur de commerce appartient à la classe des problèmes d'optimisation combinatoire NP-difficiles, c'est-à-dire que la résolution de ce problème de décision peut s'effectuer en temps polynomial mais par un algorithme non déterministe (NP-complet). Cependant, en mai 2004, le problème du voyageur de commerce pour 24.978 villes a été résolu en Suède avec une instance de l'algorithme LKH [Helsgaun 2000] tirant parti de la notion d'échange pouvant convertir une tournée en une autre en invertissant un certain nombre de nœuds dans la tournée. C'est actuellement le plus grand exemple résolu de problème de voyageur de commerce, surpassant le record précédent de 15.112 villes en avril 2001.

Le problème du voyageur de commerce international [Laporte & Nobert 1983] ou ITSP est un problème où le voyageur visite des pays et doit choisir seulement une ville à visiter par pays. On retrouve le problème classique quand chaque pays contient exactement une seule ville.
Il est aisé de trouver une analogie entre le ITSP et la réalisation d'objectifs par un véhicule autonome en rapprochant le choix de la ville à visiter pour un pays et le choix d'une manière de réaliser un objectif.

La modélisation du problème prend la forme suivante [Bouali 1996] : soit $G = (N, A, W)$ un graphe non orienté ; N l'ensemble des nœuds, A l'ensemble des arêtes et W une partition de N. Une ville est représentée par un nœud de N et un pays par un élément de W.

De nombreuses méthodes sont envisageables pour résoudre ce problème. On distingue les algorithmes appliqués directement à une instance du ITSP, dits algorithmes directs, et les algorithmes approximatifs qui utilisent la décomposition du ITSP en deux sous-problèmes : le TSP et le problème de la recherche de la plus courte séquence de nœuds adjacents dans un graphe. Ces méthodes sont largement développées dans [Bouali 1996]. On citera pour exemple : la méthode du plus proche voisin, qui consiste à construire une tournée internationale en augmentant la taille de la chaîne courante du nœud le plus proche de l'une de ses extrémités ; la méthode d'insertion, qui consiste à considérer, à chaque itération, un cycle pour l'étendre d'un seul nœud ; les méthodes de recherche de chemin dans un graphe utilisant le principe de déroutement, les méthodes de voisinage utilisant la notion de cycle, les méthodes heuristiques de type tabou, algorithme génétique et recuit simulé.

Le problème du voyageur de commerce avec récompense est une variante du TPS où il n'est pas nécessaire de visiter toutes les villes. Visiter une ville rapporte une récompense. L'objectif est de trouver un chemin qui maximise les récompenses et minimise

les coûts sur le trajet.
L'analogie avec notre problème vient du fait que dans les deux cas, le planificateur doit choisir le chemin qui optimise un critère prenant en compte des récompenses et des coûts.

L'article de [Feillet, Dejax, & Gendreau 2001] propose un récapitulatif de la littérature sur le sujet. Le problème est modélisé par un graphe où chaque nœud représente une ville et est associé à une récompense constante positive ou nulle et chaque arc représente un trajet et est associé à un coût constant, positif ou nul.

Les méthodes envisagées pour résoudre ce problème sont multiples. Pour exemple, citons les algorithmes approximatifs, les heuristiques classiques, la recherche taboue, le branch and bound, le branch and cut et la programmation dynamique.

La course d'orientation

Le problème de course d'orientation correspond au problème des participant à une course d'orientation. Chaque point de contrôle est associé à une récompense. Les participants doivent en un temps donné accumuler le plus de récompenses possible. Ce problème ressemble au problème de voyageur de commerce avec récompense, sauf que les points de départ et d'arrivée ne sont pas confondus [Keller 1989].

Le problème de course d'orientation avec fenêtres temporelles, décrit par Kantor & Rosenwein [Kantor & Rosenwein 1992], ajoute une contrainte temporelle sur les nœuds, qui ne peuvent être visités que pendant un intervalle de temps défini.
Ce problème se rapproche du problème de planification de mission pour un véhicule car les points de passage sont munis de fenêtres temporelles en dehors desquelles les points ne sont pas valides.

La modélisation proposée considère un nœud origine et un nœud destination, un ensemble de nœuds (les clients), et un ensemble d'arcs, le coût de chaque arc représentant la durée du trajet. Chaque nœud client n_i est associé à une récompense $r(n_i)$ et à une fenêtre temporelle. L'objectif est de construire un chemin acyclique commençant au nœud origine et finissant sur le nœud destination, en maximisant les récompenses, en satisfaisant les contraintes de fenêtres temporelles sur tous les nœuds clients et en ne dépassant pas une durée maximale. Le problème est classifié NP-difficile et est également employé par [Smith 2004] pour la sélection d'objectifs.

La résolution proposée par Kantor & Roseinwein est une heuristique qui génère systématiquement une liste d'itinéraires possibles, puis sélectionne le meilleur itinéraire de la liste. Soit $\underline{n}$ le vecteur contenant la séquence de nœuds à parcourir. L'algorithme utilise un rapport temps/récompense permettant de juger si un itinéraire rapporte assez par rapport à la récompense espérée. Lorsque l'itinéraire rapporte assez par rapport à la récompense espérée, on dira "$\underline{\text{n}}$ vérifie R".

Heuristique proposée par Kantor & Rosenwein

début

1 Initialisation : $\underline{n} = \{n_1\}$; nombre de nœuds dans le plan = 1 ; initialisation de la meilleure récompense associée à $\underline{n}$

2 Traitement : Chercher la liste J_k des nœuds qui peuvent être ajoutés à $\underline{n}$ en satisfaisant les contraintes. Aucun élément de J_k n'est marqué

3 $n_j \leftarrow$ premier élément non marqué de J_k. S'il n'en existe pas, aller en 5

4 Marquer n_j. $\underline{n} \leftarrow \underline{n} + n_j$. Mettre à jour le nombre de nœuds dans le plan et la récompense associée à $\underline{n}$
si *($\underline{n}$ vérifie R)* **alors** Aller en 2 **sinon** Aller en 6

5 $\underline{n}$ ne peut pas être augmenté par un élément de J_k. S'il peut être augmenté avec le nœud destination en améliorant la meilleure récompense, un nouvel itinéraire est trouvé : mettre à jour la meilleure récompense

6 Revenir au niveau précédent de l'arborescence. Mettre à jour le nombre de nœuds du plan, la récompense courante et pointer sur le dernier nœud de $\underline{n}$

7 Test de terminaison :
si *(nombre de nœuds dans le plan = 0)* **alors** FIN **sinon** Aller en 3

fin

Dans [Kantor & Rosenwein 1992], cette heuristique est comparée à une heuristique d'insertion construisant un itinéraire en insérant de manière itérative de nouveaux nœuds au meilleur endroit possible de l'itinéraire courant. Si l'heuristique ne parvient pas à résoudre des problèmes de grande taille en une durée acceptable, elle obtient néanmoins des résultats de plus de 20% meilleurs en terme de qualité que la solution de l'algorithme d'insertion pour des problèmes de taille acceptable (environ 20 nœuds clients).

Le problème du voyageur de commerce et ses dérivés et la course d'orientation présentent de nombreuses similitudes avec le problème d planification de mission. Cependant, comme on l'a souligné dans la première partie de ce chapitre, ils ne rendent pas compte du processus complexe de réalisation d'un objectif.

1.2.2 Les problèmes de sélection

Présentation

La problématique récente des problèmes de planification présentant un nombre d'objectifs non fixés *a priori* aborde des problèmes avec un grand nombre d'objectifs possibles dont les récompenses sont différentes selon l'objectif et des actions de différents coûts. La fonction de planification doit sélectionner un sous-ensemble d'objectifs maximisant les récompenses et respectant des contraintes de mission et les disponibilités des ressources. Les exemples de planification pour ces problèmes sont pour la plupart des problèmes de planification pour la NASA tels que la planification pour des télescopes comme Hubble [Kramer & Giulianno 1997], SOFIA [Frank & Kurklu 2003], la plani-

fication pour des rovers sur Mars [Smith 2004]. Ces problèmes sont très proches du problème de sélection d'objectifs lors de la réalisation d'une mission.

Le formalisme utilisé pour les problèmes de sélection

Le formalisme utilisé pour les problèmes de sélection ou problèmes "sur-prescrits" est le formalisme état-action. C'est le cas par exemple dans [van den Briel *et al.* 2004]. Le formalisme permet d'utiliser l'approche logique dans un cadre dynamique. Les états sont décrits par un ensemble de propositions logiques et une action est un opérateur permettant de passer d'un état à un autre. Ce formalisme permet de résoudre de nombreux problèmes de planification classique mais peut aussi être utilisé dans le cadre des problèmes de sélection, où le nombre d'objectifs n'est pas fixé *a priori*.

Ce formalisme consiste à considérer un quadruplet $< E, A, T, L >$ où E est un ensemble fini ou énumérable d'états, un état e étant une conjonction de faits ; $T \subseteq E \times E$ est l'ensemble des transitions possibles entre états ; A est un ensemble d'étiquettes sur T, dites actions, une action a permettant de modifier l'état ; L est une relation sur $A \times T$. Si $(a, e, e') \in L$ alors l'action a est définie en e et peut mener en e'. On note $a(e) = \{e' | (a, e, e') \in L\}$. L'action a est déterministe si $a(e)$ est réduit au plus à un singleton.

L'état du système et de l'environnement est décrit par un ensemble de propositions logiques. Une action est un opérateur instantané entre deux états. Il est courant de caractériser une action a déterministe par trois ensembles de faits :

- Les conditions d'application ou préconditions : l'ensemble de faits qui doivent être dans l'état e pour que $a(e) \neq \emptyset$.
- Les faits ajoutés : les faits qui après application de a seront dans e' même s'ils ne sont pas dans e.
- Les faits détruits : les faits qui après application de a ne seront pas dans e' même s'ils sont dans e.

Ce formalisme est à la base des langages génériques de planification tels que le *STanford Research Institute Planning System* (STRIPS) [Fikes & Nilsson 1971] introduit au début des années 1970 ou le plus récent *Planning Domain Definition Language* (PDDL) et son extension PDDL2.1 [Fox & Long 2003], langage informatique standardisé de description de problèmes de planification. Ce langage permet de spécifier les opérateurs possibles, les relations et contraintes de l'environnement, l'état de départ et les états destination.

Le plan est alors une séquence d'actions permettant de passer d'un état initial donné à un état où tous les objectifs du problème font partie des faits. Un problème étant spécifié par $< E, A, T, R >$, un état initial $e_1 \in E$ et un sous-ensemble d'états but $E_g \subset E$, on cherche une séquence d'actions $(a_2, \ldots, a_n)$ telle que $a_n(\ldots a_3(a_2(e_1)))\ldots)$ soit un état de E_g.

Les planificateurs actuels utilisant le formalisme état-action ou des variantes sont

généralement utilisés hors ligne. Le plan généré est ensuite transmis au module d'exécution. Le plus souvent, les planificateurs n'ont pour objectif que de minimiser le nombre d'actions pour parvenir à un état but. Certains d'entre eux minimisent la somme des coûts des actions : aucun n'optimise un critère pouvant prendre en compte des coûts et des récompenses non sommables par action.
Parmi ces planificateurs, certains comme GRAPHPLAN [Blum & Furst 1995] recherchent un plan après une phase d'exploration (souvent approximée plutôt qu'exacte) des états atteignables. Cette phase consiste à construire un *graphe de planification* : l'espace de recherche.
D'autres planificateurs, comme HSP ou HSP-r [Bonet & Geffner 2004], utilisent la technique de la *recherche heuristique* pour réduire l'espace de recherche et produire rapidement des plans solutions. Ces méthodes ne sont souvent ni optimales, ni complètes car elles utilisent des algorithmes de recherche locale sous-optimaux. Ces algorithmes explorent l'espace des états atteignables à partir de l'état initial. Le graphe parcouru par l'algorithme a pour nœuds des états instanciés et individualisés. Les arcs représentent les actions qui mènent d'un état à un autre. Afin de limiter la recherche, l'heuristique estime pour chaque état le nombre d'actions nécessaires pour atteindre le but. Ainsi la recherche se focalisera sur les états promettant les plans les plus courts.
Un planificateur comme FF [Hoffmann & Nebel 2001] utilise une combinaison de la recherche heuristique et de la recherche dans un graphe de planification pour trouver des plans solutions. Grâce à sa performance, cette dernière technique a été adoptée par plusieurs autres planificateurs comme AltAlt [Nigenda & Kambhampati 2000].

Plus récemment, on s'est intéressé à des domaines contenant des connaissances numériques. Le langage PDDL2.1 supporte ainsi l'intégration des connaissances numériques dans la définition des problèmes de planification. Une nouvelle génération de planificateurs parmi lesquels Metric-FF [Hoffmann 2002] et SAPA [Do & Kambhampati 2001] sont capables de résoudre des problèmes ayant des contraintes de ressources numériques.

En outre, Galindo *et al.* [Galindo, Fernández-Madrigal, & González 2004] proposent un complément intéressant aux planificateurs classiques permettant de traiter des problèmes de grande dimension, suivant le principe d'abstraction. La représentation du monde est effectuée de manière hiérarchisée, ce qui améliore l'efficacité des calculs. La planification consiste à résoudre le problème en s'appuyant sur le niveau de description le plus abstrait du monde et de raffiner la solution avec les modèles où le monde est décrit plus finement. Formellement, le monde est défini par un graphe annoté et hiérarchique (AH-graphe). Un AH-graphe (Figure 1.4) est une structure hiérarchique basé sur les graphes modélisant l'environnement du robot à différents niveaux de détail. Un groupe de nœuds d'un niveau hiérarchique peut être remplacé par un nœud au niveau supérieur, appelé *supernœud*. De même, un groupe d'arcs d'un niveau hiérarchique peut être remplacé par un arc au niveau supérieur, appelé *superarc*.

Les problèmes de sélection ne peuvent pas être résolus par les planificateurs classiques pour lesquels tous les objectifs sont supposés avoir une récompense identique et constante dans le temps : le planificateur ne peut terminer son calcul que si tous les

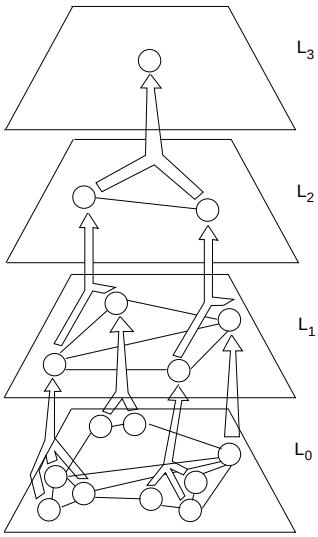


FIG. 1.4 – Exemple de AH-graphes avec quatre niveaux hiérarchiques. Illustration de l'abstraction des nœuds

objectifs sont réalisés par le plan. Ces problèmes lancent donc plusieurs défis : les critères terminaux du processus de planification changent car il n'y a plus un ensemble de buts en conjonction à satisfaire ; les heuristiques traditionnelles permettent de guider la recherche afin de réaliser un ensemble fixé d'objectifs et ne peuvent donc pas être directement réutilisées pour ces problèmes de sélection ; la qualité du plan doit prendre en compte non seulement le coût des actions, mais aussi les récompenses des objectifs réalisés dans le plan.

Les résolutions proposées

Une première approche dite gloutonne consiste à présélectionner un sous-ensemble d'objectifs et à trouver les actions pour ces objectifs par une méthode de planification classique. Pour le cas où il n'y a ni contrainte de temps, ni contrainte de ressources, l'application la plus naïve d'une méthode gloutonne consiste à sélectionner les objectifs de plus forte récompense. Ces méthodes, bien qu'efficaces, peuvent produire des plans de mauvaise qualité.
Une approche plus sophistiquée consiste à utiliser une heuristique de sélection. Celle-ci peut se calculer à partir du problème de planification relaxé, où on ne détruit pas les faits lors de l'application d'une action [van den Briel *et al.* 2004]. Les résultats expérimentaux indiquent qu'en général les objectifs sont correctement sélectionnés, mais que le fait d'employer ensuite une méthode de planification classique induit une dégradation du bénéfice due à la minimisation du nombre d'actions à la place de la minimisation du coût des actions.

Alternativement [Smith 2004], une heuristique de sélection en cas de contrainte de

temps (ou d'une ressource unique), peut se calculer en construisant un graphe de navigation correspondant à la projection des états sur les faits liés à la position du véhicule. Les coûts des arcs de ce graphe en termes de temps ou de consommation de la ressource sont calculés par un planificateur de chemin : un graphe de planification est construit et utilisé pour estimer la consommation de temps ou de ressource nécessaire à la réalisation des faits relatifs à chaque objectif. Les objectifs sont alors connectés au graphe de navigation par l'intermédiaire de deux arcs liant la position correspondante à l'objectif et l'objectif lui-même. Le coût de l'arc aller correspond à l'estimation de la consommation de temps ou de ressource calculée par le planificateur de chemin. Le coût de l'arc retour est nul. Le problème ainsi relaxé correspond à un problème de course d'orientation. La solution de ce problème, résolu par une méthode de recherche branch and bound, est utilisée pour guider le planificateur dans la recherche.
Pour le cas où il n'y a de contraintes ni sur le temps, ni sur les ressources, la sélection d'objectifs en cours de recherche d'actions peut être abordée en utilisant un algorithme de type A^* pondéré où les fonctions g et h ne correspondent pas au nombre d'actions respectivement dans le plan courant et pour atteindre un état but, mais au bénéfice net jusqu'à l'état courant et au bénéfice net additionnel pouvant être obtenu [Do & Kambhampati 2004] : contrairement à la planification dite "classique", on s'intéresse aux récompenses obtenues plutôt qu'au nombre d'actions du plan. Les états présentant une fonction f négative ne sont pas développés et l'algorithme s'arrête lorsque l'on trouve un nœud présentant une fonction h nulle et une fonction g plus grande que la meilleure fonction f : le problème est donc simplifié, car pour chaque action admise dans le plan, on suppose que la récompense est supérieure au coût de l'action. Le calcul de la fonction heuristique se fait sur le problème relaxé qui consiste à négliger les faits détruits. Cependant ce calcul ne conduit pas à une heuristique admissible puisque comme on l'a souligné, un nœud peut être écarté car il présente un f négatif alors qu'il est prometteur. Pour palier ce problème, des valeurs différentes du coût sont utilisées pour couper et pour classer. Les résultats expérimentaux montrent un intérêt par rapport à l'approche gloutonne tant en ce qui concerne le bénéfice que pour le temps de calcul.

Les problèmes de sélection sont très proches du problème de planification de mission. Le formalisme état-action permet de spécifier ce genre de problème en prenant en compte un ensemble de ressources. Le principe d'abstraction permet de représenter des mondes complexes. C'est un principe qui doit être utilisé. Néanmoins, même si le formalisme état-action présente une grande capacité de description, il présente des limites pour décrire entièrement le problème de planification de mission. Ainsi il n'est généralement pas adapté pour l'optimisation d'un critère pouvant prendre en compte des coûts et des récompenses non sommables par action. Il suppose une description complète de l'environnement et des opérateurs déterministes. De plus, l'environnement est supposé statique.

1.2.3 Traitement de l'incertitude : les processus markoviens

Dans tous les problèmes évoqués jusqu'à présent, toutes les incertitudes sur les états sont négligées : le monde dans lequel évolue le système est considéré comme déterministe. Ceci est rarement vérifié dans le monde réel : les actions effectuées peuvent avoir des effets variés, et l'état courant du processus est rarement connu avec certitude. Depuis la fin des années 40, les Processus Décisionnels de Markov (PDM) sont étudiés afin de prendre en compte les différentes incertitudes auxquelles est confronté un système évoluant dans un monde réaliste : incertitude des effets des actions et imprécision de localisation.

Les Processus Décisionnels de Markov [Puterman 1994] incluent un ensemble d'états discrets, un ensemble d'actions discrètes, une matrice de transitions donnant la probabilité d'être dans un état à un instant donné sachant l'état et l'action effectués à l'instant précédent, et une récompense élémentaire, fonction de l'état et de l'action.

Formellement, un PDM est un quadruplet $< E, A, P_T, R >$ où :
- E est l'ensemble fini des états. Tous les états possibles $(e_1, e_2, \ldots)$ sont, sauf exception, explicitement énumérés.
- A est l'ensemble fini des actions réalisables dans tous les états. Une action a permet d'envisager une distribution d'états suivants possibles (e') dépendante de l'état de départ.
- P_T et R sont les probabilités et récompenses des transitions entre états pour chaque action.

Le quadruplet vérifie la propriété de Markov :

$$P_T(e, a, e') = P(e'|a, e) \tag{1.1}$$

L'équation 1.1 signifie que la probabilité de transition entre e et e' en effectuant l'action a ne dépend que de la connaissance de l'état précédent e et non de tout l'historique des états parcourus. Il existe une variante du formalisme où les possibilités sont utilisées à la place des probabilités.

L'algorithme de résolution d'un problème de décision séquentielle relatif à un PDM suit le principe de la programmation dynamique [Bellman 1952]. La résolution produit alors une politique qui indique l'action optimale à chaque étape pour chaque état, ce qui donne un plan conditionnel.

Cette approche est par exemple utilisée pour un problème de planification d'exploration [Teichteil & Fabiani 2005]. Une approche hybride entre la factorisation de l'espace d'état et la décomposition est proposée. La factorisation consiste à utiliser une représentation plus compacte de l'espace d'état du PDM en le factorisant par composantes d'état. La décomposition, quant à elle, consiste à agréger les états d'un espace d'état entièrement explicités au sein de régions faiblement communiquantes, c'est-à-dire dont le nombre de transitions entre les états de sortie et d'entrée est faible. Elle tient compte de la quantité limitée de carburant par une variable binaire.

Un autre exemple d'application est décrit pour le contrôle de rover [Bernstein *et al.* 2001] dont le but est d'explorer une planète, prendre des photos et effectuer des expériences. La difficulté du problème vient des contraintes temporelles et sur les ressources (batterie, capacité de stockage). Le modèle de l'environnement prend en compte des incertitudes sous la forme de probabilités. Le but est de maximiser la somme des récompenses dues à l'accomplissement d'une tâche en respectant les contraintes. Ce problème, très similaire au problème de planification de mission qu'on souhaite résoudre, est modélisé par un PDM. Le problème est résolu en réduisant l'espace de recherche en simplifiant le modèle. Seuls 60 états sont ainsi énumérés.

Finalement, on distingue les Processus Décisionnels de Markov où l'état est directement observé des Processus Décisionnels de Markov Partiellement Observables (PDMPO) où l'on se pose réellement le problème de l'observation. Ce modèle prend en compte le bruit des observations. Formellement, un PDMPO est un 6-uplet $< E, A, Z, T, R, O >$, où E, A, T et R sont définis comme pour un PDM. Z représente un ensemble fini d'observations. Une observation z résulte de l'état courant au travers d'une autre distribution de probabilité. Toutes les observations possibles $(z_1, z_2, \ldots)$ sont, sauf exception, explicitement énumérées. Une fonction d'observation $O(e, a, z) = P(z|e, a)$ donne la probabilité d'obtenir l'observation z donnant l'état e et l'action a de l'instant précédent. La résolution d'un problème de décision modélisé par PDMPO est notablement plus coûteux en temps de calcul et difficilement applicable pour des applications réalistes. Il est néanmoins envisagé pour la prise de décision haut niveau pour les drones dans [Schesvold *et al.* 2003]. Cependant, pour tous les problèmes d'ordre de grandeur réel, trouver une solution est un problème difficile. Seuls des problèmes avec très peu d'états et d'actions peuvent être résolus exactement.

La résolution d'un problème de décision modélisé par PDM permet de traiter des problèmes réels en prenant en compte les incertitudes. Cependant, cela suppose de pouvoir construire un modèle probabiliste, notamment sur les transitions entre états. Une hypothèse dans notre travail est que l'on ne peut pas construire ce modèle. Par ailleurs, cette approche ne prend pas du tout en compte des contraintes temporelles, ce qui est un point important dans notre problème. Enfin, la discrétisation des variables continues en variables discrètes est un désavantage pour l'optimisation d'un critère pouvant par exemple dépendre de la vitesse du véhicule.

1.2.4 Conclusions

Cette section a présenté quelques problèmes présentant des similitudes avec le problème de planification de mission. La modélisation par un graphe permet d'appliquer de nombreux algorithmes pour chercher un chemin de coût minimal, un circuit passant par un ensemble de nœuds ou un chemin prenant en compte des récompenses obtenues aux nœuds et des coûts dispensés sur les arcs. Cette dernière forme de problèmes semble particulièrement adaptée à la planification de mission en ligne. En effet, elle permet de rendre compte d'un abandon d'objectif de mission à cause d'un coût trop

élevé. Cependant, l'affectation de revenus élémentaires aux nœuds peut ne pas suffire à rendre compte du mécanisme de réalisation des objectifs. Le problème du voyageur de commerce international permet plusieurs possibilités pour cette réalisation.

Les problèmes de sélection abordent des problèmes où le nombre d'objectifs n'est pas fixé *a priori*. Le formalisme état-action permet de spécifier ces problèmes en terme de buts de manière beaucoup plus flexible qu'un graphe mais indépendante d'un graphe de navigation. Un compromis entre ces deux approches doit être réalisé. Par ailleurs, le formalisme état-action permet de prendre en compte un ensemble de ressources. Cet aspect est de première importance dans le problème de planification de mission. En effet, pour réaliser une mission, un véhicule peut consommer différentes ressources. Enfin, le principe d'abstraction permet de décrire un monde complexe. Il sera intéressant d'exploiter cette idée dans le cadre de nos travaux.

Les processus décisionnels de Markov permettent de prendre en compte les incertitudes au travers de probabilités. Les algorithmes associés sont plus semblables à l'algorithme de Ford-Bellman (Annexe A) qu'aux explorations d'arborescence. Dans ce travail, le choix est fait de ne pas approfondir la voie liée aux processus Décisionnels de Markov. Néanmoins, au niveau de l'application, on cherchera à prendre en compte l'incertitude aux travers de ressources représentant des probabilités.

La planification de mission présente enfin un aspect relatif à la construction d'un graphe à partir d'un espace géométrique encombré. Les nœuds du graphe peuvent correspondre soit à des cellules soit à des points. L'approche proposée dans cette thèse se base sur l'hypothèse qu'un graphe a été construit et que les nœuds de ce graphe sont des points. En effet, cette solution semble plus crédible en terme d'évaluation du coût d'un chemin.

L'analyse de l'état de l'art concernant les aspects relatifs à la planification de mission amènent à cadrer ce travail de la manière suivante :
- La planification se fait dans un graphe dont les nœuds sont des points de l'espace géométrique.
- La planification considère des récompenses à certains nœuds et des coûts sur les arcs. Cependant, une affectation d'une récompense d'objectif à un nœud unique est trop simpliste. Il est nécessaire d'étendre la flexibilité de cette affectation sans pour autant aller jusqu'à la complexité du formalisme état-action.
- Le coût d'un arc ne concerne pas une variable unique, mais un ensemble de ressources.
- La modélisation des aspects probabilistes ne se fait pas dans le cadre classique. on cherchera plutôt à les prendre en compte au cas pas cas au travers de ressources particulières.

Par ailleurs, on peut noter certains manques dans les approches existantes :
- La vitesse de l'engin est importante pour la détermination de la consommation de ressources. Cette relation n'est pas explicitée.

- Les ressources sont consommées de manière incrémentale, arc par arc, alors qu'il arrive que dans certains cas, leur consommation ne puisse pas être décomposée de cette manière.

Le travail décrit dans la suite du manuscrit vise à combler ces manques.

1.3 Hypothèses et variables de la planification

La planification de mission s'effectue dans un contexte et pour un système dynamiques.

☞ On entend par contexte et système **dynamiques** un contexte et un système dont les états sont des fonctions du temps. Cependant, toutes les fonctions permettant de donner les états du système et de l'environnement sont connues.

C'est pour cette raison que le problème sera décrit par des variables d'état. C'est sur cet espace d'état que la planification va synthétiser un plan. Pour rappel, les variables d'état permettent de décrire un instantané du monde à un instant donné, les variables d'action permettront de passer d'une instanciation de l'ensemble des variables d'état à la date t_k à une instanciation de l'ensemble des variables d'état à la date t_{k+1}.

1.3.1 Planification déterministe en ligne

Etant donné qu'il y a peu de moyens d'anticiper des points de replanification au travers d'un modèle prédictif, on suppose que le déroulement de la mission va s'effectuer dans le contexte courant. Le plan obtenu peut ainsi être remis en cause. Par exemple, pour une opération de recherche d'objet, la durée de l'opération prévue dans la planification sera l'espérance mathématique de la durée réelle de l'opération. Si la durée réelle est trop différente des prévisions, il faudra déclencher une replanification en ligne.

Hypothèse 1 *La planification ne prévoit pas de replanification dans son plan : elle s'effectue pour une hypothèse unique sur l'état et son modèle d'évolution.*

1.3.2 Les variables d'état du problème

Soit N un ensemble de nœuds. Etant donnés deux nœuds successifs n_k et n_{k+1} (Figure 1.5), une décision prise entre n_k et n_{k+1} sera appliquée en n_{k+1}. D'un point de vue géométrique, n_{k+1} est le nœud vers lequel le véhicule se dirige.

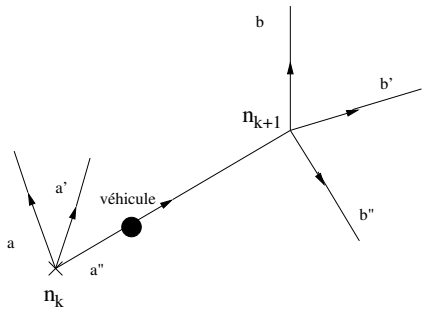


FIG. 1.5 – Succession des nœuds état

L'état à un moment donné est donc défini par :
- Le nœud vers lequel le véhicule est en train de se diriger ;
- La date t ;
- L'état des ressources ;
- L'état de réalisation des objectifs de la mission.

DÉFINITION 1 *On appellera* ***contexte de la mission*** *l'ensemble comprenant l'objet de la mission, l'état courant et l'état connu de la carte de l'environnement.*

1.3.3 Les variables d'action du problème

Les variables d'action sont aussi les variables de la planification ou d'optimisation. De manière informelle, les éléments donnés en sortie de la planification sont :
- le nœud suivant, connu par ses coordonnées dans l'espace physique où se déroule la mission, par exemple (x, y) ou (x, y, z) ;
- la date à laquelle ce nœud est atteint ;
- le mode de déplacement et d'action mis en œuvre pour atteindre le nœud visé.

1.3.4 Relation entre deux états successifs

La figure 1.6 illustre la relation entre deux états successifs. Cette relation est soumise aux hypothèses suivantes :

HYPOTHÈSE 2 *La vitesse v est constante entre deux nœuds successifs n_k et n_{k+1}. Elle est bornée sur l'ensemble des valeurs réelles strictement positives en fonction des nœuds et de l'action a_{k+1} entreprise.*
Les vitesses maximale et minimale dans $\mathbb{R}_+^*$ sont notées

$$v_{max}(n_k, n_{k+1}, a_{k+1}) \text{ et } v_{min}(n_k, n_{k+1}, a_{k+1})$$

avec $v_{min}(n_k, n_{k+1}, a_{k+1}) \leq v_{max}(n_k, n_{k+1}, a_{k+1})$.

DÉFINITION 2 *Un* ***itinéraire*** *est une liste ordonnée de nœuds à suivre pour aller d'un nœud de début n_1 à un nœud de fin d'itinéraire n_q.*

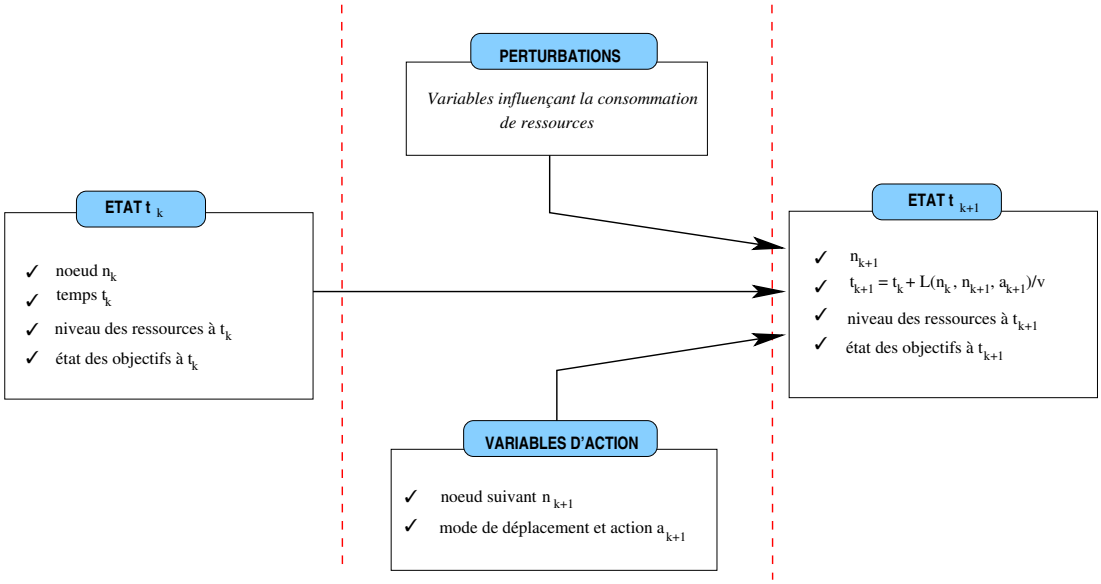


FIG. 1.6 – Représentation du changement d'état

DÉFINITION 3 *Une **trajectoire** est une courbe reliant deux nœuds consécutifs d'un itinéraire.*

Soit e le nombre de nœuds dans l'itinéraire considéré. Les variables d'action sont :
- la liste ordonnée de points d'itinéraire à parcourir, notée $\underline{n} = n_2 \dots n_e$;
- une liste d'actions associées pour atteindre chaque nœud de l'itinéraire, notée $\underline{a} = a_2 \dots a_e$;
- le vecteur des durées pour parcourir chaque arc de l'itinéraire, noté $\Delta = \Delta^{a_2}_{n_1,n_2}$-$\Delta^{a_3}_{n_2,n_3} \dots \Delta^{a_e}_{n_{e-1},n_e}$.

Pour une action a_{k+1} donnée pour aller de n_k à n_{k+1}, la composante $\Delta^{a_{k+1}}_{n_k,n_{k+1}}$ est équivalente à une vitesse v constante.

$$\Delta^{a_{k+1}}_{n_k,n_{k+1}} = \frac{L(n_k, n_{k+1}, a_{k+1})}{v} \tag{1.2}$$

où $L(n_k, n_{k+1}, a_{k+1})$ est la longueur de la trajectoire liant n_k à n_{k+1} lorsque l'action a_{k+1} est réalisée.

1.4 Formalisation du problème

On propose dans les sections suivantes un formalisme permettant de décrire entièrement le problème de planification de mission. Ce formalisme générique s'applique à une large classe de problèmes de planification de mission.

L'idée est de tirer parti du principe d'abstraction [Galindo, Fernández-Madrigal, & González 2004] pour organiser la description du problème dans une hiérarchie à deux niveaux. Le haut niveau décrit la réalisation des objectifs. Le bas niveau décrit la réalisation des actions dans le temps et l'utilisation des ressources. Un groupe de nœuds du bas niveau est représenté par un unique nœud de haut niveau, appelé supernœud. De même, un groupe d'arcs de bas niveau peut être représenté dans le haut niveau par un arc unique, appelé superarc.

1.4.1 Description haut niveau : réalisation d'un objectif

Soit $W = \{W_1, \ldots, W_e\}$ une partition de N. On définit la relation S suivante : pour un sous-ensemble W_i, $S(W_i)$ est défini comme le sous-ensemble de W des successeurs de W_i. W_j est un successeur de W_i s'il existe au moins une action de haut niveau possible entre W_i et W_j.

On définit une machine à état fini grâce à W et à la relation S. Sans perte de généralité, on définit :

- W_1 comme l'état initial ;
- W_e comme la fin de la mission.

La fonction de transition de la machine à état fini est définie par les possibilités de transition (S) et le résultat de la planification (meilleure action à effectuer définie par le plan).

Quel que soit $W_i \in W$, W_i est accessible de W_s et W_e est accessible de W_i.

Soit P l'ensemble des objectifs à essayer de réaliser. Chaque objectif o de P est défini par un triplet $(W_{s(o)}; W_{e(o)}; W_{r(o)})$ dans W^3 avec $s(o), e(o)$ et $r(o)$ dans $\{1, \ldots, e\}$.

Définition 4 *On dit qu'un objectif est* ***traité*** *lorsqu'une transition de $W_{s(o)}$ vers $W_{e(o)}$ est tirée.*

Définition 5 *La récompense associée à l'objectif est obtenue quand l'objectif a été traité et que la machine à état fini est dans l'état $W_{r(o)}$: on dit que l'objectif est* ***réalisé****.*

La récompense (gain, revenu) est une quantité réelle positive. Pour chaque objectif o, la machine à état fini ne peut être dans l'état $W_{e(o)}$ qu'une seule fois.

Soit t_k et r_k respectivement la date et le vecteur des ressources à l'état W_k ; t_1 et r_1 sont la date et le vecteur des ressources en début d'application du plan. La récompense R_o pour avoir réalisé un objectif o est alors de la forme :

$$R_o(t_{s(o)}, r_{s(o)}, t_{e(o)}, r_{e(o)}, t_{r(o)}, r_{r(o)}) \tag{1.3}$$

On indique ainsi que la récompense est une fonction qui dépend de la date à laquelle le traitement débute puis s'achève, de la date d'obtention de la récompense, ainsi que du niveau des ressources en ces instants. Par exemple, pour un problème concret de planification de mission pour un drone, la récompense pour avoir effectué une observation de zone dépend des dates de début et de fin d'observation et de l'instant auquel

l'information est transmise à la station sol. Elle peut aussi dépendre des ressources (mémoire...) disponibles pendant l'observation.

A la fin du plan, le véhicule atteint W_e à l'instant t_e avec les ressources r_e. On définit une fonction de coût de ces ressources $R_e(t_e, r_e)$.

Le but de la planification est donc de trouver une séquence Q d'états $W_{\pi(1)}, \ldots, W_{\pi(q)}$ telle que π est une fonction de $\{1, \ldots, q\}$ dans $\{1, \ldots, e\}$ avec :

$$\pi(1) = 1; \quad \pi(q) = e; \quad W_{\pi(i+1)} \in S(W_{\pi(i)})$$

La séquence Q doit minimiser la différence J entre les coûts et les récompenses, en satisfaisant les contraintes sur les ressources. Soit E_o l'ensemble des objectifs réalisés, c'est-à-dire qu'il existe i tel que $W_{\pi(i)} = W_{s(o)}$ et $W_{\pi(i+1)} = W_{e(o)}$ et qu'il existe $j > i$ tel que $W_{\pi(j)} = W_{r(o)}$. Alors J vérifie l'équation 1.4.

$$J = R_e(t_e, r_e) - \sum_{o \in E_o} R_o \tag{1.4}$$

Les ressources sont en quantité limitée. On associe à chaque composante r^s du vecteur ressources une contrainte donnée par l'inégalité à l'état W_e :

$$r_e^s - r_{min}^s \geq 0 \tag{1.5}$$

Ce problème de réalisation d'objectifs n'est pas totalement spécifié car l'évaluation des dates et des ressources dépend du choix d'un nœud dans chaque $W_{\pi(i)}$ de la séquence et de l'action permettant d'atteindre $W_{\pi(i)}$, et pour chaque nœud de la date à laquelle il est atteint. Il doit être complété par une description plus bas niveau.

1.4.2 Description bas niveau

La relation S est étendue au niveau de description des nœuds : étant donnés deux nœuds n et m de N, il existe un ou plusieurs arcs entre n et m si et seulement si il existe i et j avec n dans W_i, m dans W_j et W_j dans $S(W_i)$.

Le temps

Soit A l'ensemble des actions possibles de bas niveau. Pour chaque couple (W_i, W_j) tel que $W_j \in S(W_i)$, il existe un sous-ensemble $A_{i,j}$ de A indiquant les actions autorisées entre W_i et W_j.

Pour deux nœuds n_k et n_{k+1} choisis respectivement dans $W_{\pi(k)}$ et $W_{\pi(k+1)}$ de la séquence Q, les dates à n_k et n_{k+1} sont liées par la relation :

$$t_{\pi(k+1)} = t_{\pi(k)} + \Delta_{n_k, n_{k+1}}^{a_{k+1}} \tag{1.6}$$

où $\Delta_{n_k, n_{k+1}}^{a_{k+1}}$, définie au paragraphe 1.3.4, est bornée en fonction des nœuds n_k, n_{k+1} et de l'action a_{k+1}. Les bornes sont déduites des valeurs maximale et minimale de la vitesse (Hypothèse 2) appliquées à l'équation 1.2. Elles sont données par l'équation 1.7.

$$\underline{\Delta}^{a_{k+1}}_{n_k,n_{k+1}} \leq \Delta^{a_{k+1}}_{n_k,n_{k+1}} \leq \bar{\Delta}^{a_{k+1}}_{n_k,n_{k+1}} \tag{1.7}$$

Avec :

$$\underline{\Delta}^{a_{k+1}}_{n_k,n_{k+1}} = \frac{L(n_k,n_{k+1},a_{k+1})}{v_{max}(n_k,n_{k+1},a_{k+1})}$$

$$\bar{\Delta}^{a_{k+1}}_{n_k,n_{k+1}} = \frac{L(n_k,n_{k+1},a_{k+1})}{v_{min}(n_k,n_{k+1},a_{k+1})}$$

La date d'application du plan t_1 est connue. W_1 ne contient qu'un seul nœud, noté n_1. Chaque nœud a une fenêtre temporelle, de sorte que t_k est soumis à la relation 1.8.

$$t_{k_{min}} \leq t_k \leq t_{k_{max}} \tag{1.8}$$

La consommation des ressources

Soit une séquence de nœuds $n_1, \ldots n_i$ tels que $n_k \in W_{\pi(k)}$, soit une séquence d'actions $a_2, \ldots, a_i$ telles que a_k est l'action entre n_{k-1} et n_k, le niveau $r^s_{\pi(i)}$ de la $s^{ième}$ ressource en un point n_i ne dépend que des nœuds parcourus de n_1 à n_i, des actions entre ces nœuds et des durées de parcours entre chaque nœud.

La forme la plus générale du niveau $r^s_{\pi(i)}$ de la $s^{ième}$ ressource en un point n_i est donnée par l'équation 1.9.

$$r^s_{\pi(i)} = f^s_{\pi(i)}(n_1, \ldots, n_i, a_2, \ldots, a_i, \Delta^{a_2}_{n_1,n_2}, \ldots, \Delta^{a_i}_{n_{i-1},n_i}) \tag{1.9}$$

Dans le cas de ressources suivant une équation d'état, on peut écrire l'équation 1.10.

$$r^s_{\pi(i)} = \breve{f}^s_{\pi(i)}(r^s_{\pi(i-1)}, n_{i-1}, n_i, a_i, \Delta^{a_i}_{n_{i-1},n_i}) \tag{1.10}$$

Si on suppose de plus que le niveau de la ressource est linéaire suivant le niveau à l'état précédent et suivant les durées, alors l'équation 1.10 se simplifie en l'équation 1.11.

$$r^s_{\pi(i)} = \dot{f}^s_{\pi(i)}(n_{i-1}, n_i, a_i) r^s_{\pi(i-1)} + \ddot{f}^s_{\pi(i)}(n_{i-1}, n_i, a_i) \Delta^{a_i}_{n_{i-1},n_i} \tag{1.11}$$

Une ressource suivant une équation d'état peut aussi être décomposable. Les ressources décomposables sont définies par le fait que le niveau des ressources en un nœud n_i peut être exprimé par le niveau de ressources au nœud n_{i-1} et l'utilisation des ressources entre n_{i-1} et n_i. L'équation 1.10 se simplifie alors en l'équation 1.12.

$$r^s_{\pi(i)} = r^s_{\pi(i-1)} + \tilde{f}^s_{\pi(i)}(n_{i-1}, n_i, a_i, \Delta^{a_i}_{n_{i-1},n_i}) \tag{1.12}$$

Enfin, si l'on suppose le niveau d'une ressource décomposable et linéaire, les équations 1.11 et 1.12 se simplifient en l'équation 1.13.

$$r^s_{\pi(i)} = r^s_{\pi(i-1)} + \bar{f}^s_{\pi(i)}(n_{i-1}, n_i, a_i) \Delta^{a_i}_{n_{i-1},n_i} \tag{1.13}$$

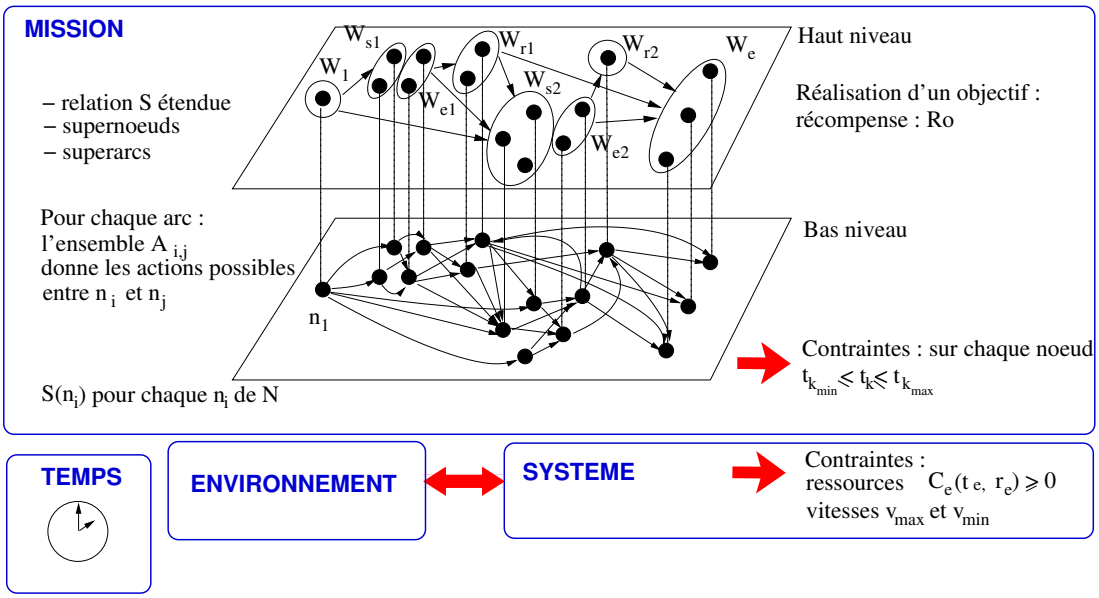


FIG. 1.7 – Les entrées du problème de planification

1.4.3 Synthèse : entrées et sorties du problème

Les entrées du problème sont :
- Un graphe de bas niveau constitué d'un ensemble de nœuds, noté N. n_1 est le nœud origine : il correspond au nœud où le plan est appliqué. Il existe un sous-ensemble de N de nœuds destination, noté W_e. Les arcs du graphe représentent les actions possibles entre deux nœuds. Une relation S donne l'ensemble de successeurs de chaque nœud du graphe.
- Un graphe de haut niveau, construit à partir d'une partition de N notée W. Il permet de décrire la réalisation des objectifs de la mission, qui donne lieu à une récompense R_o.
- Les données des fonctions de ressources du système. Les ressources sont fonction du temps, des actions et chemins choisis dans les deux graphes et de l'environnement.
- Les contraintes temporelles sur chaque nœud de N, les contraintes sur les ressources.

Les données d'entrée de la formalisation proposée pour le problème de planification sont illustrées figure 1.7. Le problème est entièrement spécifié par la donnée des deux niveaux de planification, des données du système, de l'environnement et de l'horloge. On s'aperçoit que la donnée seule du bas niveau ne suffit pas, car les récompenses dépendent du déplacement dans les supernœuds. A l'inverse, la donnée du haut niveau ne suffit pas car elle ne donne aucune indication précise sur les actions possibles entre les nœuds et les contraintes temporelles, décrites dans le bas niveau de description.

Le résultat de la planification est donc :
- Pour le haut niveau, une séquence Q d'états $W_{\pi(1)}, \ldots, W_{\pi(q)}$ telle que π est une fonction de $\{1, \ldots, q\}$ dans $\{1, \ldots, e\}$ avec :

$$\pi(1) = 1; \quad \pi(q) = e; \quad W_{\pi(i+1)} \in S(W_{\pi(i)})$$

  Cette séquence d'état définit le haut niveau de la planification : le choix des objectifs à réaliser et l'ordre de leur réalisation.
- Un vecteur de nœuds $n_1, \ldots, n_e$, un vecteur d'actions $a_2, \ldots, a_e$ et un vecteur de durées Δ, dont les composantes sont les $\Delta^{a_i}_{n_{i-1},n_i}$. Ces trois données peuvent être notées sous la version condensée $\Delta^{\underline{a}}_{\underline{n}}$. Ce vecteur représente implicitement toutes les variables de décision de bas niveau pour la réalisation des objectifs : choix des nœuds dans les ensembles $W_{\pi(i)}$, choix de l'action à effectuer entre ces nœuds, durée de l'action.

1.4.4 Quelques hypothèses et propriétés associées

HYPOTHÈSE 3 *On suppose que la fonction de récompense R_o est bornée sur l'ensemble des données d'entrée réelles.*

HYPOTHÈSE 4 *La fonction de coût du temps et des ressources $R_e(t_e, r_e)$ est supposée croissante avec le temps et décroissante avec les ressources.*

HYPOTHÈSE 5 *Les ressources sont supposées en quantité limitée, consommables et non renouvelables.*

☞ Un problème avec des ressources en quantité illimitée n'aurait aucun intérêt. En effet si les ressources sont en quantité illimitée, il n'y a aucune contrainte dessus, ni aucun avantage à minimiser leur utilisation. Donc par défaut, toutes les ressources sont disponibles en quantité limitée.

PROPRIÉTÉ 1 *Sous l'hypothèse 5, on peut écrire :*

$$r_{\pi(i+1)} \leq r_{\pi(i)}$$

PROPRIÉTÉ 2 *Lorsque toutes les ressources sont décroissantes dans le temps, même si la variation entre deux états n'est pas complètement connue, le coût relatif aux ressources courantes et au temps courant peut être considéré comme déjà dépensé.*

Grâce à la propriété 2, si on considère l'état $W_{\pi(i)}$ sur la séquence Q, on peut définir un coût partiel $R_e(r_{\pi(i)})$ sur un itinéraire courant et un coût résiduel $\sigma_{\pi(i),e}$ selon l'équation 1.14.

$$R_e(t_e, r_e) = R_e(t_{\pi(i)}, r_{\pi(i)}) + \sigma_{\pi(i),e} \text{ avec } \sigma_{\pi(i),e} \in \mathbb{R}^+ \tag{1.14}$$

1.4.5 Complexité

Le problème de planification décrit précédemment est un problème hybride : il comprend à la fois un aspect discret correspondant à la recherche de chemin dans un graphe et un aspect continu correspondant à l'optimisation des durées sur ce chemin. En faisant l'abstraction de l'aspect continu et en effectuant quelques hypothèses sur le graphe, il est possible de calculer une borne[1] sur la combinatoire de la recherche. Les hypothèses posées sont :

- quel que soit l'objectif o, le nombre de nœuds dans chacun des ensembles $W_{s(o)}$ ou $W_{e(o)}$ est fixé à K ;
- chaque W_i de W est soit l'ensemble W_1, soit l'ensemble W_e, ou l'un des ensembles définissant une zone objectif o dans P ;
- pour chaque objectif o, le nombre de nœuds dans $W_{r(o)}$ est fixé à T ;
- le nombre maximum de nœuds dans l'ensemble W_e est fixé à $\overline{W_e}$.

Soit $\overline{P}$ le cardinal de P.

PROPRIÉTÉ 3 *Sous ces hypothèses, le nombre maximum d'itinéraires est inférieur à* $\overline{W_e}.\overline{P}![K^2(1+T)]^{\overline{P}}e^{1/[K^2(1+T)]}$.

Preuve : Le nombre d'itinéraires ayant k objectifs, noté I_k, est égal à :

$$I_k = \frac{\overline{P}!}{(\overline{P}-k)!}[K^2(1+T)]^k\overline{W_e} \tag{1.15}$$

La preuve de l'équation sur I_k (1.15) est faite par récurrence. Pour un itinéraire sans objectif, il existe $\overline{W_e}$ itinéraires : $I_0 = \overline{W_e}$. En supposant l'équation 1.15 vraie et en ajoutant un nouvel objectif, alors $I_k/\overline{W_e}$ est le nombre d'itinéraires qui n'atteignent pas W_e et contiennent k objectifs. Pour le $(k+1)^{\text{ième}}$ objectif, il reste $(\overline{P}-k)$ choix d'objectifs avec K nœuds dans $W_{s(o)}$ et K nœuds dans $W_{e(o)}$ pour réaliser l'objectif, puis T nœuds dans $W_{r(o)}$ pour obtenir la récompense avant de finir l'itinéraire ou directement $\overline{W_e}$ possibilités pour finir le chemin. Le nombre d'itinéraires ayant $k+1$ objectifs est donc :

$$I_{k+1} = \frac{I_k}{\overline{W_e}}.(\overline{P}-k)K^2(1+T)\overline{W_e}$$

Soit :

$$I_{k+1} = \frac{\overline{P}!}{(\overline{P}-k-1)!}[K^2(1+T)]^{k+1}\overline{W_e}$$

Ce qui prouve l'équation 1.15.

Considérons tous les itinéraires contenant de zéro objectif jusqu'à $\overline{P}$ objectifs, le nombre maximum d'itinéraires calculé à l'aide de l'équation 1.15 est donc :

$$\mathbb{Y} = \sum_{k=0}^{\overline{P}} \frac{\overline{P}!}{(\overline{P}-k)!}[K^2(1+T)]^k\overline{W_e}$$

[1]Merci à M. Dominique Luzeaux pour ses corrections concernant le calcul de cette borne

Or pour $x > 0$, on a :

$$\sum_{k=0}^{n} \frac{n!}{(n-k)!} x^k = n! x^n \sum_{k=0}^{n} \frac{x^{k-n}}{(n-k)!} = n! x^n \sum_{k=0}^{n} \frac{1}{k!} (1/x)^k \leq n! x^n e^{1/x}$$

avec l'égalité quand n tend vers l'infini.

Donc, comme $[K^2(1+T)] > 0$:

$$\mathbb{Y} = \sum_{k=0}^{\overline{P}} \frac{\overline{P}!}{(\overline{P}-k)!} [K^2(1+T)]^k \overline{W_e} \leq \overline{P}! [K^2(1+T)]^{\overline{P}} e^{1/[K^2(1+T)]} \overline{W_e}$$

avec l'égalité quand $\overline{P}$ tend vers l'infini.

□

Le nombre maximum d'itinéraires n'est donc limité que proportionnellement à la factorielle du nombre d'objectifs. On conçoit aisément que lors de l'exploration de l'arborescence des chemins, cette borne peut être atteinte : ce majorant est un bon indicateur du nombre d'itinéraires à explorer si aucun moyen pour limiter la recherche n'est mis en œuvre.
Pour exemple, une mission simple avec 2 objectifs, 2 points de fin possibles, 2 points d'entrée et de sortie par objectif et 1 point de transmission peut amener à explorer jusqu'à 290 chemins. Une mission avec 5 zones objectif, 4 points de fin possibles, 2 points d'entrée et de sortie par objectif et 2 points de transmission peut amener à explorer de l'ordre de 10^8 chemins.

Cette étude de complexité montre l'intérêt d'élaborer des algorithmes qui n'explorent pas entièrement l'arbre de recherche.

1.5 Application à une mission d'observation

1.5.1 Présentation du problème

Nous posons le problème de décision-planification dans le cadre d'une mission d'observation pour un système de drone dans un environnement 3D dynamique, incertain et dangereux. Le système de drone est donc constitué d'un vecteur (engin), de sa charge utile (capteurs et caméras) et d'une ou plusieurs stations sol.

La mission envisagée est de type militaire. L'environnement comporte donc une zone ennemie dans laquelle on souhaite effectuer des opérations (recherche, observation, suivi de profil). Pour chaque opération, les informations récoltées sont transmises à une station sol soit en ligne, soit en fin d'opération, soit sur des points de transmission TW définis en limite de portée pour les opérations hors de portée des stations sol.

Le véhicule décolle d'un point origine, noté TOW, passe en zone ennemie, effectue des opérations sur plusieurs zones correspondant à des objectifs et transmet les informations recueillies. Il retraverse ensuite la frontière et atterrit sur un des points de fin

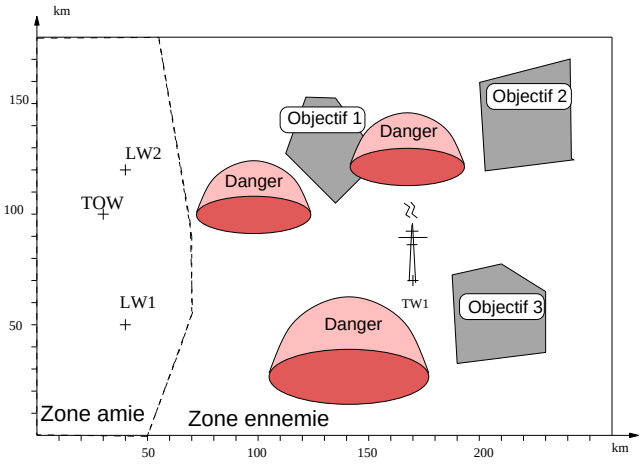


FIG. 1.8 – Exemple de carte de mission ; TOW origine, $LW1$ et $LW2$ fin, $TW1$ transmission

possibles, notés $LW1, LW2, \ldots$. La figure 1.8 illustre un exemple de carte de mission avec trois zones objectif et présence de menaces (demi-boules).

Les hypothèses formulées sont :
- Les zones sont supposées immobiles pour un calcul de planification. Une zone peut se déplacer dans le temps, mais ceci est pris en compte lors d'une replanification.
- Le problème est en trois dimensions mais il n'y a pas d'obstacle dû au relief. En effet les missions d'observation sont pour la plupart dédiées à des drones de type Moyenne Altitude Longue Endurance volant à des altitudes situées entre 5000 et 8000m.
- Les menaces sont indépendantes : elles n'ont pas la capacité de se coordonner pour poursuivre le véhicule.

Les contraintes de la mission sont liées aux objectifs (créneaux temporels pour l'arrivée sur les zones de mission et les points de transmission, priorités sur les objectifs, localisation), à l'environnement (risque d'être détruit, visibilité) et au système de drone (contraintes de vol, modèle de consommation de carburant, ...). Contrairement à certains travaux dans lesquels le véhicule vole à altitude et vitesse constantes [Richards & How 2002] et qui reviennent à guider le véhicule en deux dimensions seulement, ces variables évoluent à l'intérieur du domaine de vol de l'engin.

La fonction de planification a pour résultat une séquence de points de passage. Le suivi de cette séquence permet de réaliser les opérations qui sont les objectifs de la mission. La planification doit donc choisir et ordonner un sous-ensemble d'objectifs et doit déterminer la date d'arrivée sur chaque point, en maximisant les récompenses et en minimisant un critère de danger, de consommation et de durées tout en respectant les contraintes.

1.5.2 Discrétisation de l'espace

Une mission est modélisée par le point origine TOW en dehors de la zone ennemie, un ou plusieurs points de fin de mission LW en dehors de la zone ennemie et par la définition d'un ensemble de zones. Une zone peut être de deux types. La zone ennemie est définie par des points d'entrée $\{ENU\}$ et des points de sortie $\{EXU\}$. Les zones objectif sont situées dans la zone ennemie. Chaque zone objectif est définie par un ensemble de points d'entrée $\{ENO\}$ et un ensemble de points de sortie $\{EXO\}$. Un objectif est traité quand le véhicule est passé par un point d'entrée puis par un point de sortie de l'objectif. L'information ainsi récoltée peut être transmise soit en ligne, soit au point de sortie, soit sur un point de transmission TW défini en limite de portée pour les zones objectif dont les points de sortie sont hors de portée de la station sol. L'obtention de l'information permet d'accumuler le gain associé à l'objectif, parfois diminué par un manque de visibilité dû à la couverture nuageuse. Les points de fin de mission sont considérés comme des points de transmission : ils sont les points d'obtention des données non transmises. Un objectif est dit réalisé lorsque la récompense qui lui est associée a été obtenue. Des zones dangereuses sont définies à partir de la localisation de dangers potentiels plus ou moins bien connus. Des zones météo dynamiques évoluent dans la zone de mission. La bande de frontière avec la zone ennemie est définie comme une zone dangereuse. Le point de début du problème est le point origine pour la planification initiale ou la position courante du véhicule au moment de l'application du plan pour une replanification.

La discrétisation de l'espace permet de spécifier l'ensemble des points de passage dans lequel la planification définit son plan.

Une carte de mission incluant deux zones objectif, présentée figure 1.9, est choisie comme exemple illustratif.

Le choix des points n'est pas issu d'une méthode de construction de graphe. Les points sont donnés *a priori* par un expert du domaine. L'automatisation du choix des points n'est pas l'objet de ce travail.

1.5.3 Description haut niveau

L'ensemble N des nœuds correspond à l'ensemble des points de l'espace géométrique représenté figure 1.9. Il inclut le point de décollage TOW, l'ensemble des points d'atterrissage $\{LW\}$, les ensembles de points d'entrée et de sortie de la zone ennemie $\{ENU\}$ et $\{EXU\}$, pour chaque objectif un ensemble de points d'entrée $\{ENO_o\}$ et de points de sortie $\{EXO_o\}$, et enfin un point spécifique TW_1 pour la transmission des données.

La partition W est définie selon le type de chaque point. La figure 1.10 illustre la machine à état fini correspondant à la mission commençant au point de décollage. L'ensemble W_1 contient TOW en début de mission ou la position physique du véhicule quand le calcul de replanification est appliqué en cours de mission. W_e correspond à la fin du plan : il s'identifie à l'ensemble $\{LW\}$ des points d'atterrissage. La fonction de transition S est définie par une ou des trajectoires possibles entre deux ensembles de

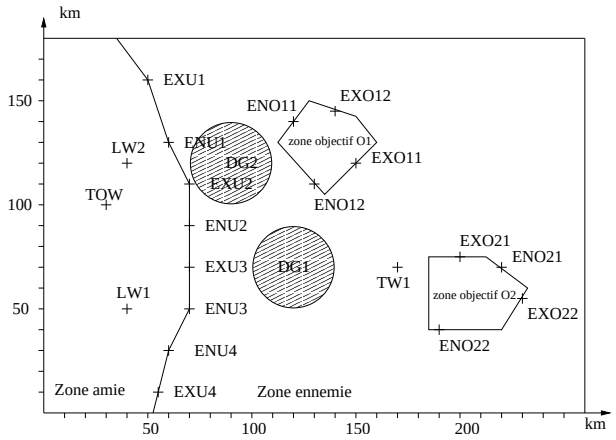


FIG. 1.9 – Carte de la mission de l'exemple illustratif après discrétisation de l'espace : TOW origine, LW fin, ENU entrée en zone ennemie, EXU sortie de la zone ennemie, ENO entrée en zone opération, EXO sortie de zone opération, TW point de transmission, DG zone de danger

points.

L'ensemble des objectifs est composé de la zone objectif $O1$ et de la zone objectif $O2$. La zone $O1$ est définie par le triplet $(W_3\,;\,W_4\,;\,W_4)$. En effet, les transmissions de données pour cette zone sont effectuées en sortie de zone. La zone $O2$ est définie par le triplet $(W_5; W_6; W_7)$. Les transmissions de données pour cette zone sont effectuées sur un point de transmission car la zone est hors de portée de la station sol.

☞ On remarque sur la figure 1.10 qu'il est possible de passer de W_4 à W_7 alors que les transmissions de données pour la zone $O1$ sont effectuées en sortie de zone. Les points TW_i sont utilisés dans ce cas comme des points de navigation.

Les ressources permettant d'accomplir la mission sont :
- l'engin ;
- son carburant embarqué ;
- sa charge utile.

La destruction de l'engin ne peut pas être prévue de manière déterministe. On sait simplement que la traversée d'une zone de danger implique un risque de destruction d'autant plus grand que la durée de traversée est élevée. On choisit donc de considérer l'engin au travers de sa probabilité de survie.
Le carburant peut être considéré soit directement, soit au travers de la masse du véhicule. En effet, la masse du véhicule est égale à la somme de la masse à vide et de la masse de carburant.
En ce qui concerne les capteurs composant la charge utile, on considère d'une part qu'ils ne risquent pas de tomber en panne et d'autre part qu'il y a suffisamment de place mémoire pour stocker l'informative relative à l'ensemble des objectifs de la mission. Il n'y

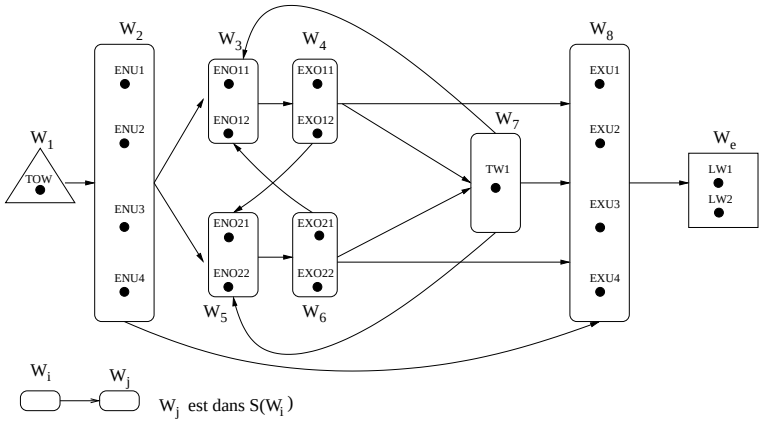


FIG. 1.10 – Machine à état fini de l'exemple illustratif en début de mission - Objectif $O1 : (W_3; W_4; W_4)$; Objectif $O2 : (W_5; W_6; W_7)$

a donc pas de modélisation de ces ressources.

Le vecteur de ressources $r \in \mathbb{R}^2$ contient donc la probabilité d'être vivant au temps courant (première composante) et la masse du véhicule au temps courant (seconde composante). La relation $r_e^s - r_{min}^s \geq 0$ (Equation 1.5) exprime deux contraintes. Le véhicule doit avoir une probabilité assez grande de terminer sa mission, c'est-à-dire que la probabilité d'être vivant à la fin de la mission est plus grande qu'une certaine valeur en deçà de laquelle on considère que le risque pris est trop élevé. La masse du véhicule à la fin de la mission doit être supérieure ou égale à la masse du véhicule à vide.

Le but de la planification est de trouver une séquence d'états commençant par le point d'application du plan, finissant par un élément de l'ensemble des points d'atterrissage et utilisant les trajectoires possibles entre deux ensembles de nœuds. La séquence doit minimiser la différence J entre les coûts résultant de l'utilisation des ressources et du temps passé et les récompenses obtenues lors de la transmission de données. Elle doit satisfaire les contraintes sur les ressources et sur le temps.

La récompense R_o pour avoir réalisé l'objectif o est une fonction de $t_{s(o)}$ et $r_{r(o)}$. Chaque objectif o a un gain maximum possible G_o. Ce gain est obtenu quand l'objectif a été traité et que l'information a été transmise, soit sur un point de transmission, soit en sortie de zone objectif, soit au fur et à mesure du traitement. Le modèle du gain utilisé est probabiliste de façon à prendre en compte la survie aléatoire de l'appareil jusqu'à la transmission $r^1_{r(o)}$ et le caractère probabiliste de la météo (visibilité aléatoire, donc gain dégradé). L'observabilité est modélisée par une probabilité d'observation $P_{Obs_o}(t_{s(o)})$ fonction de l'objectif o et de la date de début d'observation $t_{s(o)}$.
Le gain R_o, borné entre 0 et G_o, est ainsi obtenu par l'équation 1.16.

$$R_o = G_o.r^1_{r(o)}.P_{Obs_o}(t_{s(o)}) \tag{1.16}$$

1.5.4 Description bas niveau

Le vecteur ressources

Le vecteur des ressources $r = \begin{bmatrix} r^1 \\ r^2 \end{bmatrix}$ contient :

- la probabilité d'être vivant au temps courant r^1 ;
- la masse du véhicule au temps courant r^2.

Le danger est considéré comme une perturbation à laquelle est soumis le système. Dans le modèle présenté par Allo, Guettier & Lecubin [Allo, Guettier, & Lecubin 2001], le danger est une caractéristique statique d'arc. Dans cet exemple, le danger est une caractéristique dynamique. Un modèle probabiliste est choisi. Ce modèle reste modifiable et adaptable suivant les situations d'environnement et de menaces traitées.

On fait l'hypothèse que les dangers ne sont liés qu'aux menaces de type missiles sol-air. Le modèle choisi pour les menaces de ce type prend en compte :

- pm, la probabilité d'existence de la menace, donnée de l'environnement. pm pourrait dépendre de la date à laquelle la menace est traversée par le drone. Pour plus de simplicité, on considère ici qu'elle est constante dans le temps.
- pt, la probabilité de détection du drone par la menace (Figure 1.11). Elle dépend de la durée de vol dans la menace, notée d_m, mais pas de la menace (Equation 1.17).

$$pt(d_m) = \frac{1}{1+e^{-c.d_m-d}} \tag{1.17}$$

$$\text{avec :} \quad c = 2.ln(\tfrac{0.99}{0.01})\tfrac{1}{D_{max}-D_{min}}$$

$$d = ln(\tfrac{0.01}{0.99})\tfrac{D_{max}+D_{min}}{D_{max}-D_{min}}$$

Où : D_{min} est la durée avant laquelle le drone a moins de 1 % de risque d'être repéré par la menace ; D_{max} la durée après laquelle le drone a plus de 99 % de risque d'être repéré par la menace.

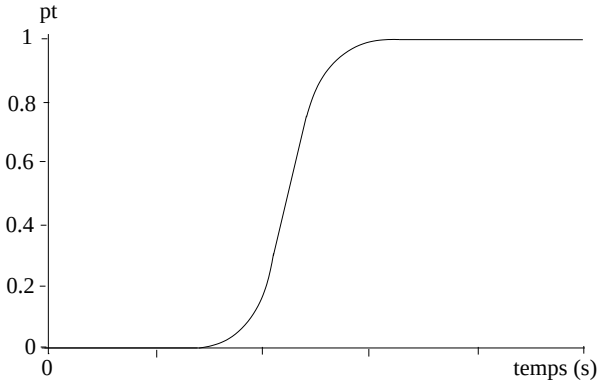


FIG. 1.11 – Modélisation de la probabilité de détection

- pa, la probabilité de touche. Comme la vitesse du missile est très grande par rapport à la vitesse du drone et sa maniabilité meilleure, la probabilité de touche

est très proche de 1. De plus, le cas où le missile abîme l'engin est négligé : une touche équivaut à la destruction du drone.

La figure 1.12 schématise la probabilité de survie à une menace de type missile.

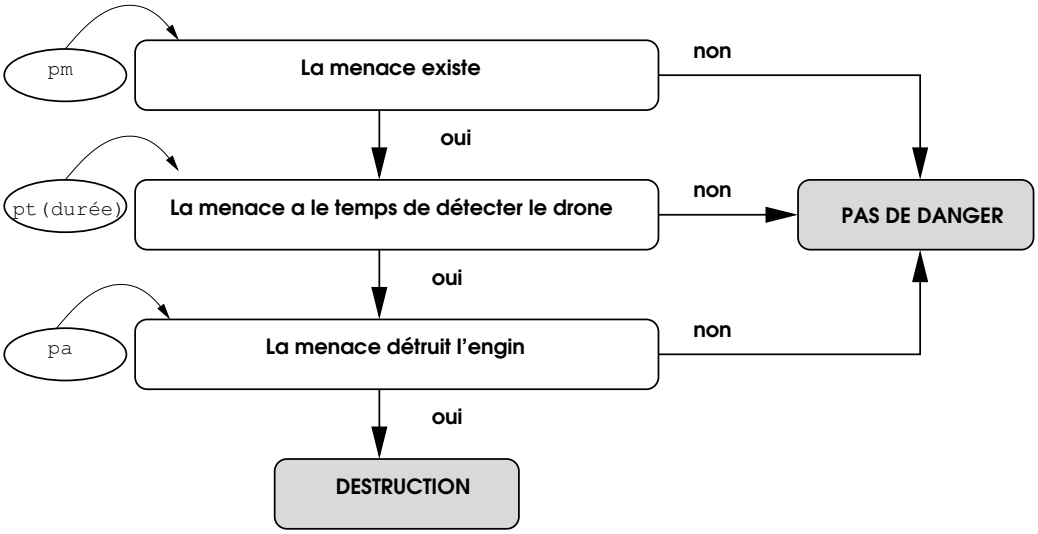


FIG. 1.12 – Probabilité de survie

Une menace est modélisée dans l'espace par une demi-boule de danger posée au sol et centrée sur l'emplacement le plus probable de la base de lancement du missile. Le drone peut donc entrer et sortir d'une menace plusieurs fois pendant son itinéraire. Un ensemble {entrée-sortie} pour une menace m est appelé une exposition. En supposant que les expositions aux menaces sont indépendantes, on montre que :

$$r_e^1 = \prod_{m \in S_M} \prod_{e_m \in E_m} \left(1 - pm_m.pa_m.pt\Big(\sum_{k/(n_k,n_{k+1}) \in I_m} \Delta_{n_k,n_{k+1}}^{a_{k+1}} . \frac{L(n_k, n_{k+1}|m)}{L(n_k, n_{k+1}, a_{k+1})}\Big)\right) \tag{1.18}$$

où S_M est l'ensemble des menaces ; E_m l'ensemble des expositions dues à la traversée d'une menace m par l'itinéraire ; I_m ensemble des arcs (n_k, n_{k+1}) de l'itinéraire interceptant la menace m pendant l'exposition e_m ; $L(n_k, n_{k+1}|m)$ la longueur de l'arc à l'intérieur de la menace m ; $\Delta_{n_k,n_{k+1}}^{a_{k+1}}$ la durée de parcours de l'arc (n_k, n_{k+1}) de longueur $L(n_k, n_{k+1}, a_{k+1})$.

La modélisation du danger implique que la probabilité d'être vivant ne peut que décroître ou rester constante au cours du temps. On peut montrer que, même si le coût en danger n'est pas décomposable sur les arcs de l'itinéraire, le coût relatif de la probabilité courante d'être vivant peut être considéré comme déjà consommé. Le calcul de r_e^1 ne contient que des expositions complètes à une menace, c'est-à-dire qu'il ne contient que des ensembles complets {entrée-sortie} de menace. Il est nécessaire de

pouvoir calculer un coût partiel sur le coût du danger. Si le nœud courant n_i est à l'extérieur d'une menace, alors le coût $r^1_{\pi(i)}$ est exprimé par l'équation 1.18 avec E_m l'ensemble des expositions dues à la traversée d'une menace m par l'itinéraire courant. Lorsque le nœud courant n_i est à l'intérieur d'une menace m^i, on note I^i_m l'ensemble des arc de l'itinéraire composant l'exposition courante et la formule 1.19 s'applique.

$$r^1_{\pi(i)} = \Big(1 - pm_{m^i}.pa_{m^i}.pt(\sum_{k/(n_k,n_{k+1})\in I^i_m} \Delta^{a_{k+1}}_{n_k,n_{k+1}}.\frac{L(n_k,n_{k+1}|m^i)}{L(n_k,n_{k+1},a_{k+1})})\Big).$$
$$\prod_{m\in S_M}\prod_{e_m\in E_m}\Big(1 - pm_m.pa_m.pt(\sum_{k/(n_k,n_{k+1})\in I_m} \Delta^{a_{k+1}}_{n_k,n_{k+1}}.\frac{L(n_k,n_{k+1}|m)}{L(n_k,n_{k+1},a_{k+1})})\Big) \quad (1.19)$$

La masse du véhicule au temps courant dépend de sa consommation en carburant. Cette consommation est calculable pour chaque arc de l'itinéraire : la masse de carburant est une ressource décomposable sur l'itinéraire. Etant donnés deux nœuds de l'itinéraire $n_k \in W_k$ et $n_{k+1} \in W_{k+1}$ avec $W_{k+1} \in S(W_k)$, et a_{k+1} l'action permettant d'atteindre n_{k+1}, le calcul de la consommation entre n_k et n_{k+1} est basé sur l'altitude moyenne de vol $h_{k,k+1}$ entre les nœuds et sur la durée de parcours de l'arc $\Delta^{a_{k+1}}_{n_k,n_{k+1}}$ (équivalent vitesse), dont on simplifiera ici la notation en h et Δ. D'après [Boiffier 2001], la poussée T est donnée par :

$$\begin{aligned} T(h,\Delta) = &\quad \alpha(h).\tfrac{1}{\Delta^2} + \beta(h).\Delta^4 \\ \text{avec :} &\quad \alpha(h) = \tfrac{1}{2}\rho(h)S.Cd_0.L^2 \\ \text{et :} &\quad \beta(h) = \tfrac{K(m_a.g)^2}{\frac{1}{2}\rho(h)SL^4} \end{aligned} \quad (1.20)$$

où ρ est la densité de l'air, L la longueur de la trajectoire entre n_k et n_{k+1}, S la surface du plan aérodynamique principal de l'avion, Cd_0 le coefficient de traînée à portance nulle, K le coefficient de traînée induite, m_a la masse de l'avion supposée constante sur un arc, g la gravité supposée constante et égale à g_0. On néglige ici l'effet du second ordre qui fait qu'un avion avec moins de carburant consomme moins.
La contrainte à respecter sur la poussée est :

$$\begin{aligned} T(h,\Delta) \le T_{max} = &\quad T_0(\rho(h)/\rho_0)(1 - M + \tfrac{M^2}{2}) \\ \text{avec :} &\quad M = \frac{L}{\Delta\sqrt{\frac{\gamma RT_p}{M_o}}} \end{aligned} \quad (1.21)$$

où T_0 est la poussée maximale au sol et à vitesse nulle, ρ_0 est la densité de l'air au sol, $\gamma = \frac{7}{5}$; $R = 8.314\ JK.mol^{-1}$ pour l'air ; $M_o = 29.10^{-3} kg.mol^{-1}$ et T_p la température absolue, soit $288 - 6,5.10^{-3} \times h$.
Or la consommation est approximativement égale à :

$$C_{SR}.T(h,\Delta).\Delta$$

où C_{SR} (en $kg.s^{-1}.N^{-1}$) est la consommation spécifique permettant d'évaluer la consommation en carburant du moteur en fonction de la poussée.
On conclut que la masse courante du véhicule $r^2_{\pi(i)}$, calculée au nœud n_i choisi dans le sous-ensemble $W_{\pi(i)}$ de la séquence Q, est donnée grâce à l'équation :

$$\left\| r^2_{\pi(i)} = m_1 - \sum_{k=1}^{i-1} C_{SR}.T(h_{k,k+1}, \Delta^{a_{k+1}}_{n_k,n_{k+1}}).\Delta^{a_{k+1}}_{n_k,n_{k+1}} \right. \tag{1.22}$$

où m_1 est la masse du véhicule à la date d'application du plan ; $m_1 = m_0$ au début de la mission.

La consommation de carburant implique que la masse du véhicule diminue toujours au cours du temps.

La fonction de coût du temps et des ressources

Le coût en durée est donné par l'équation 1.23.

$$P_{vol}.(t_e - t_1) \tag{1.23}$$

où P_{vol} est le prix d'une unité de temps vol.

La fonction de coût des ressources R_e se décompose sur les deux composantes du vecteur ressources.

Le coût sur le danger est une fonction de r^1_e donnée par l'équation 1.24.

$$P_{sys}.(1 - r^1_e) \tag{1.24}$$

où P_{sys} est le prix de l'appareil et de sa charge utile. La valeur 1 indique qu'au moment où le plan est appliqué, le système de drone n'est pas détruit.

Le coût en carburant sur l'itinéraire est donné par l'équation 1.25.

$$P_{carbu}.(m_1 - r^2_e) \tag{1.25}$$

où P_{carbu} est le prix du carburant par unité de masse consommée.

En règle générale, le coût à l'instant courant d'une ressource r est donc égal à $K.(r_1 - r)$, avec K une constante réelle positive, r_1 le niveau de la ressource au moment où le plan est appliqué et r son niveau à l'instant courant. On écrit :

$$\left\| R_e(r_e) = (t_e - t_1).P_{vol} + (r_1 - r_e)^\top.\mathbf{C} \right. \tag{1.26}$$

où $\mathbf{C}$ est un vecteur de même dimension que r, ici $\mathbb{R}^2$, de composante P_{sys} et P_{carbu} ; r_1 est le vecteur ressources à la date d'application du plan.

La fonction critère

La séquence Q doit minimiser la différence J entre les coûts résultant de l'utilisation des ressources et les récompenses obtenues lors de la transmission de données. Elle doit satisfaire les contraintes sur les ressources. La fonction J est donc donnée par l'équation 1.27.

$$J = (t_e - t_1).P_{vol} + (r_1 - r_e)^\top.\mathbf{C} - \sum_{o \in E_o} G_o.r^1_{r(o)}.P_{Obs_o}(t_{s(o)}) \tag{1.27}$$

E_o est l'ensemble des objectifs réalisés, c'est-à-dire qu'il existe i tel que $W_{\pi(i)} = W_{s(o)}$ et $W_{\pi(i+1)} = W_{e(o)}$ et qu'il existe $j > i$ tel que $W_{\pi(j)} = W_{r(o)}$. J est décroissante en fonction des ressources.

Les incertitudes sur le déroulement du plan (existence de menace, détection, échec d'observation, ...) sont donc intégrées dans le critère d'optimisation.

Calcul des longueurs de trajectoires suivant le choix des actions

Le but de ce paragraphe est, pour toutes les actions envisagées entre deux nœuds $n_i(x_i, y_i, z_i)$ et $n_j(x_j, y_j, z_j)$ de N, de calculer la distance de parcours L_p.

Différentes actions de mouvements sont possibles pour atteindre un nœud de N. Deux modes de navigation sont définis.

Le premier mode de navigation est la ligne droite. En présence ou non de danger, L_p est alors égale à la distance géométrique entre les deux points.

$$L_p = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2} \tag{1.28}$$

S'il y a un danger sur cette ligne droite, le véhicule peut soit traverser la demi-boule, soit la contourner.

Dans le cas du contournement, notons $C(x_c, y_c, z_c)$ le centre de la boule incluant la demi-boule de danger. Soit $A(x_a, y_a, z_a)$ et $B(x_b, y_b, z_b)$ les deux points de l'itinéraire qui interceptent la demi-boule (Figure 1.13). La valeur de la distance entre A et B, notée d, est donnée par l'équation 1.29.

$$d = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2} \tag{1.29}$$

On peut montrer que l'arc de longueur minimale entre A et B appartient au plan contenant A, B et C.

D'après la figure 1.14, la longueur minimale recherchée L nécessaire pour contourner la zone entre A et B est donnée par l'équation $L = R.\alpha$ où R est le rayon de la demi-boule et α l'angle formé par l'intersection de CA et CB. Or :

$$d^2 = 2R^2(1 - \cos(\alpha)) \tag{1.30}$$

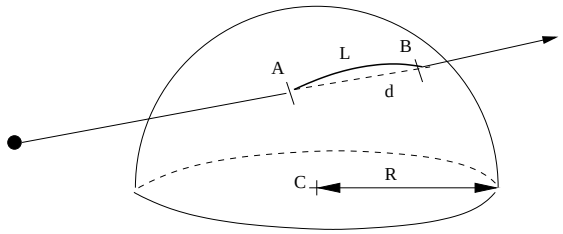


FIG. 1.13 – Intersection d'une demi-boule et d'une droite

Donc :

$$L = R.\arccos(\frac{2R^2 - d^2}{2R^2}) \tag{1.31}$$

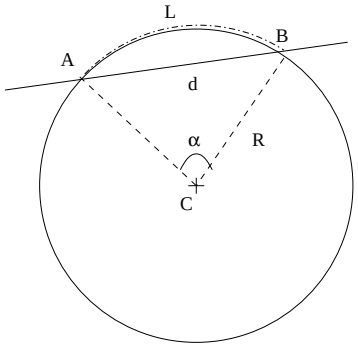


FIG. 1.14 – Recherche de la longueur minimale L de contournement de la demi-boule

De façon à rendre la modélisation plus réaliste, on rajoute 20% de marge à la distance calculée pour contourner la zone dangereuse. Le véhicule peut donc atteindre le point suivant en survolant une distance :

$$\left\| \quad L_p = 1,2.(\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2} - d + L) \right. \tag{1.32}$$

L'opération choisie pour le traitement d'un objectif entre un point d'entrée et un point de sortie d'une zone objectif est une action de balayage. Le balayage en lignes parallèles, plus simple et plus rapide à réaliser, est choisi. La figure 1.15 illustre ce type de balayage. Notons D_o la distance géométrique maximum entre un point d'entrée et un point de sortie de l'objectif o. Soit r_b la largeur des raies de balayage, la distance parcourue par le véhicule pour traiter l'objectif est prise égale à :

$$\left\| \quad L_p = (\lfloor \frac{D_o}{r_b} \rfloor + 1).D_o + (D_o - \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2}) \right. \tag{1.33}$$

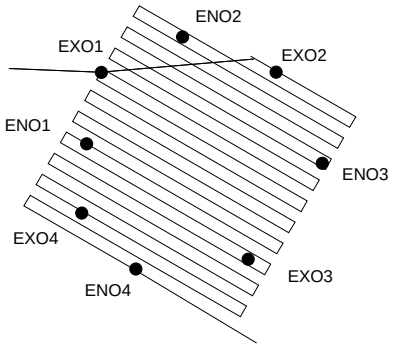


FIG. 1.15 – Balayage simple en survol d'une zone objectif

Deux nœuds peuvent donc être reliés par trois chemins de longueurs différentes avec des contraintes de vol différentes concernant les vitesses minimum et maximum sur le tronçon. Le domaine de définition de $\Delta^{a_j}_{n_i,n_j}$ est donc fonction des nœuds choisis et du mode de navigation utilisé.

Contraintes sur l'itinéraire

Le problème de planification possède des contraintes linéaires et non linéaires.

Pour cet exemple illustratif, les contraintes sur les ressources en fin de mission de l'équation 1.5 correspondent à deux inégalités linéaires données par le système 1.34.

$$\begin{array}{rcl} r^1_e - r^1_{min} & \geq & 0 \\ r^2_e - r^2_{min} & \geq & 0 \end{array} \tag{1.34}$$

où r^1_{min} et r^2_{min} sont les deux composantes du vecteur r_{min}. r^1_{min} est la valeur en-deçà de laquelle le risque pris est considéré comme trop élevé pour le véhicule. r^2_{min} correspond à la masse du système de drone sans carburant. Notons que, bien que ces inégalités soient linéaires en r, elles sont non linéaires en Δ. En effet, les équations 1.18 et 1.22 sont non linéaires.

Le problème possède en outre des contraintes linéaires sur les durées.

Le premier type de contraintes linéaires est lié aux caractéristiques de l'avion. Soient deux nœuds de l'itinéraire $n_i \in W_i$ et $n_j \in W_j$ avec $W_j \in S(W_i)$, soit $h_{i,j}$ l'altitude moyenne de vol entre les nœuds et $\Delta^{a_j}_{n_i,n_j}$ (équivalent vitesse) la durée de parcours de l'arc, simplifiés ici en h et Δ. En utilisant la projection des forces aérodynamiques, on peut calculer les vitesses minimale et maximale entre les deux nœuds n_i et n_j de l'arc en résolvant les équations de portance 1.35 et 1.36. Les paramètres ont été définis avec la poussée moteur (Equation 1.20).

$$\begin{array}{c} \max Cl \text{ avec } 0 \leq Cl \leq Cl_{max} \\ \text{sous } 0 \leq m_a.g\Big(\frac{Cd_0+KCl^2}{Cl} + \sigma\Big) \leq T_{max}(h) \end{array} \tag{1.35}$$

La solution du problème 1.35 est noté $Cl_{S_{min}}$.

$$\begin{aligned} & \min Cl \text{ avec } 0 \leq Cl \leq Cl_{max} \\ \text{sous} \quad & 0 \leq m_a.g\Big(\frac{Cd_0+KCl^2}{Cl} + \sigma\Big) \leq T_{max}(h) \\ \text{et} \quad & \sqrt{\frac{2.m_a.g}{\rho(h).S.Cl}} < VNE \end{aligned} \tag{1.36}$$

La solution du problème 1.36 est noté $Cl_{S_{max}}$.

Puis on calcule les vitesses minimale et maximale par :

$$\begin{aligned} v_{min} &= \frac{2.m_a.g}{\rho(h)SCl_{S_{min}}} \\ v_{max} &= \frac{2.m_a.g}{\rho(h)SCl_{S_{max}}} \end{aligned} \tag{1.37}$$

où σ est la pente entre l'origine et l'extrémité de l'arc, Cl le coefficient de portance et Cl_{max} sa valeur maximum, T_{max} est la poussée maximum et VNE la vitesse (Never Exceed) à ne jamais dépasser. La VNE conduit irrémédiablement à une dégradation grave de l'avion ou à sa destruction.

Le second type de contraintes linéaires est lié au graphe des données : t_{min} et t_{max} contraignent la date d'arrivée sur certains points de N. Pour les points d'entrée et de sortie de la zone ennemie, les fenêtres temporelles proviennent des intervalles pendant lesquels le franchissement de la frontière par ces points est possible. En effet, la zone frontière est infranchissable, hormis pendant les fenêtres temporelles où les points d'entrée et de sortie sont définis. Pour chaque objectif, la fenêtre temporelle indique le moment où l'observation est valide.
Par ailleurs, la mission doit se terminer avant la date de fin au plus tard de l'ensemble des nœuds de W_e : cette contrainte est exprimée par l'équation 1.8 sur tous les nœuds. En effet, aucun nœud ne peut avoir une date de passage postérieure à la date d'atterrissage au plus tard.

☞ La zone frontière est modélisée par une zone de danger linéaire, sur le même modèle que les menaces avec $pm = 1$, $pt = 1$, $pa = 1$, de sorte qu'elle est infranchissable, puisque dès qu'elle est franchie, le système est considéré comme détruit.

La date d'application du plan t_1 étant connue, les contraintes linéaires définissent pour chaque point de N une fenêtre temporelle donnée par l'équation 1.8.

1.6 Conclusion

Le problème de planification de mission pour un véhicule a été formalisé d'une manière relativement générique. Cette formalisation fait intervenir deux niveaux de description. Le premier niveau est relatif à la sélection des objectifs et le second

à leur réalisation, liée aux choix des nœuds, des actions et des vitesses entre les nœuds.

La pertinence de ce modèle générique a été montrée au travers de son paramétrage pour l'application à une mission militaire d'observation menée par un véhicule aérien autonome.

Par ailleurs, l'analyse de la complexité du problème indique qu'une exploration complète des alternatives est extrêmement coûteuse en terme de temps de calcul. Il est donc nécessaire d'envisager des méthodes d'exploration arborescentes adaptées.

2 Les algorithmes de planification

Résumé : Le problème de planification de mission est résolu en utilisant un graphe décrit dans le chapitre précédent. Les arcs du graphe indiquent une action de bas niveau possible entre deux nœuds. Ce chapitre expose un cadre algorithmique, basé sur l'algorithme A^*, permettant d'effectuer une recherche d'itinéraire sur l'arborescence des chemins possibles. A chaque développement de nœud, l'optimisation d'une partie du critère J est effectuée, en prenant en compte les contraintes non linéaires induites par l'expression de certaines ressources en fonction des durées et en gérant les contraintes linéaires sur les durées.
Différentes méthodes sont envisagées pour rechercher une première solution admissible, évaluer le coût d'un nœud courant à un nœud fin, élaguer l'arbre d'exploration et ranger les nœuds lors de l'exploration.

2.1 Introduction

Le cadre formel décrivant le problème de planification de mission permet de développer des algorithmes de planification pour résoudre le problème.

Pour résumer le chapitre précédent, rappelons les entrées du problème sur lesquelles vont s'effectuer le calcul du plan :

- Un graphe de bas niveau constitué d'un ensemble de nœuds, noté N. n_1 est le nœud origine : il correspond au nœud où le plan est appliqué. Il existe un sous-ensemble de N de nœuds destination, noté W_e. Les arcs du graphe représentent les actions possibles entre deux nœuds. Une relation S donne l'ensemble de successeurs de chaque nœud du graphe.
- Un graphe de haut niveau, construit à partir d'une partition de N notée W. Il permet de décrire la réalisation des objectifs de la mission, qui donne lieu à une récompense R_o.
- Les données des fonctions de ressources du système. Les ressources sont fonction du temps, des actions et chemins choisis dans les deux graphes et de l'environnement.
- Les contraintes temporelles sur chaque nœud de N, les contraintes sur les ressources.

D'un point de vue algorithmique, on peut noter que la recherche arborescente et l'algorithme A^* [Nilson 1971] sont utilisés à la fois pour l'optimisation dans les graphes

et dans le cadre du formalisme état-action. Il semble donc intéressant de se baser sur ce type d'algorithmes pour traiter des problèmes de planification de mission. L'idée proposée dans ce travail est d'effectuer une exploration dans l'arborescence décrivant les chemins possibles, construite à partir du graphe de bas niveau. A chaque développement de nœud, le graphe de haut niveau permet la détermination des récompenses accumulées le long du chemin et le graphe de bas niveau permet le calcul des coûts relatifs au temps et aux ressources.

2.2 Rappel sur l'algorithme A^* et ses dérivés

☞ Ce rappel s'appuie sur des notions de la théorie des graphes. On pourra se reporter à l'annexe A pour plus de détails sur les termes indicés(1).

L'algorithme **A* (A-star ou A-étoile)** reprend l'algorithme de Dijkstra(1) en ajoutant une évaluation du coût futur pour guider la recherche. Au lieu de placer les nœuds dans la liste des pendants(1) $listeP$ en fonction de leur potentiel(1) g_i, ils sont placés en fonction de la somme f_i de leur potentiel et d'une estimation h_i de la distance pour atteindre le nœud de destination le plus proche.

$$f_i = g_i + \mu h_i$$

Avec μ une constante donnant l'importance ou la confiance en l'évaluation h_i.

Algorithme A^*

début
- Initialisation :
- Placer n_1 dans $listeP$
- **tant que** $listeP$ *n'est pas vide* **faire**
  - Traitement :
  - Traiter $\hat{u}$: dans le graphe, effectuer tous les déroutements pour atteindre $\hat{u}$
  - **pour** *chaque* n_i *dans* $S(\hat{u})$ **faire**
    - Calculer h_i
  - Rangement des suivants dans $listeP$:
  - Mettre les éléments de $S(\hat{u})$ qui ne sont pas dans $listeP$ en rangeant les nœuds par $f_i = g_i + \mu h_i$ croissant
  - Mettre $\hat{u}$ dans $listeQ$
  - Effacer $\hat{u}$ de $listeP$

fin

NOTATIONS : $listeP$ est la liste des nœuds pendants ; $listeQ$ est la liste des nœuds déjà développés ; $\hat{u}$ est le premier élément de $listeP$ et $S(\hat{u})$ est l'ensemble des successeurs(1) de $\hat{u}$ dans le graphe.

L'estimation h_i est obtenue en effectuant une relaxation des contraintes. Certaines méthodes de recherche de plus court chemin passent outre les obstacles, d'autres négligent les contraintes temporelles, ou les contraintes de ressources. Les méthodes basées sur le formalisme état-action consistent parfois à ne pas tenir compte des faits détruits ou à développer un graphe de planification ou de navigation, comme on l'a vu dans la section 1.2.2. L'algorithme A^* permet de se concentrer sur les nœuds qui ont le plus de chance d'aboutir à une solution optimale. La recherche se fait vers un des nœuds destination tout en conservant une approche optimale.

Pour des environnements statiques, les algorithmes de recherche de plus court chemin de type A^* ou Dijkstra sont les mieux adaptés. On trouve dans la littérature de nombreux exemples d'application de ces algorithmes en robotique mobile [L.Kavraki *et al.* 1994], [Pai & Reissel 1998], [Pettersson & Doherty 2004].

PROPRIÉTÉ 4 *Pour tout graphe orienté G fini ou infini dans lequel les arcs ont des coûts positifs ou nuls et où il existe au moins un chemin de n_1 à un nœud destination, et pour toute fonction h minorant les coûts du nœud courant à un nœud destination, l'algorithme A^* est* ***admissible*** *: il s'arrête en fournissant un chemin optimal entre n_1 et un nœud destination.*

Pour les environnements dynamiques, et lorsque l'on a une connaissance incomplète de l'environnement en début de mission, des variantes du A^* ont été développées.

L'**algorithme** D^* [Stentz 1994] est la base de tous les algorithmes dynamiques, dans le sens où le coût des arcs change au cours de la résolution du problème. Il résout des problèmes où l'ensemble destination est réduit à un nœud. L'algorithme D^* consiste principalement en deux fonctions. La première est utilisée pour calculer les coûts de l'itinéraire optimal jusqu'au nœud destination. Elle est appelée de manière itérative jusqu'à ce qu'un itinéraire ait été trouvé ou que l'algorithme retourne qu'il n'y a pas de solution. Le véhicule suit alors l'itinéraire trouvé jusqu'à ce que le nœud destination soit atteint ou qu'une erreur dans le coût d'un arc soit détectée. Dans ce cas, la seconde fonction est appelée pour corriger les coûts (phase de réparation) et placer les nœuds affectés dans la liste des pendants. Le changement de coût est propagé jusqu'à ce que tous les nœuds dont la fonction d'évaluation est supérieure à celle du nœuds où le changement a été détecté aient disparu. On recalcule ensuite un nouvel itinéraire optimal.
Des améliorations et des variantes de cet algorithme ont été développées par la suite : Focussed D^* [Stentz 1995] améliore le D^* en se concentrant sur la partie réparation ; Constrained D^* [Stentz 2002] replanifie une solution optimale soumise à une contrainte globale ; Delayed D^* [Ferguson & Stentz 2005] améliore les temps de calcul ; et récemment Anytime Dynamic A^* [Likhachev *et al.* 2005] produit des solutions sous-optimales bornées d'une manière anytime : contrairement à un algorithme standard, où aucun résultat n'est accessible avant la fin de son exécution, l'algorithme anytime peut être stoppé à tout instant, et doit fournir des solutions de qualité croissante au cours du temps.

Le **Lifelong Planning** A^* **(**LPA^***)** [Koenig, Likhachev, & Furcy 2004] est une version incrémentale du A^* qui trouve de manière répétée des plus courts chemins d'un

nœud origine à un nœud destination lorsque le coût des arcs change, ou que des arcs sont ajoutés ou enlevés au graphe. L'idée est d'effectuer une première recherche identique au A^*, puis, lors des calculs suivants de réutiliser les parties de la recherche identiques à la recherche déjà effectuée.

Ces algorithmes, très efficaces pour des environnements mal connus dont la dynamique est peu rapide, correspondent souvent à un A^* lorsque les changements observés sur l'environnement sont trop importants. De fait, pour le problème de planification qui nous intéresse, les changements à prendre en compte seront de type nouvel objectif ou modification importante des contraintes sur les ressources. Les algorithmes de type D^* et LPA^* ne sont donc pas envisagés.

2.3 Le cadre algorithmique

2.3.1 L'algorithme de base : vue d'ensemble

L'algorithme de base (Algorithme 3) est inspiré de l'algorithme A^*. Par rapport au A^*, l'algorithme proposé prend en compte des coûts pouvant être négatifs sur les arcs et le fait que l'inégalité triangulaire n'est pas satisfaite.

☞ Rappel : l'inégalité triangulaire est une propriété sur les distances d : quels que soient x, y et z, $d(x,z) > d(x,y) + d(y,z)$ (i.e. pour aller d'un point à un autre directement, le trajet n'est jamais plus long que si l'on faisait une étape).

NOTATIONS : n_1 nœud d'application du plan ; $BORNE$ valeur optimale du critère pour l'itinéraire de n_1 à un nœud fin ; J critère ; $listeP$ liste des nœuds pendants, initialement vide ; n_i nœud ; $S(\hat{u})$ ensemble des successeurs de $\hat{u}$, premier nœud de $listeP$; $listeT$ une liste de nœuds, initialement vide ; a action ; n_i^a indique que le nœud n_i est atteint avec l'action a ; h évaluation du critère du nœud courant à un nœud fin ; I critère partiel, de valeur optimale $\hat{I}$.

La première idée est de rechercher une première solution admissible de manière naïve (**Ligne 1**), en considérant le problème sans ses contraintes et en ne développant qu'un nombre limité de nœuds. Si une première solution est trouvée, elle permet d'obtenir une valeur maximale pour les critères admissibles (**Ligne 2-3**). Cette valeur maximale est appelée $BORNE$. Dans le cadre d'une replanification nécessitant une réponse très rapide, la recherche d'une première solution admissible permet souvent d'obtenir une solution dans le temps imparti.

Le nœud origine, noté n_1, est ensuite placé dans la liste des nœuds pendants $listeP$, initialement vide (**Ligne 4**). Le processus itératif traite tour à tour tous les éléments arrivant en tête de la liste des nœuds pendants ; ce premier élément est noté $\hat{u}$.
Le traitement d'un nœud doit être adapté au problème de planification de mission. Pour chaque nœud développé, on effectue l'optimisation précise de la valeur du critère (**Ligne 11**) pour le plan allant de n_1 au nœud traité. La valeur trouvée est notée $\hat{I}$. Cette

Algorithme 3 : Algorithme de planification de base

```
début
    Recherche d'une première solution admissible
    si une première séquence admissible a été trouvée alors
        Initialiser BORNE avec le J trouvé
    Initialisation :
    Placer n_1 dans listeP
    tant que listeP n'est pas vide faire
        Traitement :
        pour chaque n_i dans S(û) faire
            listeT = ∅
            pour chaque action a possible faire
                Construire la séquence de n_1 à n_i^a
                Optimiser I en choisissant t pour chaque nœud en respectant les
                contraintes
                si Il y a une solution alors
                    Calculer h de n_i^a à un nœud fin
                    Stocker dans n_i^a : Î et h
                    Ajouter n_i^a dans listeT
                    si n_i ∈ W_e et Î < BORNE alors BORNE ← Î
            Élaguer l'arbre d'exploration
        Rangement des suivants dans listeP :
        Mettre les éléments de listeT dans listeP
        Effacer û de listeP
fin
```

recherche permet d'élaguer l'arbre en cas d'impossibilité au niveau des contraintes sur les ressources ou au niveau des contraintes temporelles. Elle requiert une optimisation des dates de passage sur chaque point de l'itinéraire : pour chaque nœud développé, un sous-problème d'optimisation est résolu. D'après la nature du problème de planification, le nombre d'objectifs réalisés dans le plan n'est pas fixé *a priori*. L'ensemble E_o des objectifs réalisés dans la plan reste donc inconnu tant que l'ensemble W_e n'est pas atteint dans le développement. Par contre, au nœud courant n_i, la connaissance partielle de la séquence Q des états jusqu'à l'ensemble $W_{\pi(i)}$ permet de connaître l'ensemble des objectifs réalisés avant n_i, noté E_o^i. Il est alors possible de définir un critère partiel de n_1 à n_i, noté I.

A ce stade du développement, les nœuds décrivant le plan sont choisis, ainsi que les actions pour les atteindre. Le but est donc d'optimiser le critère partiel I par le choix des durées $\Delta^{a_2}_{n_1,n_2}, \ldots, \Delta^{a_i}_{n_{i-1},n_i}$ en respectant :

- les contraintes temporelles linéaires : les contraintes égalité entre $t_{\pi(k)}$, $t_{\pi(k+1)}$ et

$\Delta^{a_{k+1}}_{n_k,n_{k+1}}$, les contraintes inégalité sur $t_{\pi(k)}$;
- les contraintes linéaires sur le temps, induites par les contraintes $r^s_{\pi(i)} - r^s_{min} \geq 0$, pour les ressources dont la consommation s'exprime linéairement par rapport à Δ (Equations 1.11 et 1.13).
- les contraintes non linéaires sur le temps, induites par les contraintes $r^s_{\pi(i)} - r^s_{min} \geq 0$ pour les ressources dont la consommation ne s'exprime pas linéairement par rapport à Δ.

Ce problème est transformé en une optimisation d'un critère non linéaire sous contraintes linéaires. Il est résolu par un algorithme basé sur Frank-Wolfe [Frank & Wolfe 1956]. Si le problème a une solution de valeur $\hat{I}$ (**Ligne 12-15**), alors l'évaluation h du critère du nœud n_i à un nœud fin est calculée et les valeurs $\hat{I}$ et h sont stockées dans la structure du nœud développé. Le nœud traité est conservé dans une liste notée $listeT$. Si le nœud traité est un nœud fin et que le critère optimisé est meilleur que la borne, alors la valeur de la borne est mise à jour (**Ligne 16**).
Une fois tous les nœuds suivants développés, on effectue un élagage (**Ligne 17**). Cet élagage sert à limiter le nombre d'états à stocker et répond à des contraintes d'espace mémoire. Les fonctions d'élagage de la littérature ne sont pas adaptées à notre problème, car le critère n'est pas une simple somme sur les arcs du chemin développé et peut augmenter ou diminuer lors du développement d'un nœud suivant. Il faut donc développer des fonctions d'élagage spécifiques.

2.3.2 Recherche d'une première solution admissible

Le calcul rapide d'un premier itinéraire admissible augmente la réactivité du système et l'efficacité de l'élagage en donnant une première borne utilisée ensuite dans l'algorithme de recherche. La méthode décrite dans l'algorithme 4 calcule un premier itinéraire sans optimisation des dates de passage et en développant un nombre limité de nœuds. Seules les meilleures séquences partant de n_1 et arrivant à un nœud fin sont optimisées au niveau des dates de passage sur chaque nœud. Le résultat de l'algorithme 4 est la meilleure séquence optimisée.

Commentons cette recherche.
(**Ligne 1**) Si la planification est une préparation de mission, la borne est initialisée à zéro. En effet, si le plan trouvé ne rapporte pas plus qu'il ne coûte, il paraît inutile de commencer la mission. Par contre, si l'algorithme est utilisé en replanification, il est possible que les coûts restants pour finir la mission soient supérieurs aux gains obtenus par les objectifs traités. Dans ce cas, la borne est initialisée à l'infini.
(**Ligne 2**) Initialisation de la liste des nœuds pendants $listeP$ avec le nœud origine n_1.
(**Ligne 3**) La recherche est effectuée sur un nombre de nœuds limité à cpt_{max} de façon à limiter la durée de recherche de la première solution admissible. Le nombre de nœuds développés cpt_{max} dépend de l'application. Il est donné *a priori* par un expert du domaine en effectuant un compromis entre la qualité de la première solution et la durée de calcul pour la trouver.

Algorithme 4 : Algorithme de recherche d'une première solution admissible

```
début
1   Initialisation :
    si replanification alors
    |  BORNE = +∞
    sinon
    └  BORNE = 0
2   Mettre n_1 dans listeP
3   tant que (listeP n'est pas vide & cpt < cpt_max) faire
4   |  Traitement :
5   |  cpt ← cpt + 1
6   |  pour chaque n_i dans S(û) faire
7   |  |  listeT = ∅
8   |  |  pour chaque action a possible faire
9   |  |  |  Construire l'itinéraire de n_1 à n_i^a
10  |  |  |  Calculer I à vitesse donnée
11  |  |  |  g ← I
12  |  |  |  Ajouter n_i^a dans listeT
13  |  |  |  si (n_i ∈ W_e & g < BORNE) alors
14  |  |  |  |  Ĵ ← optimisation de J en respectant les contraintes
15  |  |  |  |  si Ĵ < BORNE alors
16  |  |  |  |  |  Itifinal ← itinéraire de n_1 à n_i^a
17  |  |  |  |  |  BORNE ← Ĵ
18  |  |  |  |  sinon
19  |  |  |  |  |  si (cpt = cpt_max & Itifinal vide) alors
20  |  |  |  |  |  |  Itifinal ← ∅
21  |  |  |  sinon
22  |  |  |  |  si (cpt = cpt_max & Itifinal vide) alors
23  |  |  |  |  |  Itifinal ← ∅
24  |  |  |  |  sinon
25  |  |  |  |  |  Calculer h de n_i^a à un nœud fin
26  |  |  |  Élaguer l'arbre d'exploration
27  |  |  Rangement des suivants dans listeP :
28  |  |  Mettre les éléments de listeT dans listeP
29  |  |  Effacer û de listeP
30  |  Imprimer Itifinal et retourner BORNE
fin
```

(**Ligne 10**) Il n'y a pas d'optimisation du critère lors du développement des nœuds. La vitesse doit être donnée *a priori*. Elle peut être fixée par le concepteur en tenant compte de son influence sur le temps et la consommation des ressources : le but du concepteur peut être par exemple de choisir la vitesse qui minimise $R_e(t_{\pi(i)}, r_{\pi(i)})$ pour des hypothèses raisonnables sur le trajet.
(**Ligne 13**) Si le nœud développé est un nœud fin et que la valeur du critère pour le plan est inférieure à la borne, alors on effectue l'optimisation des vitesses en prenant en compte toutes les contraintes du problème.
(**Ligne 15-17**) Si l'optimisation précédente donne un plan avec un critère de meilleure qualité que la valeur de la borne, alors cet itinéraire est mémorisé et la valeur de la borne est mise à jour.
(**Ligne 22-25**) Si l'optimisation précédente ne donne pas un plan avec un critère de meilleure qualité que la valeur de la borne, alors, si le nombre maximum de nœuds à développer est atteint, on indique qu'aucun itinéraire admissible n'a été trouvé, sinon, l'évaluation du critère jusqu'à un nœud fin pour le nœud considéré est calculée.
(**Ligne 27-28**) Le rangement des nœuds dans *listeP* est effectué selon la méthode de rangement choisie pour toute la planification, c'est-à-dire la même méthode que celle de l'algorithme 3.

☞ Lorsque le nombre maximum de nœuds à développer est atteint et qu'aucun itinéraire admissible n'a été trouvé, une autre solution envisageable pourrait être d'indiquer le meilleur itinéraire partiel trouvé. Ceci permettrait d'appliquer un plan en cas de contraintes dures sur la durée des calculs.

2.3.3 Traitement des contraintes et définition du critère

Les algorithmes 3 et 4 prennent en compte les contraintes temporelles sur chaque nœud et les contraintes sur les ressources. On définit ci-dessous de quelle manière.

Les contraintes (Equations 1.5) sur les ressources s'exprimant de manière non linéaire en fonction de Δ sont ajoutées au critère sous la forme de termes de pénalisation. Pour chacune de ces contraintes, on introduit une fonction de pénalisation.

Soit $r_e^s - r_{min}^s \geq 0$ la contrainte non linéaire sur la $s^{\text{ième}}$ composante du vecteur ressources. On a montré que r^s est une fonction de la séquence Q des états et des $\Delta_{i,j}$ entre les états sélectionnés. Soit Δ le vecteur des durées entre les états sélectionnés. La contrainte est en fait $r_e^s(\Delta) - r_{min}^s \geq 0$. Pour chaque composante du vecteur ressources, on introduit donc dans le critère une fonction $\Psi^s : \mathbb{R}^{q-1} \rightarrow \mathbb{R}$ définie par l'équation 2.1. Cette fonction est ajoutée au critère.

$$\Psi^s(\Delta) = [1 - H(r_e^s(\Delta) - r_{min}^s - \epsilon)] \times (r_e^s(\Delta) - r_{min}^s - \epsilon)^\alpha \tag{2.1}$$

où H est la fonction Heavyside, α un entier positif pair et ϵ un facteur de précision. La fonction est continue, sa dérivée est prolongeable par continuité. Les valeurs limites vont pénaliser le critère. La figure 2.1 montre la forme de cette fonction de pénalisation.

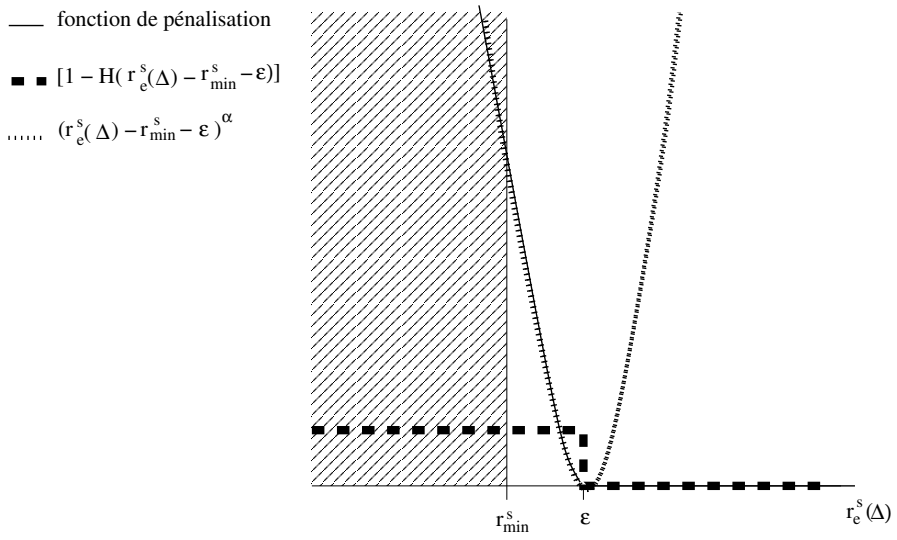


FIG. 2.1 – Exemple de fonction de pénalisation Ψ^s

Le critère à optimiser par l'algorithme 4 pour un nœud de W_e est donc :

$$J = R_e(t_e, r_e) - \sum_{o \in E_o} R_o + \sum_s \Psi^s(\Delta) \tag{2.2}$$

Dans l'algorithme 3, le critère partiel est ainsi :

$$I = R_e(t_{\pi(i)}, r_{\pi(i)}) - \sum_{o \in E_o^i} R_o(t_{s(o)}, r_{s(o)}, t_{e(o)}, r_{e(o)}, t_{r(o)}, r_{r(o)}) + \sum_s \Psi^s(\Delta) \tag{2.3}$$

Les contraintes linéaires sur le temps sont prises en compte dans les contraintes d'un simplexe (Voir Annexe B, Equation B.1) sous la forme de l'inégalité

$$\mathbf{A}.\Delta \leq \mathbf{b}$$

Soit $i-1$ le nombre d'arcs de l'itinéraire partiel de n_1 à n_i. $i-1$ est la taille de Δ.

Les matrices $\mathbf{A}(4(i-1), i-1)$ et $\mathbf{b}(4(i-1), 1)$ ont la forme suivante :

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots \\ -1 & 0 & 0 & 0 & \dots \\ 1 & 1 & 0 & 0 & \dots \\ -1 & -1 & 0 & 0 & \dots \\ 1 & 1 & 1 & 0 & \dots \\ -1 & -1 & -1 & 0 & \dots \\ \vdots & \vdots & \vdots & & \\ 1 & 0 & 0 & 0 & \dots \\ -1 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & \dots \\ 0 & -1 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & \dots \\ 0 & 0 & -1 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} t_{2_{max}} - t_1 \\ t_1 - t_{2_{min}} \\ t_{3_{max}} - t_1 \\ t_1 - t_{3_{min}} \\ t_{4_{max}} - t_1 \\ t_1 - t_{4_{min}} \\ \vdots \\ \frac{L(n_1,n_2,a_2)}{v_{min}(n_1,n_2,a_2)} \\ \frac{-L(n_1,n_2,a_2)}{v_{max}(n_1,n_2,a_2)} \\ \frac{L(n_2,n_3,a_3)}{v_{min}(n_2,n_3,a_3)} \\ \frac{-L(n_2,n_3,a_3)}{v_{max}(n_2,n_3,a_3)} \\ \frac{L(n_3,n_4,a_4)}{v_{min}(n_3,n_4,a_4)} \\ \frac{-L(n_3,n_4,a_4)}{v_{max}(n_3,n_4,a_4)} \\ \vdots \end{pmatrix}$$

Les $2n$ premières lignes correspondent aux contraintes de temps sur chaque point, données par l'équation 1.8. Les $2n$ dernières lignes correspondent aux contraintes de vitesses maximum et minimum données par l'équation 1.37.

Pour les ressources dont le niveau est une fonction linéaire sur les composantes de Δ (Equations 1.11 et 1.13) , l'ajout d'une ligne dans la matrice $\mathbf{A}$ et d'un élément au vecteur $\mathbf{b}$ permet d'éviter l'utilisation d'une fonction de pénalisation.

2.3.4 Optimisation du critère

A chaque nœud, un sous-problème d'optimisation est résolu sur le critère partiel. A ce stade du développement de l'arbre, les nœuds décrivant le plan sont choisis, ainsi que les actions pour les atteindre. La valeur du critère partiel ne dépend donc que du choix des durées $\Delta^{a_2}_{n_1,n_2}, \dots, \Delta^{a_i}_{n_{i-1},n_i}$. Ces durées sont les composantes du vecteur Δ. L'algorithme 5 présente l'optimisation sous contraintes des vitesses.

Une méthode basée sur l'algorithme de Frank & Wolfe [Frank & Wolfe 1956] est utilisée. Les contraintes non linéaires étant ajoutées au critère sous la forme de termes de pénalisation, le problème devient une optimisation d'un critère non linéaire sous des contraintes linéaires (Equation 2.4).

$$\min_{\Delta} I(\Delta) \text{ sous } \mathbf{A}.\Delta \leq \mathbf{b} \tag{2.4}$$

La résolution consiste à se ramener à un problème linéaire de façon à appliquer des méthodes de programmation linéaire malgré le critère non linéaire.
On construit une solution initiale (**Ligne 2-3**) au problème 2.4, noté $\Delta^{(0)}$, à partir de la connaissance des matrices $\mathbf{A}$ et $\mathbf{b}$. Cette construction s'effectue par la méthode des

Algorithme 5 : Optimisation du critère au nœud n_i

Entrées : séquence de nœuds de n_1 à n_i, séquence d'action de a_2 à a_i
Sorties : $I(\hat{\Delta})$: le critère partiel optimisé
début

1 i = 0
2 Construire **A** et **b**
3 $\Delta^{(0)} \leftarrow$ SolutionInitiale(**A, b**)
4 $I(\Delta) \leftarrow \mathrm{I}(\Delta^{(0)})$
5 **répéter**
6 $I(\hat{\Delta}) \leftarrow I(\Delta)$
7 $\mathbf{C}^\top \leftarrow I_\Delta^\top(\Delta^{(i)})$
8 $\hat{\Delta} \leftarrow$ SolutionSimplex($\mathbf{A}, \mathbf{b}, \mathbf{C}$)
10 k $\leftarrow$ Discret($\Delta^{(i)}, \hat{\Delta}$)
11 $\Delta^{(i+1)} \leftarrow \Delta^{(i)} + k(\hat{\Delta} - \Delta^{(i)})$
12 $i \leftarrow i + 1$
13 $I(\Delta) \leftarrow \mathrm{I}(\Delta^{(i)})$
14 **jusqu'à** $(|I(\hat{\Delta}) - I(\Delta)| < \epsilon_{max})$ *ou* $(i > I_{max})$
15 **retourner** $I(\hat{\Delta})$ *et* $\hat{\Delta}$

fin

deux phases, utilisée dans la méthode du simplexe (Annexe B, Section B.4.2) dans le cas où la forme simpliciale n'est pas évidente.
La solution admissible $\Delta^{(0)}$ permet d'initialiser l'itération (**Ligne 5-14**).

Les hypothèses suivantes sont ensuite posées.

HYPOTHÈSE 6 *$R_e(t_e, r_e)$ est continue et dérivable par rapport à t_e et par rapport à r_e.*

HYPOTHÈSE 7 *$R_o(t_{s(o)}, r_{s(o)}, t_{e(o)}, r_{e(o)}, t_{r(o)}, r_{r(o)})$ est continue est dérivable par rapport à $t_{s(o)}, r_{s(o)}, t_{e(o)}, r_{e(o)}, t_{r(o)},$ et $r_{r(o)}$.*

HYPOTHÈSE 8 *$f^s_{\pi(i)}(n_1, \ldots, n_i, a_2, \ldots, a_i, \Delta^{a_2}_{n_1,n_2}, \ldots, \Delta^{a_i}_{n_{i-1},n_i})$ est continue et dérivable par rapport aux composantes de Δ.*

En utilisant les séries de Taylor, le terme $I(\Delta)$ est alors développé au premier ordre autour de $\Delta^{(i)}$:

$$I(\Delta) = I(\Delta^{(i)}) + I_\Delta^\top(\Delta^{(i)}).(\Delta - \Delta^{(i)})$$

où I_Δ est le gradient de I.
Le problème revient alors à :

$$\min_\Delta \left(I(\Delta^{(i)}) + I_\Delta^\top(\Delta^{(i)}).(\Delta - \Delta^{(i)}) \right) \text{ sous } \mathbf{A}.\Delta \leq \mathbf{b}$$

En posant le vecteur $\mathbf{C}^\top = I_\Delta^\top(\Delta^{(i)})$, on retrouve la forme classique :

$$\min_{\Delta} \mathbf{C}^\top.\Delta \text{ sous } \mathbf{A}.\Delta \leq \mathbf{b} \tag{2.5}$$

La résolution se fait en deux étapes : on cherche d'abord la solution optimale $\hat{\Delta}$ du problème linéaire 2.5 par un simplexe [Dantzig 1963] (**Ligne 8**).
La deuxième étape consiste à écrire l'approximation de Taylor de la solution. Les contraintes sont linéaires, l'espace de recherche est donc convexe : pour tout point X_1 et X_2 de l'espace des solutions, toute combinaison linéaire $kX_1 + (1-k)X_2$, où $k \in [0,1]$ est aussi un point de l'espace des solutions. En particulier pour $\hat{\Delta}$ et $\Delta^{(i)}$, toute combinaison

$$\Delta^{(i+1)} = \Delta^{(i)} + k(\hat{\Delta} - \Delta^{(i)}) \text{ avec } k \in [0,1]$$

est admissible. Le coefficient k est optimisé par une méthode discrète d'optimisation monovariable de façon à minimiser le critère (**Ligne 10**). Le vecteur $\Delta^{(i+1)}$ est réutilisé pour l'itération suivante. Deux tests d'arrêt sont utilisés (**Ligne 14**) : le nombre d'itérations maximum possible I_{max} et le fait que la différence entre $I(\Delta^{(i+1)})$ et $I(\Delta^{(i)})$ soit suffisamment petite. Le réglage de ces paramètres doit être effectué en fonction des ordres de grandeur du critère.

2.4 Les méthodes de l'algorithme

Différentes méthodes sont envisagées pour la stratégie d'exploration de l'arbre de recherche, ainsi que pour l'élagage. En effet, un bon guidage pour l'exploration et un élagage efficace permettent d'obtenir de bonnes performances pour l'algorithme de replanification en ligne.

2.4.1 Les méthodes d'évaluation du coût du nœud courant à un nœud fin

Plusieurs méthodes d'évaluation du critère h (Algorithmes 3 et 4) du nœud courant à un nœud de fin sont étudiées. Pour ces méthodes, il est nécessaire de considérer le critère au nœud courant n_i de la séquence Q.

La valeur minimale de I, notée $\hat{I}$, est un minorant de I pour le plan qui peut être trouvé sur cette branche. En effet, la prise en compte de la suite du plan a pour effet de rajouter des contraintes sur le temps et les ressources et de considérer l'optimisation d'un critère différent.

Méthodes H_1 et H_i

Les méthodes H_1 et H_i ont pour but de trouver une borne inférieure de $J - \hat{I}$. Or il est possible de montrer l'équation 2.6.

$$J - \hat{I} \geq \sigma_{\pi(i),e} - \sum_{o \in P \setminus E_o^i} R_o(t_{s(o)}, r_{s(o)}, t_{e(o)}, r_{e(o)}, t_{r(o)}, r_{r(o)}) \tag{2.6}$$

L'ensemble $P \setminus E_o^i$ ("P privé de E_o^i") correspond à l'ensemble des objectifs non encore réalisés.

Preuve : Le critère J de n_1 à n_e est donné par l'équation 2.2 :

$$J = R_e(t_e, r_e) - \sum_{o \in E_o} R_o + \sum_k \Psi^k(\Delta)$$

Le critère partiel de nœud n_1 au nœud courant n_i est donné par l'équation 2.3 :

$$I = R_e(t_{\pi(i)}, r_{\pi(i)}) - \sum_{o \in E_o^i} R_o(t_{s(o)}, r_{s(o)}, t_{e(o)}, r_{e(o)}, t_{r(o)}, r_{r(o)}) + \sum_k \Psi^k(\Delta)$$

Donc :

$$J - I = R_e(t_e, r_e) - \sum_{o \in E_o} R_o - R_e(t_{\pi(i)}, r_{\pi(i)}) + \sum_{o \in E_o^i} R_o$$

Or $R_e(t_e, r_e)$ se décompose en un coût entre le nœud de planification initial et le nœud courant $R_e(t_{\pi(i)}, r_{\pi(i)})$, et en un coût résiduel $\sigma_{\pi(i),e}$ (Equation 1.14) :

$$R_e(t_e, r_e) = R_e(t_{\pi(i)}, r_{\pi(i)}) + \sigma_{\pi(i),e} \text{ avec } \sigma_{\pi(i),e} \in \mathbb{R}^+$$

Donc :

$$J - I = \sigma_{\pi(i),e} - (\sum_{o \in E_o} R_o - \sum_{o \in E_o^i} R_o)$$

Or l'ensemble $P \setminus E_o^i$ inclut l'ensemble des objectifs réalisés par le plan privé des objectifs déjà réalisés :

$$(E_o \setminus E_o^i) \subset (P \setminus E_o^i)$$

D'où :

$$J - \hat{I} \geq J - I = \sigma_{\pi(i),e} - \sum_{o \in P \setminus E_o^i} R_o(t_{s(o)}, r_{s(o)}, t_{e(o)}, r_{e(o)}, t_{r(o)}, r_{r(o)})$$

□

H_1 et H_i prennent en compte les objectifs de P non encore réalisés ($P \setminus E_o^i$), sachant que cela surévalue le nombre d'objectifs traités.

$\sigma_{\pi(i),e}$ étant positif, la méthode H_1 minimise $J - \hat{I}$ en utilisant une borne maximum des récompenses R_o sans tenir compte de l'utilisation des ressources (Equation 2.7).

$$\left\| \quad H_1 : h = - \sum_{o \in P \setminus E_o^i} \max_{t_{s(o)}, t_{e(o)}, t_{r(o)}} R_o(t_{s(o)}, r_1, t_{e(o)}, r_1, t_{r(o)}, r_1) \right. \tag{2.7}$$

r_1 représente le vecteur ressources au nœud d'application du plan : H_1 ne tient pas compte de l'utilisation des ressources jusqu'au nœud courant. H_1 est donc une borne

inférieure de la valeur du critère du nœud courant n_i à un nœud fin.

H_i minimise $J - \hat{I}$ en utilisant une borne maximum des récompenses R_o en tenant compte de l'utilisation des ressources jusqu'au nœud courant (Equation 2.8)

$$H_i : h = - \sum_{o \in P \setminus E_o^i} \max_{t_{s(o)}, t_{e(o)}, t_{r(o)}} R_o(t_{s(o)}, r_{\pi(i)}, t_{e(o)}, r_{\pi(i)}, t_{r(o)}, r_{\pi(i)}) \tag{2.8}$$

$r_{\pi(i)}$ représente le vecteur ressources au nœud courant : H_i tient compte de l'utilisation des ressources jusqu'au nœud courant. H_i est donc une borne inférieure de la valeur du critère du nœud courant n_i à un nœud fin.

☞ Une variante de la méthode H_i pourrait consister à chercher le maximum pour $t_{s(o)}$, $t_{e(o)}$ et $t_{r(o)}$ supérieurs à $t_{\pi(i)}$.

Pour ces deux méthodes, lors de la maximisation, on doit considérer que les variables $t_{s(o)}, t_{e(o)}$ et $t_{r(o)}$ sont contraintes par le minimum des $t_{k_{min}}$ et le maximum de $t_{k_{max}}$ des nœuds des ensembles $W_{s(o)}$, $W_{e(o)}$ et $W_{r(o)}$ respectivement.
Les méthodes H_1 et H_i n'évaluent pas les coûts de l'itinéraire dans le futur mais seulement les meilleurs gains espérés. Par ailleurs, l'utilisation des ressources n'est pas non plus évaluée : on ne prend donc pas en compte le fait que certaines branches de l'arbre d'exploration conduisent de toute évidence à des itinéraires non admissibles. De même, les objectifs dont les fenêtres temporelles ne peuvent pas être respectées sont pris en compte lors de l'évaluation de ces gains futurs.

Méthode H_r

La méthode H_r est basée sur la résolution d'un problème relaxé : le problème est résolu sans optimiser le critère I à chaque développement de nœud, les instants sont choisis comme s'il n'y avait aucune contrainte. La différence entre la valeur calculée pour le critère de cet itinéraire entier et la valeur courante donne une valeur de h. De plus, le nombre de nœuds développés est fixe. Par conséquent, la valeur de h est améliorée si d'autres nœuds fin sont développés avec un meilleur critère global. L'algorithme 6 explicite cette recherche.

La recherche devrait *a priori* être mieux guidée que pour H_1 ou H_i. Cependant, l'arbre peut ne pas être assez élagué et la recherche de tous les chemins possibles peut prendre du temps. H_r n'est ni une borne minimum, ni une borne maximum du critère. L'inconvénient majeur de H_r est que la séquence utilisée pour l'évaluation n'est pas optimisée : les coûts, les contraintes temporelles et l'utilisation des ressources sont mal pris en compte.

Algorithme 6 : Méthode H_r

Entrée : n_i
Sortie : Évaluation du critère de n_i à un nœud fin
début

```
1  Initialisation :
2  si n_i ∈ W_e alors Retourner 0
3  BORNE = +∞ ; I_0 = Calcul du critère I de n_1 à n_i à vitesse optimale
4  Mettre n_i dans listeP
5  tant que (listeP n'est pas vide & cpt < cpt_max) faire
6      Traitement :
7      cpt ← cpt + 1
8      pour chaque v dans S(û) faire
9          listeT = ∅
10         pour chaque action a possible faire
11             Construire l'itinéraire de n_1 à v^a
12             Calculer I à vitesse optimale
13             h ← I − I_0
14             Ajouter v^a dans listeT
15             si (v ∈ W_e & h < BORNE) alors
16                 BORNE ← h
17             Élaguer l'arbre d'exploration
18         Rangement des suivants dans listeP :
19         Mettre les éléments de listeT dans listeP
20         Effacer û de listeP
21 Retourner BORNE
```

fin

Méthode H_W

La méthode H_W a pour objectif d'utiliser au mieux des renseignements obtenus par une résolution effectuée sur un haut niveau de description pour guider la recherche. Elle utilise la hiérarchie à deux niveaux du graphe où le haut niveau correspond au niveau des objectifs. Les supernœuds sont le nœud de début de planification, un nœud pour chaque objectif et un nœud pour la fin de la mission. Un superarc spécifie une action possible entre deux supernœuds.

Soit $G_H(N_H, A_H)$ le graphe de haut niveau défini par l'ensemble des supernœuds N_H et des superarcs A_H. Soit $\overline{P}$ le cardinal de P, l'ensemble des objectifs à réaliser. N_H est composé de l'ensemble W_1, de l'ensemble W_e et de $\overline{P}$ éléments représentant pour chaque objectif $o \in P$, les triplets $(W_{s(o)}, W_{e(o)}, W_{r(o)})$. La figure 2.2 illustre le graphe construit à partir des données de la figure 1.9.
Le graphe de haut niveau G_H aura $\overline{P}+2$ nœuds. G_H contient des cycles. La contrainte "*pour chaque objectif o, la machine à état fini peut être dans l'état $W_{s(o)}$ une seule fois*" s'étend au graphe haut niveau pour les éliminer.

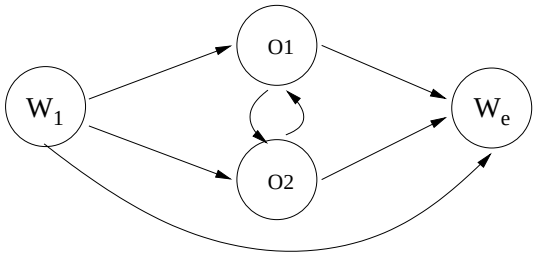


FIG. 2.2 – Graphe de haut niveau

Le but final de H_W est d'effectuer une recherche arrière sur le graphe de haut niveau puis d'affecter un vecteur heuristique à chacun des supernœuds, image de la valeur du critère du supernœud considéré jusqu'à un supernœud but. Ensuite, lors du développement, les nœuds seront évalués en fonction de la valeur heuristique du supernœud qui leur correspond.

La recherche arrière effectuée sur des supernœuds de N_H ne permet pas d'obtenir des renseignements de bonne qualité. En effet, chaque nœud a une unique valeur heuristique, ayant pour but de donner une information sur le coût du critère futur. Montrons-le sur l'exemple du graphe de haut niveau de la figure 2.2, développé sur la figure 2.3. Imaginons que cette évaluation porte simplement sur la distance restante jusqu'à un nœud fin : pour chaque supernœud, il faut donc déterminer un minorant de la distance restante jusqu'à un supernœud fin. Pour le supernœud W_e, cette distance est nulle. Pour le supernœud $O1$, elle sera égale à $min\Big(d(O1, W_e), d(O1, O2) + d(O2, W_e)\Big)$ où $d(A, B)$ représente la distance minimum entre les ensembles A et B. On procède de même pour le supernœud $O2$: $min\Big(d(O2, W_e), d(O2, O1) + d(O1, W_e)\Big)$ et pour W_1 :

$min\Big(d(W_1, W_e), d(W_1, O2)+d(O2, W_e), d(W_1, O1)+d(O1, O2)+d(O2, W_e), d(W_1, O1)+ d(O1, W_e), d(W_1, O2) + d(O2, O1) + d(O1, W_e)\Big)$.

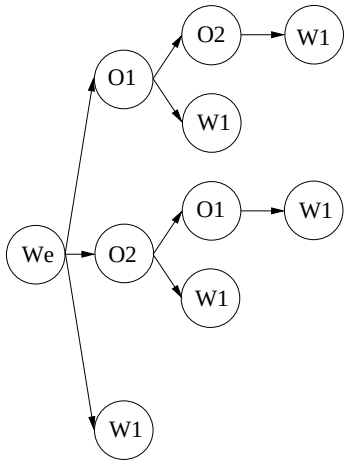


FIG. 2.3 – Développement en recherche arrière du graphe G_H

Ainsi, quel que soit le chemin parcouru avant le supernœud, il n'existe qu'une valeur pour l'évaluation h du critère du supernœud considéré au supernœud fin. C'est une information trop pauvre : le graphe G_H n'est pas assez développé pour trouver des valeurs heuristiques efficaces. Plusieurs développements partiels du graphe sont proposés afin de mieux situer la place du supernœud dans l'itinéraire. Le problème est double :

1. Il faut évaluer le coût du critère du supernœud correspondant au supernœud fin grâce à une recherche arrière.
2. Sachant le nœud de bas niveau en cours de traitement, il faut déterminer quelle est la place du supernœud correspondant dans le plan : à chaque graphe décrit, il faut trouver un moyen de coder l'emplacement du supernœud dans l'exploration pour retrouver l'information acquise en recherche arrière lors du développement en recherche avant.

Les divers développements consistent à explorer partiellement le graphe et à construire un nouveau graphe plus informatif sur lequel est effectuée la recherche arrière.

Énumération de tous les supernœuds La première méthode d'exploration envisagée est définie par l'énumération de la succession possible de tous les supernœuds, sans prendre en compte les chemins finissant avant la réalisation de tous les objectifs de la mission. Un supernœud indique donc l'objectif en train d'être réalisé et l'enchaînement de tous les objectifs qui ont été réalisés. La figure 2.4 illustre le graphe correspondant à une mission avec 4 objectifs. On code chaque nœud de l'arbre par un vecteur à $\overline{P}$ composantes correspondant respectivement au premier, deuxième, ..., et $\overline{P}^{ième}$ objectif atteint. Chaque composante peut prendre $\overline{P}+1$ valeurs parmi $\{0, 1, \ldots, \overline{P}\}$. Le zéro correspond à "pas d'objectif". Pour l'exemple, le codage [4 1 3 0] indique que le système a réalisé l'objectif 4 puis l'objectif 1 et qu'il est en train de réaliser l'objectif 3.

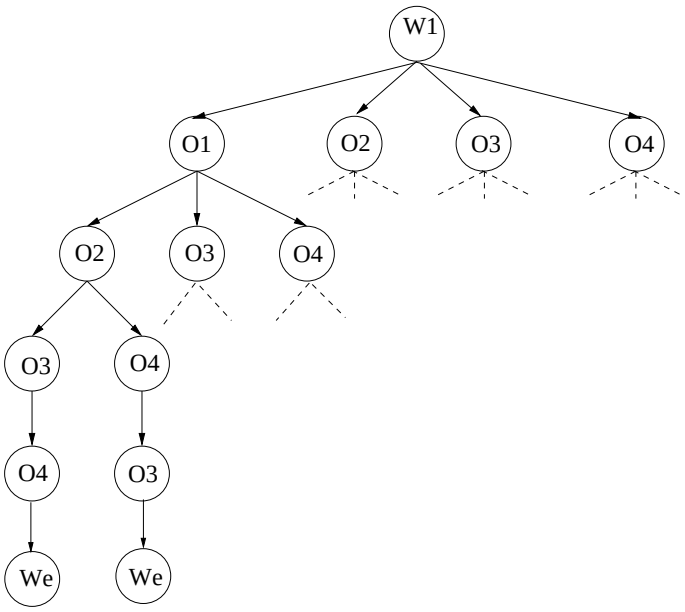


FIG. 2.4 – Graphe d'énumération

Cette méthode semble trop développée, car le nombre de supernœuds à développer est $\sum_{p=0}^{\overline{P}} \frac{\overline{P}!}{p!}$. Par exemple, pour une mission avec 4 objectifs, le nombre de supernœuds à développer est de 89. L'évaluation d'un itinéraire, sans optimiser les vitesses, ne nécessite pas une connaissance si précise de l'ordre dans lequel les objectifs sont réalisés.

Description type STRIPS - formalisme état-action Pour la deuxième exploration envisagée, on définit pour chaque objectif $o \in P$ trois états :
- Non Traité (NT) : la machine à état n'est pas passée par l'état $W_{e(o)}$;
- Traité (T) : la machine à état est passée par $W_{s(o)}$ et $W_{e(o)}$, mais pas par $W_{r(o)}$;
- Réalisé (R) : la machine à état est passée par $W_{e(o)}$ et $W_{r(o)}$.

On note l'état pour un objectif o : state(o, E), avec $o \in P$ et $E \in \{NT, T, R\}$.
On peut définir deux actions pour chaque état.

L'action *transmettre* s'applique sans argument sur l'ensemble P des objectifs de la mission :
transmettre ()
Précondition : $\exists o \in P$ tel que $state(o, T)$
Faits détruits : Pour tous les o tels que $state(o, T)$, détruire l'état $state(o, T)$
Faits ajoutés : Pour tous les o tels que $state(o, T)$, ajouter l'état $state(o, R)$

L'action *aller à* s'applique sur l'objectif o donné en argument :
aller à (o)

Précondition : $\{state(o, NT)\}$
Fait détruit : détruire $\{state(o, NT)\}$
Fait ajouté : ajouter $\{state(o, T)\}$

On code l'état des objectifs. On considère donc un vecteur à $\overline{P}$ composantes, chaque composante représentant un objectif. Chaque composante a une valeur parmi $\{0, 1, 2\}$ qui correspond à l'état de l'objectif (NT, T, R). Il y a donc $3^{\overline{P}}$ états. Pour l'exemple de la mission avec quatre objectifs, le codage [2 0 1 2] indique que le système a réalisé l'objectif 4 et l'objectif 1 et qu'il est en train de réaliser l'objectif 3.
La représentation de type STRIPS a l'avantage d'être très concise, mais il est difficile d'effectuer une recherche arrière dessus. On propose donc une dernière représentation qui possède les mêmes informations et sur laquelle on effectue la recherche arrière.

Représentation adoptée Pour répondre à la partie du problème concernant la détermination de la place d'un supernœud dans le plan, la dernière méthode d'exploration du graphe haut niveau prend en compte les besoins de représentation. En effet lors de l'exploration, l'objectif o traité et les objectifs déjà réalisés doivent être connus. Le graphe partiellement développé, dont un exemple est donné figure 2.5, est constitué de nœuds étiquetés par l'objectif traité et les objectifs déjà réalisés. Par exemple, un nœud étiqueté "2/1,4" indique que le système traite l'objectif 2, et a réalisé les objectifs 1 et 4. Plus formellement, soit o_i l'objectif courant et E_o^i la liste des objectifs ayant été traités, l'étiquette du nœud n_H de haut niveau sera :

$$label(n_H) = [o_i/E_o^i] \text{ lu } o_i \text{ sachant } E_o^i$$

L'étiquette [0] montre le début de mission, celle à [99] indique les nœuds fin. Le nombre de nœuds sera $2 + \overline{P}.2^{\overline{P}-1}$, soit 34 pour 4 objectifs. Il constitue un compromis entre les deux premières méthodes d'exploration.

Pour coder un nœud, le choix d'un vecteur à $\overline{P} + 1$ composantes semble pertinent même s'il utilise des valeurs inutiles. La première composante du vecteur prend ses valeurs parmi $\{0, 1, \ldots, \overline{P}, 99\}$ et indique dans l'objectif en train d'être réalisé, le zéro indiquant qu'aucun objectif n'a été réalisé et le 99 que le supernœud destination W_e a été atteint. Les $\overline{P}$ autres composantes prennent leur valeur dans $\{0, 1\}$ et indique si l'objectif a été réalisé. Par exemple, la valeur [3 1 0 1 1] indique que l'objectif 3 est en train d'être réalisé et que 1 et 4 ont été réalisés.
Un autre codage possible est celui qui découle de la solution de type STRIPS. Le codage est un vecteur de $\overline{P}$ composantes, prenant leur valeur parmi $\{0, 1, 2\}$. Le zéro indique que l'objectif n'a pas été réalisé, le 1 que l'objectif est en train d'être réalisé, le 2 que l'objectif a été réalisé. Pour le même exemple, la valeur sera : [2 0 1 2].
Le codage adopté de manière arbitraire est le premier codage proposé.

On construit donc un nouveau graphe développé où chaque supernœud est associé à un matricule (correspondant au codage défini plus haut) permettant de l'identifier. Le matricule permet de passer du graphe de bas niveau sur lequel s'effectue la recherche avant au graphe des supernœud qu'on vient de définir. Les superarcs de ce

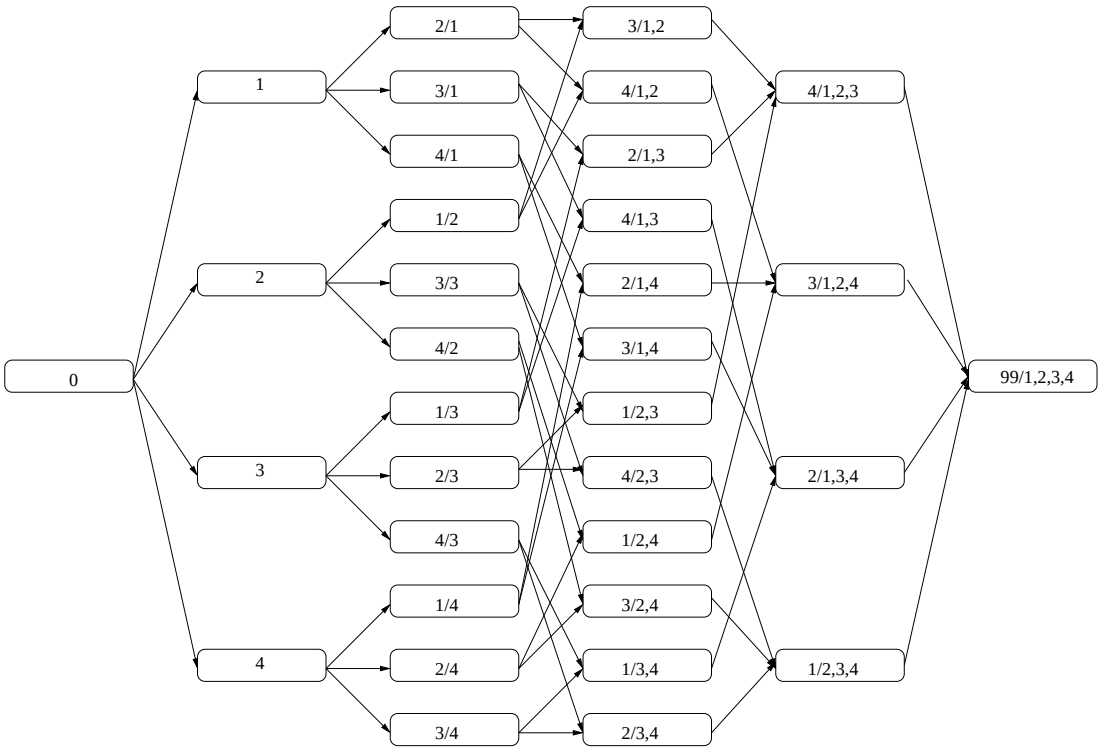


FIG. 2.5 – Solution compromis pour la description de l'exploration

graphe sont munis d'un vecteur A. La première composante de A représente un minorant de la durée pour parcourir le superarc. Les composantes suivantes ont pour but de représenter des minorants de la consommation des ressources sur le superarc. Si les ressources sont décomposables (Equation 1.12), il suffit de déterminer un minorant de $\tilde{f}^s_{\pi(i)}(n_{i-1}, n_i, a_i, \Delta^{a_i}_{n_{i-1},n_i})$ sur toutes les paires de nœuds reliées par le superarc. Dans le cas contraire, la composante de A doit permettre d'obtenir un minorant de la consommation de la ressource sur le superarc. L'idée est d'effectuer une programmation dynamique inverse [Bellman 1952] sur ce graphe pour affecter à chaque supernœud un vecteur V correspondant à une évaluation de la quantité minimale de ressources consommées, à une évaluation d'un minorant de la durée de la mission et à un gain maximal obtenu entre le supernœud considéré et le supernœud W_e.

Pour effectuer cette programmation dynamique, il est nécessaire de définir les opérateurs *égalité (=)*, *somme (+)*, et *inférieur à (<)* pour deux vecteurs de dimension n, $X = x_1x_2 \dots x_n$ et $Y = y_1y_2 \dots y_n$ représentant des vecteurs de type A et V. Les opérateurs *égalité* et *somme* sont définis terme à terme.

DÉFINITION 6 *$X = Y$ ssi $\forall i = 1, \dots n$, $x_i = y_i$*

DÉFINITION 7 *$Z = X + Y$ est un vecteur de dimension n. $Z = z_1z_2 \dots z_n$ / $\forall i = 1, \dots n, z_i = x_i + y_i$*

L'opérateur *inférieur à* permet d'effectuer la programmation dynamique inverse dans le graphe. Il s'agit alors d'une optimisation multicritère dans un graphe où les critères sont les composantes du vecteur V. L'objectif n'est pas de ranger les supernœuds, mais de leur attribuer un vecteur de valeur que l'on peut évaluer par la suite. Ainsi, il ne semble pas judicieux de transformer l'ensemble des critères en une fonction pondérée unique. L'idée est d'utiliser un objectif hiérarchisé [Focacci & Godard 2002].

DÉFINITION 8 *Soit n critères considérés hiérarchiquement. Pour deux solutions s_1 et s_2, s_1* ***est meilleur que*** *s_2 ssi il existe k tel que $(z_k(s_1) < z_k(s_2)$ et $(z_h(s_1) \leq z_h(s_2) \forall h = 1, \dots k-1))$*

Cette méthode nécessite un classement prédéfini parmi les critères $z_1, \dots z_c$. La méthode choisie privilégie les solutions avec les meilleures valeurs de critères de haut rang si $z_1, \dots z_c$ représente la liste des critères rangés du moins important au plus important. La méthode est appliquée pour définir l'opérateur *inférieur à*. Les composantes des vecteurs de type A sont rangées du critère le plus important au critère de moindre importance.

DÉFINITION 9 *$X < Y$ ssi il existe k tel que $(x_k < y_k$ et $(x_h \leq y_h \forall h = 1, \dots k-1))$*

L'algorithme de programmation dynamique inverse (Algorithme 2.4.1) ressemble à un algorithme de Ford où les potentiels f_i sont remplacés par des vecteurs V.

Algorithme 7 : Algorithme de Programmation Dynamique Inverse

```
début
    Initialisation :
    Initialiser tous les vecteurs V à leur valeur maximale
    Placer le supernœud W_e dans listeP
    tant que listeP n'est pas vide faire
        Traitement :
        pour chaque v dans P(û) faire
            Amélioration des potentiels :
            si A^{v,u} + V^u < V^v alors
                remplacer V^v par A^{v,u} + V^u
            si v n'est pas dans listeP alors
                Placer v dans listeP
        Effacer û de listeP
fin
```

NOTATIONS : $A^{v,u}$ est le vecteur A pour le superarc (v, u) (équivalent longueur dans l'algorithme de Ford) ; V^v est le vecteur V pour le supernœud v (équivalent potentiel dans l'algorithme de Ford) ; $P(\hat{u})$ est l'ensemble des prédécesseurs du supernœud $\hat{u}$.

Une fonction de pré-traitement va fournir une liste de supernœuds munis de leur vecteur V. Cette liste est utilisée comme argument dans l'algorithme principal de planification : on planifie avec une table heuristique qui fournit une valeur heuristique du coût du nœud courant jusqu'à un nœud fin à partir de la donnée des V.

Soit n_i le nœud courant et v le supernœud auquel il est associé, dans le cas de ressources décomposables, l'évaluation h de n_i à un nœud fin est donnée par :

$$\left\| \quad H_W : h = \Gamma(V^v) \right. \tag{2.9}$$

où Γ est une fonction simple donnant une évaluation du critère de n_i à un nœud fin en connaissant le vecteur V^v.

En résumé, le matricule permet de faire le lien entre le nœud de bas niveau en cours de traitement et le supernœud correspondant. La fonction Γ, appliquée au vecteur V, permet l'évaluation du coût du critère du supernœud au supernœud fin, égale à l'évaluation h souhaitée du critère du nœud traiter à un nœud fin.

Dans le cas de l'existence de ressources non décomposables, une solution consiste à hiérarchiser les ressources selon leur importance et à trouver une fonction pour comparer deux nœuds de bas niveau en connaissant le critère partiel optimisé $\hat{I}$ pour y parvenir et le vecteur V du supernœud qui leur est associé. Un exemple est donné dans la section 2.5.

2.4.2 Les méthodes d'élagage

Deux méthodes d'élagage sont proposées. Elles tiennent compte du fait que le critère n'est pas monotone en descendant une branche de l'arbre d'exploration. Les méthodes d'élagage classiques, qui consistent à élaguer tous les nœuds dont le potentiel est supérieur à la valeur BORNE, ne s'appliquent pas. Les méthodes développées doivent tenir compte de l'évolution future du critère.

La première méthode, notée E_1, est utilisée si h est une borne minimum du critère du nœud courant à un nœud fin :

RÈGLE 1 **si** $g + h > BORNE$ **alors** *élaguer le nœud v*

La seconde, notée E_2, est utilisée dans les autres cas. Elle introduit un facteur $\gamma \geq 0$ indiquant le degré de confiance dans la fonction d'évaluation du nœud courant vers un nœud but. Plus la valeur est petite, plus la confiance est grande.

RÈGLE 2 **si** $((g + h) -\gamma |g + h|) > BORNE$ **alors** *élaguer le nœud v*

2.4.3 Les méthodes de rangement de la liste des nœuds pendants

Le choix du rangement des éléments de $listeT$ dans $listeP$ est essentiel. Si le rangement n'est pas efficace, la durée de la recherche peut être très grande. Quatre rangements sont considérés.

R_1 et R_2 correspondent à des recherches en profondeur ordonnée. Ces rangements consistent tout d'abord à ordonner les successeurs du nœud développé $\hat{u}$ par valeur croissante (soit en g croissant, soit en $g + h$ croissant), puis de les ranger en tête de $listeP$ selon cet ordre : le nœud qui sera développé à l'itération suivante sera le successeur de $\hat{u}$ à g minimal pour R_1 et à $g + h$ minimal pour R_2. On parlera de **recherches (ou rangements) en profondeur ordonnée**.

R_3 et R_4 correspondent à des recherches au meilleur g (respectivement $g+h$) d'abord. Ces rangements consistent à mettre les successeurs dans $listeP$ et à réordonner $listeP$ en g (ou $g + h$) croissants. Le nœud qui sera développé à l'itération suivante sera le nœud pendant de g ou $g + h$ minimal. On parlera de **recherches (ou rangements) au meilleur d'abord**.

☞ Notons que pour les rangements R_1 et R_3, la fonction d'évaluation ne sert que pour l'élagage. Elle ne sert pas à guider la recherche.

2.5 Quelques interprétations sur l'exemple applicatif

Rapprochons maintenant nos propositions algorithmiques de leur application sur l'exemple de planification de mission d'observation pour un drone.

La recherche d'une première solution admissible calcule un chemin sans optimiser les dates de passage. Cela signifie physiquement que le véhicule est supposé voler à vitesse constante. La vitesse est donnée *a priori*. Le concepteur peut par exemple choisir de minimiser la consommation du carburant en faisant l'hypothèse que le véhicule vole à altitude constante et à vitesse constante, et qu'il n'y a pas de vent : sa consommation est donc constante. Il existe alors une vitesse, dite vitesse "économique", telle que la durée de vol est maximale. Ce choix est arbitraire, on peut aussi choisir la vitesse maximale possible, pour minimiser les temps d'expositions au danger par exemple.

L'optimisation du critère consiste à trouver les durées pour chaque arc, qui minimise la consommation des ressources en probabilité de survie et carburant et maximise les récompenses pour chaque réalisation d'objectif. Les contraintes sur les ressources sont traitées selon le type de ressource (décomposable ou pas). Ainsi, la contrainte $r_e^1 - r_{min}^1 \geq 0$, concernant la probabilité d'être vivant, est traitée par une fonction de pénalisation, car la ressource est non décomposable. La contrainte $r_e^2 - r_{min}^2 \geq 0$, concernant la consommation, est aussi traitée par une fonction de pénalisation. En effet, même si la ressource est décomposable, elle présente une fonction $f_{\pi(i)^2}$ non linéaire en Δ.

Les hypothèses liées à la méthodes d'optimisation du critère (hypothèses 6, 7 et 8) implique d'une part que la fonction $P_{Obs_o}(t_{s(o)})$ est continue et dérivable en $t_{s(o)}$, d'autre part que $T(h, \Delta)$ soit continue et dérivable en Δ. Comme les durées entre deux nœuds sont physiquement non nulles, cette hypothèse est vérifiée.

La méthode H_1 prend en compte une borne supérieure de la somme des récompenses sans tenir compte de l'utilisation des ressources. Une évaluation des gains non encore obtenus est effectuée. H_1 est ainsi l'opposé de l'évaluation des futurs profits maximum. L'équation 2.7 devient :

$$H_1 \ : h = -r_1^1 \sum_{o \in P \setminus E_o^i} G_o . \max_{t_{s(o)}} P_{Obs_o}(t_{s(o)}) \tag{2.10}$$

La méthode H_i prend en compte une borne supérieure de la somme des récompenses et tient compte de l'utilisation des ressources sous la forme d'une pondération par la probabilité d'être vivant. Ainsi, cette méthode consiste à évaluer les récompenses non encore obtenues et à pondérer cette évaluation avec la probabilité d'être vivant au nœud courant. L'équation 2.8 devient :

$$H_i \ : h = -r_{\pi(i)}^1 \sum_{o \in P \setminus E_o^i} G_o . \max_{t_{s(o)}} P_{Obs_o}(t_{s(o)}) \tag{2.11}$$

L'inconvénient majeur des méthodes H_1 et H_i est que le coût du danger futur n'est pas évalué.

La méthode H_r est basée sur la solution d'un problème relaxé : le problème est résolu avec une vitesse constante sur l'itinéraire, en développant un nombre fixe de nœuds. La vitesse choisie est identique à celle choisie pour la recherche d'une première solution admissible.

La méthode H_W utilise une description haut niveau : chaque objectif est considéré globalement et une valeur heuristique lui est affectée. Cette valeur est utilisée lors de l'exploration de l'arbre des chemins. La solution compromis, décrite dans le troisième paragraphe de la section 2.4.1, est choisie. Les arcs du nouveau graphe sont munis d'un vecteur A dont la première composante est la distance minimale dans le danger, la deuxième composante est la quantité de masse consommée et la troisième composante est la durée consommée pour l'arc. La dernière composante de A est l'opposée de la récompense maximum acquise pour l'arc. Ainsi, la probabilité d'être vivant au temps courant, ressource non décomposable, est remplacée par une image simplifiée de l'utilisation de cette ressource : la durée minimale dans le danger.
A partir de ce graphe, on effectue une programmation dynamique inverse selon l'algorithme 2.4.1 pour munir chaque supernœud d'un vecteur V, dont la première composante est une image simplifiée de la probabilité d'être vivant "consommée" minimale du nœud considéré à un nœud destination, la deuxième composante est la quantité de masse consommée et la troisième composante est un minorant de la durée de la mission du nœud considéré à un nœud destination. La dernière composante de V est l'opposée de la récompense maximum acquise du nœud considéré à un nœud destination. Pour effectuer cette programmation dynamique, tous les vecteurs V sont initialisés à $[\ 1 \quad m_{carbu} \quad \infty \quad 0\ ]$ sauf celui du supernœud de fin initialisé à $[\ 0 \quad 0 \quad 0 \quad 0\ ]$.
Le résultat de la programmation dynamique inverse est un graphe où chaque supernœud possède un vecteur V optimisé.
Tous les supernœuds sont stockés sous la forme d'une liste de supernœuds, munis de leur vecteur V. Cette liste est utilisée comme argument dans l'algorithme principal (Algorithme 3) : à l'étape "Calculer h de v^a à un nœud fin ; Stocker dans v^a : $\hat{I}$ et h" (**Ligne 13-14**), la procédure suivante est adoptée :

Remplacement des lignes 13-14 de l'algorithme 8 pour H_W

Entrées : n_i^a, liste des supernœuds munis de leur vecteur V
début
  Construire le matricule de n_i
  Trouver le supernœud w correspondant au matricule de n_i^a
  Stocker dans n_i^a : $\hat{I}$ et V^w
fin

Ensuite, l'opérateur "meilleur que" des rangements en $g+h$ (R_2 et R_4) est remplacé, en utilisant les composantes du vecteur V. On note ainsi $V[i]$ la $i^{\text{ième}}$ composante du vecteur V.

DÉFINITION 10 *Soit deux nœuds n_a et n_b de critère partiel optimisé $\hat{I}^a$ et $\hat{I}^b$ et de vecteur heuristique associé V^a et V^b, alors n_a* ***est meilleur que*** *n_b ssi :*

$$V^a[1] < V^b[1]$$

ou

$$\Big(V^a[1] = V^b[1] \ et$$
$$\hat{I}^a + V^a[2].P_{carbu} + V^a[3].P_{vol} - V^a[4] < \hat{I}^b + V^b[2].P_{carbu} + V^b[3].P_{vol} - V^b[4]\Big)$$

Ainsi la hiérarchisation des critères permet de ranger les nœuds lors de l'exploration, même en cas de ressource non décomposable.

2.6 Conclusion

Un algorithme de base permet de chercher des solutions au problème de planification de mission. Cet algorithme s'inspire de l'algorithme du A^* pour combiner recherche dans le graphe et optimisation continue. Plusieurs méthodes de guidage et d'élagage de la recherche conduisent à plusieurs variantes de l'algorithme. Il est difficile de savoir *a priori* quelle méthode choisir. Il est donc nécessaire d'évaluer ces méthodes sur des problèmes de planification en ligne. Par ailleurs, l'algorithme proposé doit être intégré dans le système de conduite du véhicule. Ces deux aspects font l'objet de la suite de ce mémoire.

3 Intégration de la planification dans une architecture embarquée

L'ennui, en matière de décision,
c'est de ne jamais savoir si on pourra vraiment s'y tenir.
Jacques Languirand – Tout compte fait

Résumé : Ce chapitre décrit une architecture embarquée intégrant la fonction de planification. L'architecture doit permettre au système d'effectuer sa mission et de prendre en compte des événements invalidant le plan courant en cours de mission. Une étude de la littérature sur les liens entre planification et exécution est effectuée. Elle permet de dégager les choix de la planification en ligne sur événement et d'une architecture hybride hiérarchique. Après une présentation des fonctionnalités et composants exigés pour une telle architecture, on s'attache à décrire les différents niveaux de la hiérarchie proposée et le fonctionnement de chaque composant de l'architecture. Enfin, on explique pourquoi et comment le logiciel ProCoSA peut être utilisé pour décrire et mettre en œuvre l'architecture. Une dernière partie présente les critères pour évaluer la criticité temporelle d'un calcul de replanification.

3.1 Introduction

L'objectif est de définir une architecture embarquée compatible avec une autonomie décisionnelle. Les algorithmes de planification d'objectifs et d'itinéraires proposés au chapitre précédent doivent être intégrés dans cette architecture pour permettre au système d'adapter son comportement à son état et à l'environnement dynamique. L'intégration doit prendre en compte le fait que l'activation des calculs de plan est déclenchée par des événements.

Définition 11 *On appelle* ***aléa*** *un événement pouvant survenir en cours de mission dont la date d'occurrence est imprévisible.*

Un système autonome a deux objectifs majeurs : **effectuer au mieux sa mission tout en restant vivant** et **réagir aux aléas de mission, de l'environnement ou du système**. L'architecture embarquée doit répondre à ces deux objectifs en organisant les tâches physiques de la mission ainsi que les tâches de raisonnement.

3.2 Littérature sur les liens entre planification et exécution

Les missions avec des communications limitées entre les opérateurs et le véhicule requièrent de celui-ci une autonomie de haut niveau. Après une définition de l'autonomie de haut niveau, on décrira ici les deux approches majeures pour obtenir une réaction en ligne à des événements pouvant perturber la réalisation du plan courant : la planification hors ligne et la planification en ligne. Cette réaction est conditionnée par la prise en compte de la planification pendant l'exécution dans une architecture de contrôle. On étudiera les architectures existantes en dégageant celles qui peuvent être utilisées pour intégrer un module de planification de mission.

3.2.1 Les niveaux d'autonomie

L'autonomie d'un véhicule se définit par rapport au rôle de l'opérateur.
Plusieurs classifications ont été proposées dans la littérature. Les niveaux d'autonomie ont tout d'abord été définis par Fargeon [Fargeon 1993] (Table 3.1). Le critère de classification utilisé est la localisation géographique de l'opérateur par rapport à l'engin.

Niveau	Description
engin semi-autonome	l'opérateur est à bord de l'engin mais reçoit une aide à la décision
engin télécommandé	l'opérateur est déporté mais en vue directe avec l'engin
engin téléopéré	l'opérateur est déporté et n'est pas en vue directe avec l'engin ; l'environnement lui est restitué (téléprésence)
engin autonome	engin disposant de capacités significatives de prises de décision, de mobilité et d'adaptation à la mission

Tab. 3.1 – Niveaux d'autonomie définis par Fargeon

L'Office de la Recherche Navale américain (ONR) propose une classification en 6 niveaux [ONR 2000](Table 3.2). Cette classification porte sur la position du centre de décision. Celui-ci va de l'opérateur jusqu'à l'engin, en passant par plusieurs niveaux de partage de la décision.

Ces deux premières définitions permettent de classifier les niveaux d'autonomie pour un seul engin.

Une nouvelle classification en dix niveaux a été donnée par l'office du secrétariat de la défense des USA [OSD 2002] sur laquelle peuvent se positionner les drones les plus connus et les travaux de cette thèse sur la planification de mission en ligne (Figure 3.1). Le critère de classification porte sur les fonctions embarquées. La planification de mission en ligne pour un engin n'est pas le plus haut niveau d'autonomie. Au-delà, c'est l'autonomie d'un groupe multi-véhicules qui est définie. La replanification embarquée est le plus haut niveau décisionnel d'un engin évoluant sans formation.

D'autre critères de classification peuvent être envisagés, par exemple le niveau cognitif de l'interaction entre l'opérateur au sol et le véhicule. L'interaction peut aller d'un

Niveau	Nom	Description
0	Engin téléopéré	Toute l'activité du système est le résultat d'entrées/commandes humaines. Le système ne prend aucune décision.
1	Engin humainement assisté	Le système peut agir en parallèle avec des entrées opérateur, il améliore la réussite de l'opérateur sur l'activité qu'il effectue mais n'a pas la capacité d'agir sans ordre de l'opérateur. Par exemple une voiture à boîte automatique.
2	Délégation du contrôle	Le système peut avoir un contrôle de son activité sur la base de la délégation. Par exemple un pilote automatique pouvant être activé ou non par un pilote.
3	Supervision humaine	Le système peut effectuer un large panel d'activités avec la permission ou le contrôle d'un humain. Le système renvoie suffisamment d'informations sur ses opérations internes et son comportement pour qu'un humain puisse le superviser et le rediriger si nécessaire. Le système n'a pas la capacité de prendre des initiatives sur des comportements qui ne seraient pas en rapport avec les tâches qui lui sont assignées au départ.
4	Initiative partagée	L'humain et le système peuvent tous les deux prendre des initiatives basées sur les données des capteurs. Le système peut coordonner son comportement avec celui de l'humain tant explicitement qu'implicitement. Des règles régissent le partage de l'autorité.
5	Autonomie totale	Le système n'a besoin d'aucune intervention humaine pour effectuer sa mission et s'adapte à son environnement.

Tab. 3.2 – Niveaux d'autonomie de l'ONR

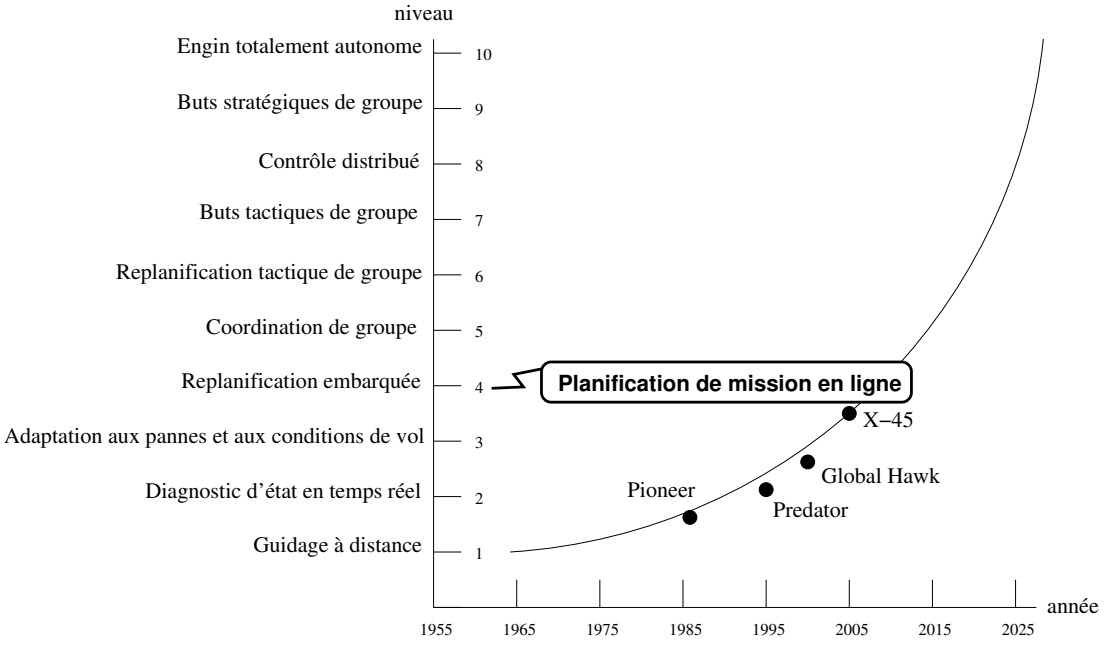


FIG. 3.1 – Catégories de drones - Source OSD UAV Roadmap 2002-2027

écran vidéo et un joystick jusqu'au langage parlé [Doherty *et al.* 2000].

3.2.2 Calcul de plans hors ligne ou en ligne

Le calcul de plans hors ligne consiste à trouver, avant l'exécution de la mission, un plan qui va prévoir et traiter tous les événements possibles. Une approche simple consiste à élaborer des règles de décision naïves afin de traiter toutes les situations possibles. Par exemple, le système peut décider d'aller réaliser l'objectif le plus proche si les ressources le permettent. Cette approche est clairement sous-optimale.

La planification probabiliste, ou conditionnelle, est une des solutions envisageables pour calculer un plan hors ligne. Dans ce cas, le plan est un arbre d'actions. Pendant l'exécution, un chemin spécifique est suivi en fonction des événements qui surviennent. Le calcul tient compte des incertitudes sur l'état de l'environnement et sur les effets des actions. Des plans conditionnels sont obtenus grâce aux Processus Décisionnels de Markov ou aux Processus Décisionnels de Markov Partiellement Observables (Section 1.2.3). C'est une approche où les règles de décision sont optimisées grâce à l'utilisation d'un modèle probabiliste.

Le traitement d'un calcul de plans hors ligne est assez élémentaire. Toutes les règles ou politiques sont calculées en préparation de mission. Suite à un changement de situation, le planificateur renvoie de manière quasi immédiate une action ou un plan d'actions à exécuter.

Le calcul de plans en ligne est effectué pendant l'exécution de la mission. Il est défini

comme une séquence d'actions. Le nombre d'actions dans la séquence est variable : la séquence peut se limiter à une action unique [Damiani, Verfaillie, & Charmeau 2004] et peut contenir des actions permettant d'atteindre un état destination pour la mission [Chanthery, Barbier, & Farges 2004]. Plusieurs approches sont envisageables :

- La planification continue : le planificateur est tout le temps actif. Il travaille perpétuellement à améliorer le plan en cours tout en considérant l'évolution de la situation. Les actions ne sont décidées définitivement qu'à leur date d'application.
- La planification sur événements : le planificateur n'est actif qu'après un changement significatif de la situation. Il calcule un plan pendant un intervalle de temps. A la fin de cet intervalle, la suite d'actions du plan est *a priori* décidée. Les actions seront appliquées, à moins qu'un changement de situation ne survienne. Les ressources informatiques du système ne sont sollicitées que sur événements. Si la mission se déroule de façon nominale, le module de décision/planification n'est jamais actif.

Dans ces deux cas se posent les problèmes suivants :

- réutilisation des calculs relatifs au plan précédent en cas de changement de situation ;
- obtention d'un plan dans un intervalle de temps.

En ce qui concerne le premier problème, Les algorithmes de type D^* ou LPA^*, présentés dans la section 2.2 permettent par exemple de replanifier une solution optimale sujette à des contraintes globales en moins d'une seconde. Cependant, si les modifications sont significatives, l'algorithme utilisé est un A^* car les calculs doivent tous être repris du début.

En ce qui concerne le second problème, il est intéressant de noter que certains algorithmes possèdent des propriétés spécifiques. Contrairement à un algorithme standard, où aucun résultat n'est accessible avant la fin de son exécution, un algorithme *anytime* peut être stoppé à tout instant, et doit fournir des solutions de qualité croissante au cours du temps. Un algorithme anytime organise ses calculs de façon à pouvoir donner une réponse dès qu'on la lui demande et à continuer à améliorer les calculs tant qu'on ne lui demande pas de réponse. L'approche anytime réalise un compromis entre le temps de calcul et la qualité de la réponse. On distingue généralement deux types d'algorithmes anytime : les algorithmes contractuels et les algorithmes interruptibles. Un algorithme contractuel nécessite que le contrat de temps alloué soit connu et fixé au début de la résolution. L'objectif, pour un tel algorithme, est de fournir la meilleure solution possible au terme du contrat de temps ; s'il est interrompu avant la fin de son contrat de temps, aucune garantie n'est fournie sur la qualité de la solution produite. Pour un algorithme interruptible, le temps disponible est (*a priori*) inconnu à l'avance et la résolution peut être stoppée à tout instant. Il est donc primordial, pour un tel algorithme, de pouvoir retourner à tout instant la meilleure solution possible et d'améliorer la qualité de cette solution au fur et à mesure du temps. Cette approche se veut plus souple et incrémentale, en produisant des résultats partiels de façon quasi continue. Elle est utilisée par exemple pour la gestion de la mission de satellites d'observation de la terre [Lemaitre & Verfaillie 1997], [Damiani, Verfaillie, & Charmeau 2004].

La figure 3.2 résume les différentes approches évoquées.

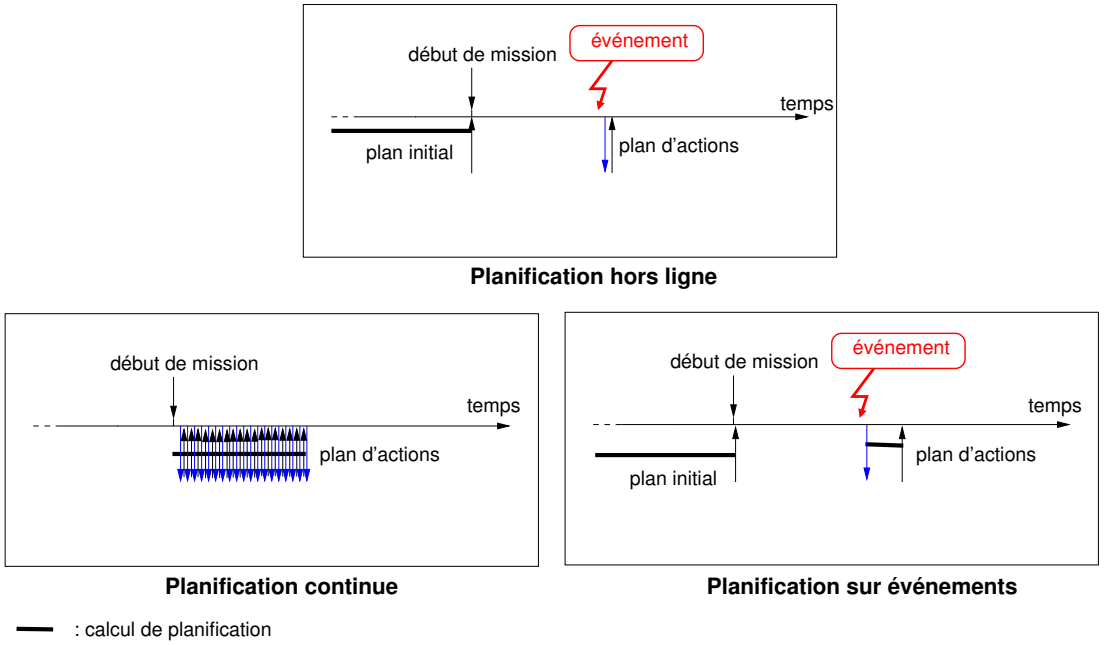


FIG. 3.2 – Les différentes approches de planification

Les applications pratiques ont rendu nécessaires les études sur les liens entre le calcul de plans et leur exécution. Le contexte de ces études remet en cause les hypothèses généralement adoptées en planification, telles qu'un environnement statique et aucun d'échec. En effet, de nouveaux événements peuvent se produire et remettre en cause le plan en cours ; l'exécution des actions prévues peut échouer et la planification doit modifier le plan courant. C'est ce qui motive le choix de la planification en ligne. L'architecture du système autonome doit lui permettre de s'adapter de manière asynchrone à la mise à jour de l'état de l'environnement et de système lui-même.

3.2.3 Les architectures des systèmes autonomes

D'après [Hassoun 1997], on peut distinguer trois classes d'architecture :

- Les architectures *réactives* ne comportent pas de niveau de planification. Le robot réagit uniquement aux stimuli externes de l'environnement. L'absence de modélisation de l'environnement des architectures réactives conduit à une navigation réactive rapide et de ce fait très utile pour un véhicule évoluant dans un environnement hautement dynamique. Ce point n'est pas vérifié dans les architectures hiérarchiques. Un inconvénient majeur pour les architectures réactives est qu'il n'est pas évident de faire prendre en compte une mission par le véhicule (c'est-à-dire un comportement de haut niveau de type "aller-à").
- Les architectures *cognitives* ou *délibératives* à planification de plans d'action comportent plusieurs niveaux décisionnels dont un niveau de planification de plans

d'action pour le robot.
- Les architectures *hybrides* combinent les avantages des deux approches précédentes : elles visent à prendre en compte un ordre de haut niveau et sont réactives.

Les architectures réactives

Dans les architectures réactives, l'engin ne dispose pas d'une tâche particulière à effectuer, il n'y a donc pas de plan d'actions à exécuter. L'engin ne fait que réagir aux stimuli externes de son environnement. Les architectures réactives représentent le fonctionnement de l'agent au moyen de composantes avec une structure de contrôle simple, et sans représentation évoluée des connaissances de l'agent. L'intelligence de l'agent est vue comme le résultat des interactions entre ces composantes et l'environnement. Cela veut dire qu'une telle architecture peut résoudre des problèmes complexes, qui normalement demandent un comportement intelligent, sans traiter l'intelligence du point de vue classique de l'intelligence artificielle. Cette approche est apparue comme une alternative aux critiques visant les approches symboliques, notamment au niveau de la complexité des calculs, qui paraît incompatible avec les ressources limitées des agents, et la difficulté de trouver le bon modèle cognitif pour certaines applications.

L'architecture de subsomption L'architecture réactive la plus connue et la plus influente est celle proposée par Rodney Brooks au MIT [Brooks 1986], appelée architecture de subsomption, en anglais Subsumption Architecture. Elle comporte plusieurs modules de compétence, chaque module étant responsable de la réalisation d'une tâche simple (par exemple "éviter la collision avec les objets", "se déplacer au hasard", ...). Les modules sont organisés de manière hiérarchique, chacun ayant une priorité différente. Les modules de haut niveau réalisent les tâches les plus abstraites, détaillées à l'aide de tâches plus concrètes et plus simples. Les modules de haut niveau ont une priorité plus petite que les modules de bas niveau, correspondant aux tâches les plus simples.
Par exemple (Figure 3.3), pour un robot mobile ayant pour but l'exploration de la planète Mars, on peut définir trois modules, correspondant à trois niveaux hiérarchiques :
- $M0$: éviter la collision avec d'autres objets ;
- $M1$: se déplacer dans l'environnement tout en évitant les obstacles ;
- $M2$: explorer l'environnement en utilisant les observations visuelles pour choisir les lieux intéressants et les déplacements du niveau $M1$.

Un module de bas niveau a une priorité plus grande qu'un module situé sur un niveau plus élevé parce qu'il est responsable d'une tâche plus simple mais plus critique. Dans cette optique, le fonctionnement d'un module situé sur un niveau supérieur est subordonné à un module inférieur. Un module de bas niveau peut ainsi modifier l'entrée d'un module de haut niveau (*suppression*) et même invalider l'action du module supérieur (*inhibition*).

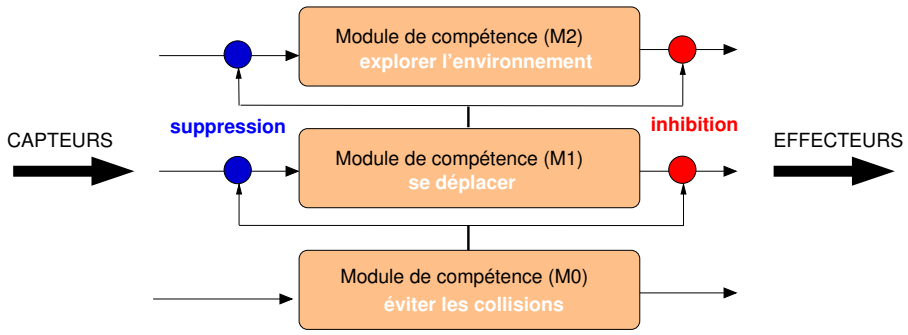


FIG. 3.3 – Architecture de subsomption

L'architecture DAMN (Distributed Architecture for Mobile Navigation) Présentée en 1995, elle est issue des travaux de Rosenblatt [Rosenblatt 1995] de l'Université Carnegie Mellon (CMU). Basée sur l'utilisation de comportements sensori-moteurs, elle constitue une évolution de l'architecture de subsomption. Elle utilise la méthode de vote. Comme le montre la figure 3.4 tirée de [Dalgalarrondo 2003], un ensemble de comportements, composé de comportements de sécurité et de comportements de navigation, envoie ses votes vers deux arbitres. Les arbitres établissent respectivement la consigne de direction et la consigne de vitesse du déplacement de l'engin. Un gestionnaire de mode affecte, éventuellement dynamiquement, des poids à chaque vote de façon à créer des priorités entre les comportements. Ceux-ci sont toujours actifs. Ils fonctionnent simultanément et de manière asynchrone. A partir des votes reçus et de leur poids respectif, les arbitres choisissent les commandes qui ont reçu le plus de vote et les envoient aux contrôleurs des actionneurs de l'engin.

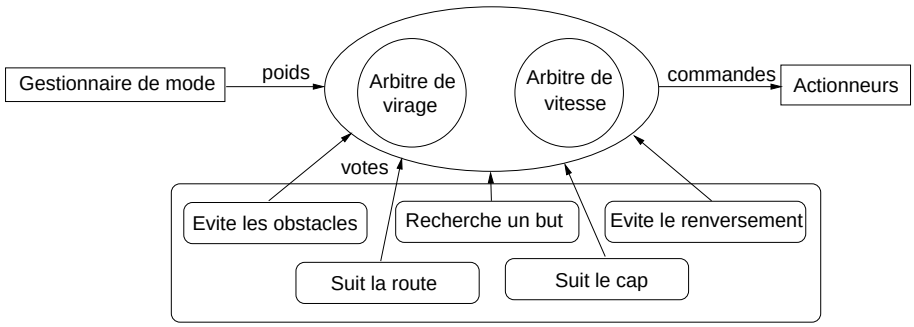


FIG. 3.4 – Exemple d'architecture de type DAMN

L'architecture DAMN présente quelques inconvénients. Ainsi, les incertitudes liées au vote, c'est-à-dire la confiance qu'on peut leur accorder, n'apparaissent pas. L'architecture ne propose aucun mécanisme pour désactiver un comportement inadapté à un contexte ou incohérent avec un autre comportement. En outre, comme dans l'architecture de subsomption, les comportements ne mettent pas en commun leurs connaissances. Il peut donc y avoir une mauvaise utilisation des ressources de calcul. Par ailleurs, dans le cas de comportements plus nombreux ou de tâches plus complexes, la méthode d'at-

tribution des poids semble difficile à définir.

Les architectures réactives ont l'avantage de la simplicité et de l'efficacité de calcul. Le fonctionnement en parallèle et la gestion très simple du choix du comportement actif procure une grande réactivité. Ce parallélisme est aussi bien adapté à ce que l'on attend d'un engin autonome, c'est-à-dire la capacité à mener plusieurs actions en même temps comme par exemple effectuer une observation et suivre un plan de vol.

Pourtant, elles présentent plusieurs limitations qui empêchent leur utilisation dans de nombreuses classes d'applications. Les principales objections contre les architectures réactives sont les suivantes : les agents ont une vision de courte durée sur la résolution du problème, ils peuvent ne pas toujours choisir la meilleure action à exécuter à un certain moment ; comme les agents réactifs ne maintiennent pas une représentation de l'environnement, ils ne peuvent pas manipuler des buts, organiser un plan d'action, et encore moins doter les états d'utilité ; la modélisation par modules de compétence hiérarchisés peut entraîner des interactions imprévisibles entre les niveaux s'il y a un trop grand nombre de modules : dans ce cas, même le grand avantage de cette architecture, l'émergence de l'intelligence par interaction, peut devenir un désavantage, car des interactions indésirables peuvent émerger.

Les architectures réactives ont l'avantage de la simplicité et de l'efficacité de calcul. Le fonctionnement en parallèle et la gestion très simple du choix du comportement actif procure une grande réactivité. Cependant, comme les agents réactifs ne maintiennent pas une représentation de l'environnement, ils ne peuvent pas manipuler des buts, organiser un plan d'action, et encore moins doter les états d'utilité. Cet inconvénient permet de dire qu'elles ne peuvent pas être utilisées dans le cadre de la réalisation d'une mission menée par un véhicule autonome.

Les architectures cognitives ou délibératives

Dans ce type d'architecture, le robot doit réaliser de manière autonome une tâche particulière. Le robot est doté d'une fonction de délibération, appelée aussi planification. Les architectures comportent plusieurs niveaux décisionnels, dont un niveau de planification de plans. Les premières architectures délibératives sont basées sur une boucle ouverte des fonctions "Perception/Planification/Action" : percevoir l'environnement, transformer les données sur l'environnement en un modèle, générer un plan à réaliser basé sur ce modèle, puis exécuter le plan. L'exécution du plan en boucle ouverte est un inconvénient majeur dans les environnements dynamiques. Les architectures plus récentes intègrent donc toutes des composantes réactives : même si l'architecture est fortement délibérative, elle peut aussi être considérée comme hybride.

On présente ici deux architectures à composantes fortement délibératives.

L'architecture TCA (Task Control Architecture) Cette architecture, présentée par R.G. Simmons [Simmons 1994], est construite autour d'une structure appelée arbre des tâches. Un arbre des tâches est une représentation hiérarchique de la réalisation d'une tâche, décomposée en plusieurs sous-tâches. La figure 3.5 montre que TCA comporte un nombre arbitraire de modules, spécialisés chacun dans la réalisation d'une tâche précise, et dialoguant avec un module de gestion central. Ce processus est responsable de l'acheminement des messages entre les modules et du maintien des informations internes. Il doit aussi répartir les tâches en accord avec les contraintes temporelles imposées par le système. Les arbres des tâches sont construits dynamiquement. Cette organisation donne au système un fort caractère centralisé.

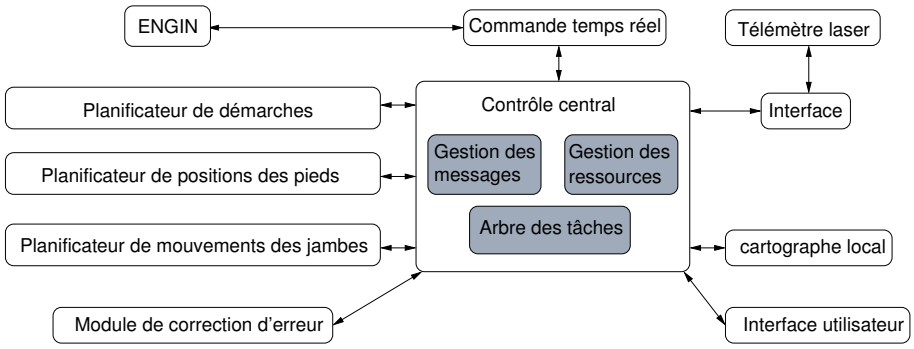


FIG. 3.5 – L'architecture TCA appliquée au robot à pattes AMBLER

La figure 3.5 représente l'application de l'architecture TCA à la gestion du robot à pattes AMBLER. Dans cette application, l'accent est mis beaucoup plus sur la planification que sur la réactivité. Les différents modules sont spécialisés dans des tâches de planification particulières. Les seuls traitements réellement réactifs concernent des actions entreprises lorsque des exceptions sont déclenchées (stabilisation lors d'un glissement par exemple).

Le reproche généralement fait à cette architecture provient de la centralisation du traitement et du mécanisme de communication par messages. En effet cela peut impliquer des temps de réponse importants incompatibles avec des applications sur engins rapides. Pour éviter ces inconvénients, des actions réflexes d'urgence peuvent être implémentées.

L'architecture LAAS L'architecture LAAS [Alami *et al.* 1998], illustrée figure 3.6, comporte trois niveaux. Elle a été implantée sur les robots Hilare 1.5, 2 et 2-bis, et sur le robot Dala.

- Le niveau décisionnel contient les éléments de décision que sont le superviseur et le planificateur. Des travaux récents [Lemai, Ingrand, & Gallien 2005] sont encore menés sur la couche décisionnelle, et notamment sur le planificateur IxTeT-EXEC. Ce dernier produit un plan pour réaliser un ensemble d'objectifs donné par l'opérateur. La partie temporelle de IxTeT décide quand lancer ou arrêter une action

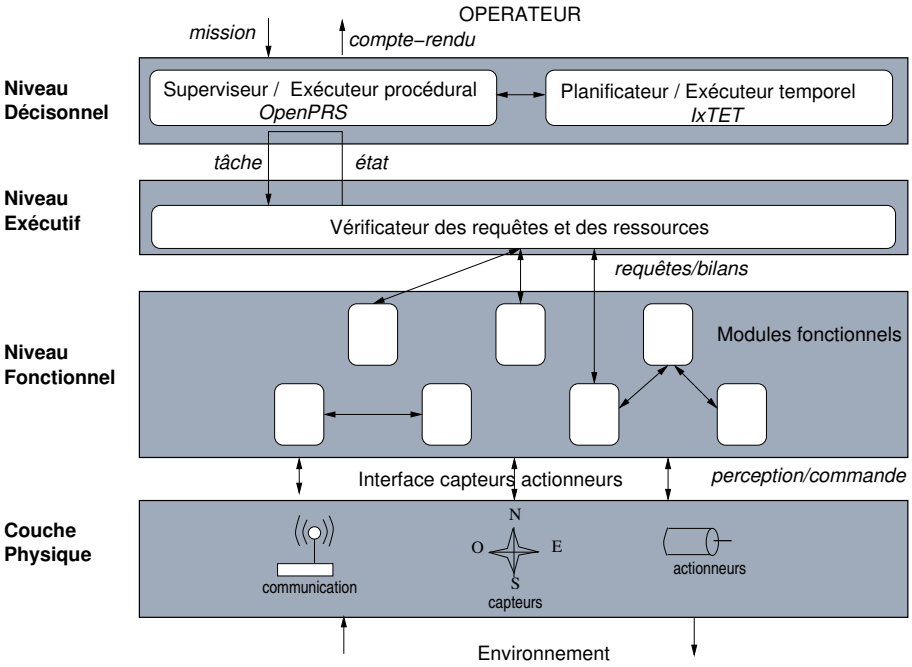


FIG. 3.6 – Architecture LAAS

dans le plan et effectue des modifications du plan. Un exécutif procédural (OpenPRS) développe et raffine l'action en commandes envoyées au niveau fonctionnel, vérifie son exécution et peut gérer quelques échecs spécifiques.

- Le niveau exécutif fait le lien entre le niveau décisionnel et le niveau fonctionnel. Il filtre les requêtes suivant l'état courant du système et un modèle formel des états autorisés ou interdits.
- Le niveau fonctionnel inclut des modules permettant au robot d'effectuer toutes ses actions et lui donnant ses capacités de perception. Ces modules sont activés par des requêtes des niveaux supérieurs.

Dans cette architecture, les actions réflexes sont principalement associées à la planification conditionnelle. Le niveau décisionnel a la charge de prévoir des alternatives aux plans et de les déclencher au moment opportun. Toutefois, chaque module fonctionnel dispose d'un mécanisme de traitement d'exceptions qui lui permet de modifier ses paramètres ou d'activer un autre module. Ainsi, cette architecture intègre un mécanisme réactif au niveau fonctionnel. Elle peut ainsi être considérée comme hybride, même si les flux de commandes qui aboutissent aux modules fonctionnels sont majoritairement descendants.

Les architectures purement délibératives posent un certain nombre de problèmes :
- leur principal inconvénient vient de leur faible vitesse de réaction. Elles sont incapables de prendre en compte la dynamique de l'environnement. Les architectures étudiées ici proposent donc des actions réflexes. Elles ne sont pas purement délibératives.
- Les différences potentielles entre le modèle sur lequel se base la planification et le monde réel introduisent des incertitudes fortes sur la position de l'engin et sur l'environnement. La prise en compte de ces incertitudes, indispensable pour obtenir un système robuste, rend la planification beaucoup plus complexe.

L'utilisation de l'approche délibérative pure semble donc limitée aux engins autonomes évoluant dans des environnements statiques et dont la structure est fortement contrainte et connue *a priori*. Elles ne peuvent donc pas être utilisées dans le cadre de notre travail. Cependant, leur point fort reste leur capacité à prendre en compte des raisonnements de haut niveau. Il leur est possible de gérer des missions complexes enchaînant plusieurs objectifs successifs. Cet aspect doit être intégré dans notre architecture.

Les architectures hybrides

Une architecture hybride est composée d'un ensemble de modules organisés dans une hiérarchie, chaque module étant soit une composante cognitive avec représentation symbolique des connaissances et capacités de raisonnement, soit une composante réactive. De cette manière, on combine le comportement pro-actif de l'agent, dirigé par les buts, avec un comportement réactif aux changements de l'environnement. En plus, on espère obtenir simultanément les avantages des architectures cognitives et réactives, tout en éliminant leurs limitations. Les développements les plus récents convergent vers l'utilisation d'architectures à deux ou trois niveaux.

Les architecture 3T et ATLANTIS Les architectures 3T de la NASA [Bonasso *et al.* 1997] et ATLANTIS [Gat 1992] présentent de nombreuses similitudes. On retrouve dans ces architectures trois parties principales (Figure 3.7) : un planificateur pour la partie délibérative, un ensemble de mécanismes de commande réactifs et un niveau intermédiaire chargé de gérer la séquence d'actions.

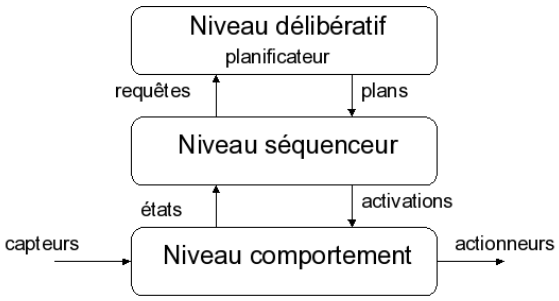


FIG. 3.7 – Architecture à trois niveaux

- La couche basse rassemble plusieurs algorithmes implémentant des comportements réactifs simples tels que le suivi de trajectoire ou l'évitement d'obstacle. Ces com-

portements ne gèrent aucune mémoire sur l'état interne du robot. Ils constituent une bibliothèque de "talents" utilisés à la demande par les niveaux supérieurs.
- Le niveau intermédiaire doit sélectionner quel comportement ou quel groupe de comportement doit être actif à un moment donné : il joue le rôle de séquenceur en effectuant l'ordonnancement des tâches. Il peut également fournir les paramètres utiles à l'exécution des comportements. Pour 3T, le niveau est implémenté en utilisant le langage RAP (Reactive Action Packages) proposé par R. J Firby [Firby 1987]. RAP est un langage implémenté en LISP permettant la spécification d'un ensemble de séquences d'actions activées lorsque certaines conditions sont vérifiées. Pour ATLANTIS, le niveau s'appuie sur le langage ESL [Gat 1997], dont le fonctionnement est assez semblable à celui de RAP.
- La couche supérieure s'exécute dans une échelle de temps différente des deux autres niveaux. Elle a pour but de fournir les séquences d'actions utilisées par la couche intermédiaire. Pour 3T, elle produit à l'avance le plan utilisé par la couche d'ordonnancement. Dans ATLANTIS, le plan est produit sur demande en fonction de requêtes émises par le niveau intermédiaire.

Les trois niveaux fonctionnels de ces architecture se retrouvent dans beaucoup d'architectures pour les robots récents ou les systèmes intelligents [Hayes-Roth 1995]. Le niveau intermédiaire peut être vu comme une interface entre les composantes réactives et délibératives du système. Il doit intégrer à la fois une capacité de réaction rapide aux événements internes ou externes, et une possibilité de représenter des informations de haut niveau.

L'architecture CLARAty développée au Jet Propulsion Laboratory (JPL) [Nesnas *et al.* 2003], est implantée sur les rovers Rocky 8, FIDO et Rocky 7. Elle est aussi utilisée sur le rover K9 au NASA Ames Research Center (ARC) et sur le rover ATRV à l'Université Carnegie Mellon (CMU).
CLARAty est composée de deux niveaux : un niveau fonctionnel, et un niveau décisionnel regroupant les niveaux exécutif et décisionnel d'une architecture trois niveaux classique.

Le niveau fonctionnel est conçu suivant une approche objet, pour traduire la modularité de la couche matérielle et permettre une granularité d'abstraction au niveau décisionnel. L'état du système est gardé séparément dans chaque objet. Ils sont interrogés par le niveau décisionnel quand il a besoin d'une information.

Le niveau décisionnel prend ses décisions compte tenu des ressources du système et des contraintes de la mission. Il inclut les fonctions de planification, d'exécution, d'ordonnancement et les bases de données des activités. Il planifie, ordonnance et exécute le plan des activités. Ce niveau décisionnel unique et générique a pour but d'éviter la gestion d'un modèle du système différent pour chaque niveau en unifiant les représentations des activités, et de permettre un meilleur contrôle des domaines décisionnels concurrents réactif/planification. Cela permet de tenir compte des tendances actuelles des systèmes de planification qui incorpore des capacités de contrôle d'exécution. Ces développements visent des capacités de planification plus développées afin de permettre

des missions autonomes de longue durée.

L'achitecture HARPIC développée au CTA [Luzeaux & Dalgalarrondo 2001] pour un robot de type Pioneer AT est constituée de deux niveaux, illustrés figure 3.8.

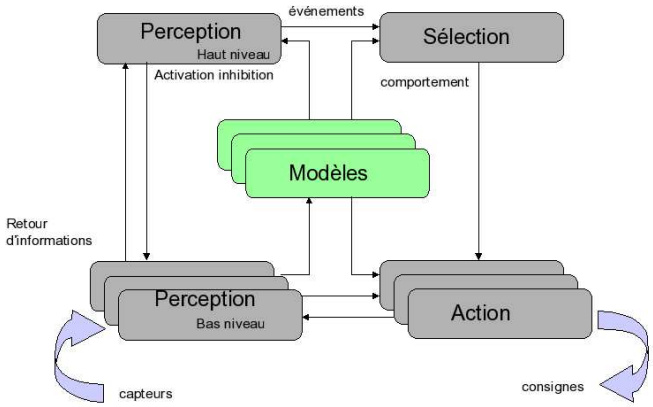


FIG. 3.8 – Architecture HARPIC

Le niveau bas contient les fonctions de perception et d'action. A partir des données issues des capteurs, la fonction de perception bas niveau modélise l'environnement. Cette modélisation est mise à disposition des autres fonctions de l'architecture. Le niveau bas permet de définir plusieurs paires "perception/action" indépendantes, gérant un comportement. Cette approche est donc très modulaire.

Le haut niveau est constitué de fonctions de perception et de sélection de comportement. La fonction de sélection a pour rôle le choix et l'activation (ou l'arrêt) des comportements du robot. L'originalité de l'architecture tient principalement dans la fonction de perception au haut niveau. Elle permet, indépendamment du comportement en cours d'exécution, de percevoir l'environnement et donc de veiller à la sécurité du robot et d'être réactif aux opportunités de l'environnement du robot. Cette fonction signale des événements à la fonction de sélection lorsqu'un fait nouveau apparaît ou qu'une erreur survient. Elle a aussi pour tâche d'utiliser au mieux les ressources de calcul du robot. Les tests de l'architecture sur de petits robots montrent qu'une telle fonction permet d'accroître la capacité d'adaptation du robot. Cette architecture souligne le fait qu'il est avantageux d'être attentif à l'environnement à tous les niveaux de l'architecture pour permettre une meilleure réactivité et optimiser les ressources du robot.

Les architectures hybrides combinent un comportement délibératif du système avec un comportement réactif aux changements de l'environnement. Ce type d'architecture est le plus adapté aux contraintes de notre travail. Cependant, les architectures présentées semblent trop générales pour que l'une d'elles soit utilisée directement dans le cadre d'une mission menée par un véhicule autonome. Nous allons donc proposer une architecture hybride dédiée à réalisation d'une mission.

3.2.4 L'environnement logiciel ProCoSA

Développé par l'ONERA, ProCoSA (PROgrammation et COntrôle des Systèmes à forte Autonomie)[Barrouil & Lemaire 1998] [Barbier, Lemaire, & Toumelin 2001], est un environnement logiciel destiné à intégrer et contrôler de manière souple et modulaire les différentes fonctions des systèmes fortement autonomes organisés dans la double boucle autour d'un superviseur. Ce logiciel permet de concevoir une architecture de contrôle (Figure 3.9) intégrant un certain nombre de sous-systèmes indépendants, codés dans des serveurs, correspondant à des fonctions algorithmiques. Ces fonctions algorithmiques peuvent correspondre à des fonctions délibératives ou envoyer des actions vers la couche physique. ProCoSA permet donc de développer des architectures hybrides.

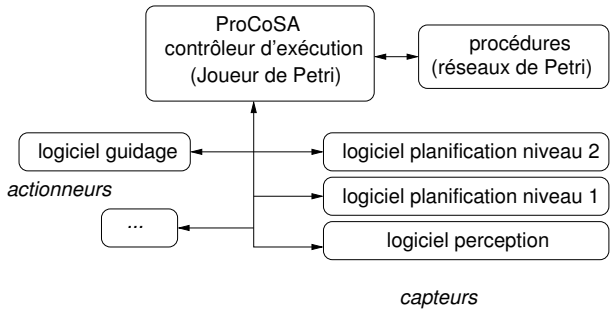


FIG. 3.9 – Architecture hybride basée sur ProCoSA

Hors ligne, le logiciel ProCoSA permet de spécifier les procédures de coopération entre les sous-systèmes et de les coder par réseaux de Petri en vue de leur mise en œuvre embarquée. Les réseaux de Petri [Murata 1989] sont des graphes orientés avec deux types de nœuds : des places et des transitions. Un arc relie soit une place et une transition, soit une transition et une place. Il ne relie jamais une place et une place ou une transition et une transition. Le marquage du graphe est donné par la présence de jetons dans les places : on parle alors de places marquées. La figure 3.10 donne l'exemple d'un réseau avec quatre places et deux transitions. La place $P1$ est marquée : elle possède un jeton.

DÉFINITION 12 *Une transition est dite* ***validée*** *si toutes les places en amont (c'est-à-dire en entrée) de celle-ci possèdent au moins un jeton.*

DÉFINITION 13 *Si une transition est validée, alors la transition est* ***franchissable****. Le* ***franchissement ou tir*** *consiste à retirer un jeton dans chacune des places en entrée de la transition et à ajouter un jeton dans chacune des places en sortie.*

Les réseaux de Petri sous ProCoSA ont la signification suivante :
- une place = un état, une activité élémentaire ou une macro-activité ;
- une transition = un changement d'état.

La décision peut ainsi être répartie dans une hiérarchie de réseaux de Petri, la marquage de certaines places d'un réseau supérieur représentant l'activation d'un réseau

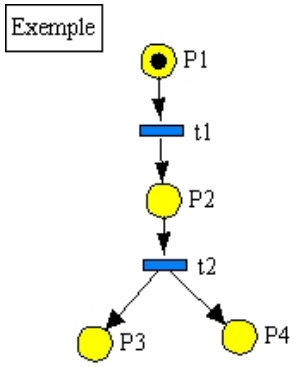


FIG. 3.10 – Exemple de réseau de Petri

plus bas dans la hiérarchie.
Le développeur a la possibilité d'associer à chaque transition un ou plusieurs événements et/ou une ou plusieurs requêtes d'activation (Figure 3.11). Le superviseur tire une transition si elle est validée et si un des événements associés est reçu. Quand une transition est tirée, les places précédentes sont démarquées, les places suivantes sont marquées et le superviseur déclenche la ou les requêtes d'activation associées à la transition.

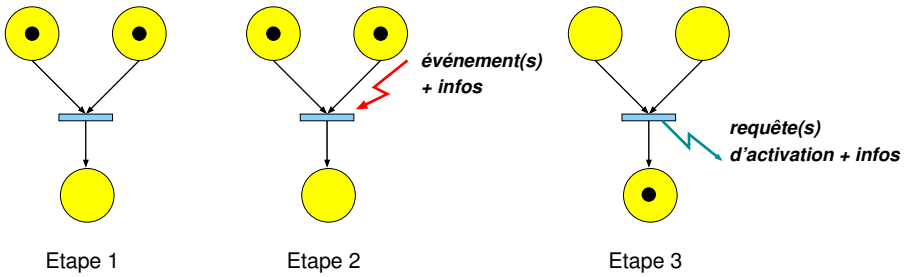


FIG. 3.11 – franchissement d'une transition sous ProCoSA - Etape 1 : la transition est validée ; Etape 2 : la transition est validée, un des événements associés à la transition est reçu ; Etape 3 : franchissement de la transition : les places en entrée sont démarquées, les places en sortie sont marquées, les requêtes d'activation associées à la transition sont envoyées

Une interface graphique appelée EdiPet permet la saisie hors ligne des procédures et l'affichage en ligne de l'état des procédures.
Le Joueur de Petri de ProCoSA est le superviseur du système. Central, il gère l'évolution du marquage des réseaux et les communications. Il réceptionne et traite les événements émis par les réseaux de Petri ou les serveurs, et émet les requêtes d'activation destinées à un réseau de Petri ou à un serveur. Les flux d'événements et de requêtes constituent une sorte de commande en boucle fermée comparable à celle de l'automatique.

Le logiciel ProCoSA présente à l'implémentation une architecture client/serveur. Le superviseur déroule des procédures prédéfinies, codées dans des réseaux de Petri, pour activer opportunément les serveurs lors de l'occurrence de certains événements.

ProCoSA a été validé dans le cadre d'une application réelle, puisqu'il a été utilisé pour concevoir l'architecture de contrôle d'un véhicule sous-marin autonome effectuant une mission d'observation [Barrouil & Lemaire 1998], [Barbier, Lemaire, & Toumelin 2001].
Le logiciel a aussi été utilisé pour réaliser l'architecture d'un banc test d'un véhicule aérien autonome [Barbier & Chanthery 2004]. L'architecture décrite dans ce papier prévoit un module de planification calculant un nouveau plan en cas d'aléa ainsi qu'un module de guidage. Le déroulement d'une mission est spécifié par une hiérarchie de réseaux de Petri de ProCoSA. Un réseau, nommé "Mission" décrit le comportement général du véhicule du décollage jusqu'à l'atterrissage. Une place marquée dans ce réseau indique soit la phase de haut niveau dans laquelle est le véhicule soit une action de haut niveau :

- quatre places symbolisent les quatre phases de navigation ;
- un ensemble de places détaillent le suivi d'un plan, avec la boucle de navigation vers la nouvelle zone d'opération, jusqu'à l'épuisement des zones objectifs dans le plan ;
- les autres places détaillent le comportement du véhicule pendant la planification et la replanification.

La faisabilité d'une telle architecture est montrée lors de tests simulant des aléas pendant l'exécution de la mission.

3.2.5 Conclusions

L'analyse de l'existant indique que l'on doit distinguer planification hors-ligne, planification continue et planification sur événement. Il semble que la planification continue soit plutôt adaptée à des véhicules pour lesquels la mission ne comprend pas une terminaison explicite et n'est pas brusquement modifiée. Dans le type de mission abordé ici, la consommation des ressources conduit à une fin explicite et les modifications sont importantes et d'occurrence imprévisible. Il est donc logique d'envisager une planification sur événement.

En ce qui concerne les architectures, il semble que l'on ne puisse pas échapper à la définition de différents nivaux dans une architecture à la fois fonctionnelle et fréquentielle. En effet, ces niveaux se retrouvent dans certaines architectures cognitives et réactives et on distingue dans les architectures hybrides au moins un niveau fonctionnel et un niveau décisionnel. Ce travail s'inscrit donc dans la logique du découpage en niveaux. Par ailleurs, les avantages des architectures hybrides sont indéniables. Ce mémoire se place donc dans ce cadre.

Finalement, il est pratique et sûr de pouvoir définir une architecture en s'appuyant sur une description formelle. Le logiciel ProCoSA permet ces facilités en exécutant directement des enchaînements décrits par des réseaux de Petri. Etant à notre disposition, il est donc utilisé dans ce travail.

L'analyse de la littérature concernant les liens entre la planification et l'exécution amène à effectuer les choix suivants :
- Les calculs de planification seront déclenchés sur événements.
- Une architecture hybride à plusieurs niveaux doit être développée. Cette architecture embarquée sera dédiée à la réalisation d'une mission par un véhicule autonome.
- Le logiciel ProCoSA est choisi pour développer cette architecture.

3.3 Fonctionnalités de l'architecture de contrôle

3.3.1 Effectuer la mission

Pour que le système puisse réaliser une mission décrite par des objectifs, il doit posséder les fonctionnalités suivantes.

1. Prendre en compte :
   - des données opérateur ;
   - des données des capteurs : état du véhicule, état des ressources, ...
   Ainsi, le système doit pouvoir obtenir toutes les informations nécessaires pour lancer un calcul de planification.
2. Quand le plan est trouvé, savoir le traiter, c'est-à-dire mettre en œuvre l'enchaînement des segments définis par la planification dans le bon ordre et aux instants voulus.
3. Faire les calculs pour rallier correctement les points de passage, trouver les dates adéquates pour activer et désactiver la charge utile.
4. Calculer des consignes pour guider correctement l'engin vers les points de passage trouvés. Envoyer des consignes pour activer et désactiver la charge utile.
5. Contrôler le bon déroulement des phases de calculs et d'actions.

3.3.2 Réagir aux aléas

Afin de réagir aux aléas, l'architecture doit être capable de les détecter lorsqu'ils surviennent, et ce dans un laps de temps en accord avec les exigences temporelles du système. La détection des aléas ne fait pas partie de ce travail. Une fois détectés, les aléas doivent être pris en compte : l'architecture doit lancer des calculs correctifs aux niveaux de décision adaptés et appliquer ces corrections.

Les aléas prévus sont de quatre types.

1. Des modifications sur la carte des dangers ou sur la carte de mission. Ces modifications peuvent survenir à cause :
   - d'une mise à jour effectuée par l'opérateur ;
   - d'une détection par la charge utile ;
   Ces informations doivent être comparées aux cartes de la base de données. En cas de changement, la base de données doit être mise à jour. Lorsque la base de données est mise à jour, il faut déterminer si le plan courant est toujours valide et agir en conséquence.

2. Un calcul impossible au niveau du plan : cela signifie que les contraintes de la mission sont incompatibles avec la réalisation des objectifs. La prise en compte de cette impossibilité s'effectue au niveau de la détermination du contexte de planification. Des contraintes doivent être relâchées de manière à ce que le module de planification puisse proposer un plan valide. Sur l'exemple d'une mission d'observation pour un drone, les contraintes pouvant être relâchées sont par exemple :
   - des contraintes temporelles sur les objectifs ; dans une certaine mesure, la fenêtre temporelle sur laquelle les points sont définis peut être agrandie ;
   - la probabilité d'existence ou la dangerosité d'une menace peut être revue à la baisse.

   Par ailleurs, si en début de mission, le vecteur r_{min} est surévalué (Equation 1.5), il est possible de consommer cette marge en cas de problème grave.
3. Un calcul impossible de trajectoire ou des consignes envoyées à l'engin. Ces impossibilités doivent être traitées au niveau supérieur. Une impossibilité sur le calcul de trajectoire doit déclencher un nouveau calcul de plan, une impossibilité de calcul de consignes doit déclencher un nouveau calcul de la trajectoire.
4. Une non-conformité entre le résultat d'une prévision et les observations des données courantes du système à n'importe quel niveau hiérarchique doit déclencher un calcul à ce niveau. Ainsi la non-conformité de la mission courante avec les observations signifie que la mission a changé, elle déclenche la synthèse d'une nouvelle carte de la mission. De la même façon, une non-conformité du plan courant avec les prévisions signifie que le plan n'a pas pu être suivi, elle déclenche un calcul de plans. Une non-conformité de la trajectoire courante avec la trajectoire prévue déclenche un calcul de trajectoire et une non conformité des consignes courantes avec les consignes prévues déclenche un calcul de consignes. Les non-conformités observées peuvent être dues à des phénomènes non prévus par la planification : vent, variations de vitesse trop importante en une durée courte, perturbations, panne de capteurs, panne moteur, fuite de carburant, dégradation de l'engin due à une destruction partielle, ... Elles peuvent aussi être observées en cas de détection d'une menace sur le plan courant.

3.4 Niveaux de l'architecture hiérarchique

L'idée est de développer une architecture hybride, combinant des éléments délibératifs et des éléments réactifs. Les éléments sont organisés selon une hiérarchie inspirée de l'architecture de sumbsomption et des définitions des niveaux d'autonomie : les éléments de haut niveau correspondent à un niveau d'autonomie plus important, détaillé à l'aide d'éléments plus bas niveau au sens de l'autonomie décisionnelle.
Chaque niveau est construit sur le principe de l'asservissement en boucle fermée de l'Automatique (Figure 3.12) : la sortie est asservie avec une valeur de référence correspondant à la valeur souhaitée. L'erreur est la différence entre la référence et l'estimation de la sortie. L'idéal est que l'erreur soit nulle pour que la sortie soit exactement l'image de la valeur de référence.

Par analogie avec ce principe, la fonction "Estimateur" est remplacée par une fonction

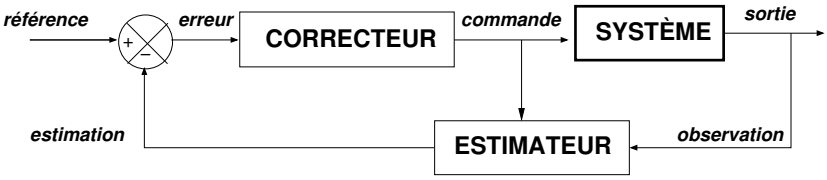


FIG. 3.12 – Principe de l'asservissement d'un système en boucle fermée

"**Prévoir**" qui calcule des données comparables à la référence en fonction des données de sortie et de données du système et de l'environnement. Le comparateur est remplacé par une fonction "**Comparer**" qui distingue s'il existe ou non une différence entre la référence et la sortie estimée. Une fonction "**Corriger**" remplace le bloc de correction : en fonction de la différence entre la référence et la sortie estimée, elle envoie une nouvelle commande au système.

Les niveaux sont numérotés selon leur niveau dans la hiérarchie :
- niveau 3 : gestion de la mission et de l'environnement ;
- niveau 2 : gestion du plan ;
- niveau 1 : gestion de la trajectoire ;
- niveau 0 : guidage.

Une couche physique incluant capteurs et effecteurs et une couche correspondant à la base de données sont définis autour de ces niveaux.

☞ Les capteurs sont définis comme les moyens de mesures des variables relatives au déplacement du véhicule, à ses ressources, à la mission et à l'environnement.

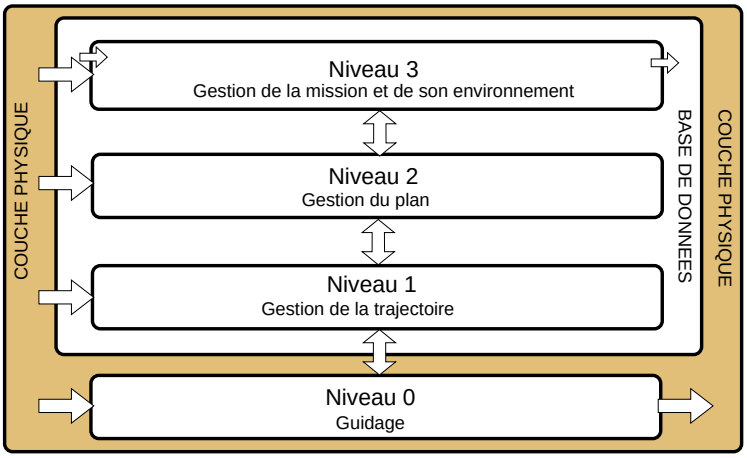


FIG. 3.13 – Organisation hiérarchique des niveaux dans l'architecture, couche physique et couche de la base de données

Les niveaux communiquent selon le principe de la figure 3.13. Les capteurs ont accès à la couche physique : ils ont les informations concernant l'état. Le niveau de guidage

envoie les consignes directement vers la couche physique. La base de données contient les cartes de la mission et des dangers en cours. Elle est mise à jour par le niveau le plus haut et est utilisée par les niveaux 3 à 1.

L'architecture hiérarchisée est ainsi définie par une couche délibérative (niveaux 3 à 1) et une couche réactive (niveau 0).

La hiérarchisation des prises de décision permet d'organiser les tâches de raisonnement. Pour chaque niveau de la hiérarchie, on décrit les échanges de données entre les blocs fonctionnels et les activations de tâches.

Certains aléas détectés à un niveau sont redondants avec les aléas détectés aux niveaux supérieurs. L'architecture traitera en priorité les aléas sur les niveaux les plus hauts. Cela est géré en partant du fait qu'un événement d'un niveau (i+1) vers un niveau (i) provoque l'arrêt des tâches en cours dans le niveau (i).

3.4.1 Niveau 3 : gestion de la mission et de son environnement

Le niveau 3 veille à la conformité de la mission traitée avec les observations de l'environnement. Les figures 3.14 et 3.15 présentent ce niveau. Il a pour entrées les cartes de la base de données, les données capteurs et de l'opérateur. En sortie, il met à jour les cartes de la base de données. Il est activé par des changements dans les bases de données capteurs ou opérateur. Il gère ainsi tous les changements survenus sur l'environnement et détectés soit par les capteurs soit par l'opérateur. Il gère aussi les événements de modification de mission. Le niveau 3 active le niveau 2 en lançant une planification/replanification. Le niveau est constitué de trois fonctions.

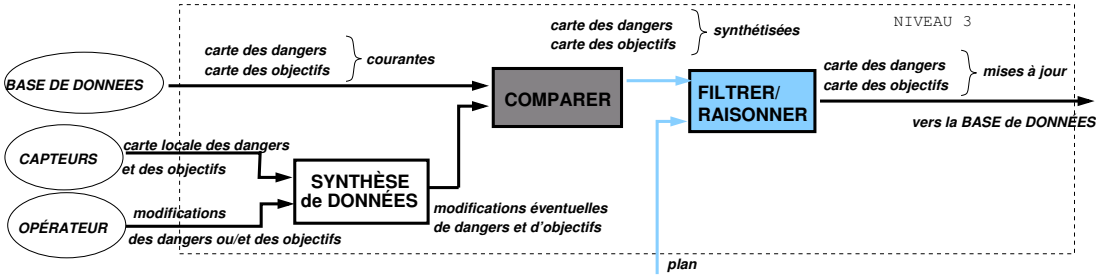


FIG. 3.14 – Niveau 3 : description des échanges de données

Synthèse de données

Cette fonction a pour entrées la carte locale des dangers et des objectifs vus par les capteurs et les modifications d'objectifs et de dangers données par l'opérateur. Les modifications d'objectifs provoquées par les capteurs viennent par exemple d'une fonction de traitement des données permettant de détecter des objectifs intéressants parmi les observations. Le rôle de la synthèse de données est de déterminer les modifications

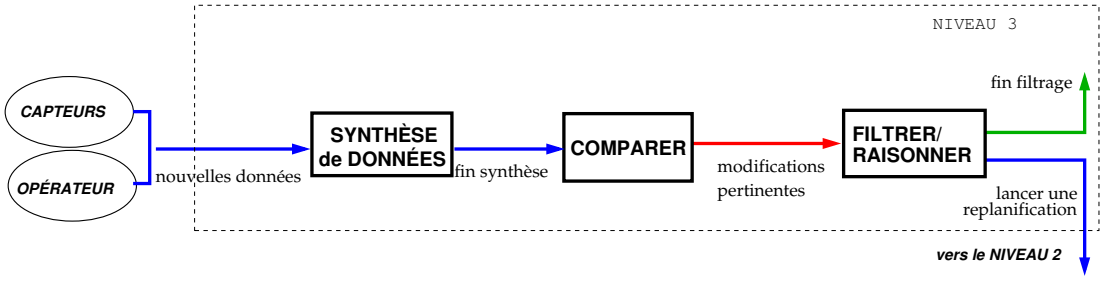


FIG. 3.15 – Niveau 3 : description des activations de tâches

potentielles sur les cartes des dangers et des objectifs.
La fonction est activée lorsque de nouvelles données arrivent des données capteurs et/ou des données opérateur. La fin de la synthèse provoque l'activation de la fonction "Comparer".

Comparer

Cette fonction a pour entrées la carte des dangers et la carte des objectifs courantes présentes dans la base de données, et les modifications éventuelles issues de la fonction "Synthèse de données". Elle a pour but de vérifier que les modifications éventuelles sont pertinentes.
La fonction est activée par la fin d'une "Synthèse de données", elle envoie un ordre d'activation de tâche en cas de modifications éventuelles pertinentes. Dans ce cas, elle produit deux cartes synthétisées à partie des cartes courantes et des modifications pertinentes.

Filtrer/ Raisonner

Cette fonction a pour entrées les cartes produites par la fonction "Comparer" et le plan courant.
Elle est activée en cas de modifications pertinentes détectées par la fonction "Comparer". Elle détermine si les changements observés sont de nature à déclencher une replanification. Trois types de changements sont observés :

- changements de la carte des objectifs : apparition ou disparition d'objectifs, modifications de contraintes temporelles, déplacement des nœuds d'entrée et/ou de sortie, modification des récompenses maximales ;
- changements de la carte des dangers impliquant une variation du critère courant ou de l'utilisation prévue des ressources ;
- changements de la carte des dangers sans conséquence sur le critère courant ou sur l'utilisation prévue des ressources .

Les deux premiers types de changements sont de nature à déclencher une planification. Cependant, seul les changements de la carte des objectifs vont provoquer une réaction de la fonction Filtrer/Raisonner. En effet, les changements de la carte des dangers qui invalident le plan seront détectés par le niveau inférieur, car ils ont un impact sur la consommation des ressources. Ainsi, seuls les changements de la carte des objectifs vont

déclencher un ordre d'activation de tâche vers le niveau de gestion du plan. Néanmoins, dans tous les cas, la fonction Filtrer/Raisonner met à jour les cartes dans la base de données.

3.4.2 Niveau 2 : gestion du plan

Le niveau 2 vérifie la conformité de la situation par rapport au plan grâce à une boucle automatique de contrôle et gère les calculs de plan. Il possède une base de données qui lui est propre contenant le plan courant, le segment de plan traité courant ainsi que l'évolution prévue des ressources pour ce plan au moment où il a été calculé. Les figures 3.16 et 3.17 présentent ce niveau. Il a pour entrées les cartes de la base de données générale, les données des capteurs et le plan courant de la base de données de niveau 2. En sortie, il met à jour le plan dans sa base de données. Il peut être activé soit par une boucle automatique de contrôle interne, soit par le niveau supérieur (demande de calcul d'une planification/replanification), soit par le niveau inférieur (détection d'un segment infaisable). Entre le niveau 2 et le niveau 1, on insère un séquenceur, dont le rôle est de gérer l'enchaînement des segments du plan et la mise à jour de la base de données du niveau 2. Il a pour entrée le plan calculé par le niveau 2 et provoque des calculs de trajectoire (niveau 1) pour chaque segment de plan. Le niveau 2 est constitué de quatre fonctions.

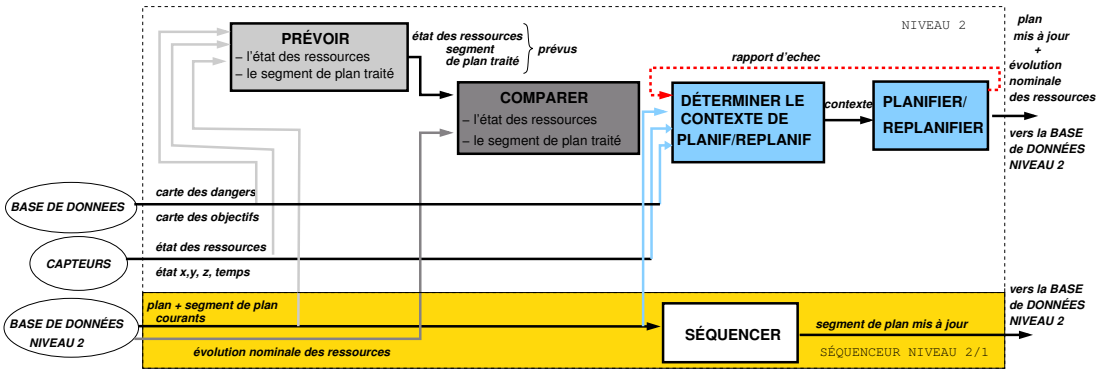


FIG. 3.16 – Niveau 2 : description des échanges de données

Prévoir

Cette fonction a pour entrées les cartes des dangers et des objectifs de la base de données, l'état des ressources et l'état (x, y, z, temps) du système de drone donnés par les capteurs et le segment de plan traité donné par la base de données de niveau 2. Elle calcule l'état des ressources et le segment de plan traité sur l'horizon du plan.
Elle est activée régulièrement par une boucle de contrôle automatique gérée par l'architecture de manière à remettre à jour les données prévues. Pour l'exemple applicatif de cette thèse, la boucle automatique devrait être activée toutes 5 secondes environ, cela correspondrait à une distance de environ 250 m pour l'engin. La fin de la prévision provoque une comparaison.

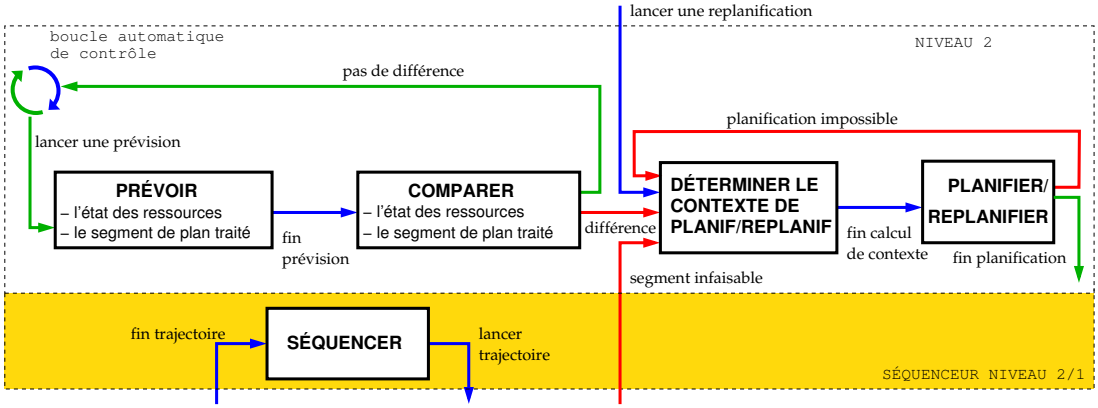


FIG. 3.17 – Niveau 2 : description des activations de tâches

Comparer

Cette fonction a pour entrées l'évolution prévue des ressources et le segment de plan traité prévu d'une part, l'évolution nominale des ressources et le segment de plan traité reçus de la base de données de niveau 2 d'autre part. Cette fonction compare l'évolution des données prévues et leur évolution prévisible aux vues des données courantes.

La figure 3.18 montre la comparaison entre l'évolution prévue d'une ressource et son évolution nominale lorsqu'il y a une différence. Dans ce cas, la fonction "Comparer" détecte une différence.

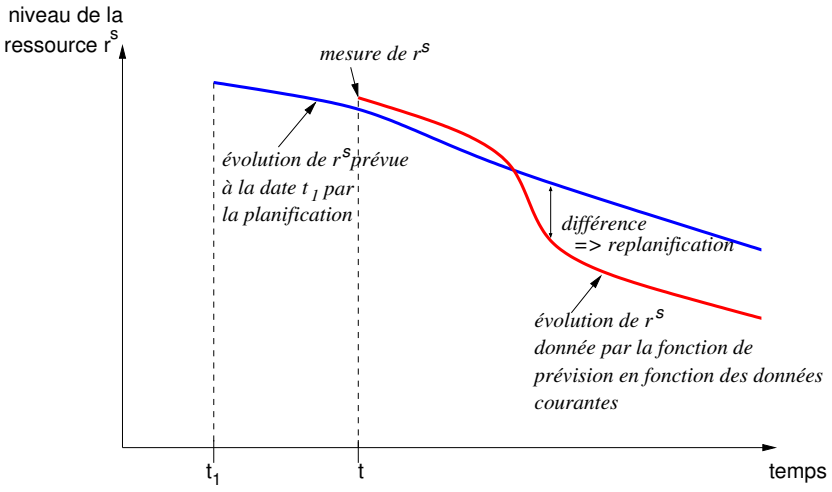


FIG. 3.18 – Comparaison entre l'évolution prévue d'une ressource et son évolution nominale

La fonction est activée par la fin de la fonction "Prévoir", elle envoie un ordre d'activation de tâche en cas de différence entre les données prévues et les données courantes. C'est donc à ce niveau que les changements de la carte des dangers vont être pris en

compte : en effet, les changements d'environnement ont un impact sur la consommation des ressources. C'est à ce niveau que sont détectés des aléas de type fuite de carburant, mauvais suivi de plan (dates prévues incorrectes sur les points de passage impliquant un retard ou une avance par rapport aux dates prévues, manque de temps). En ce qui concerne le temps, l'avance comme le retard peuvent entraîner une replanification. En ce qui concerne les ressources, seul un niveau inférieur au niveau prévu lors de la planification entraîne une replanification (Figure 3.18).

Déterminer le contexte de planification

Cette fonction a pour entrées les cartes des dangers et des objectifs de la base de données, l'état des ressources, l'état (x, y, z, temps) du système de drone des données capteurs, le segment de plan traité et le plan courants de la base de données du niveau. Elle peut aussi avoir pour entrée un rapport d'échec de la fonction de "Planifier/Replanifier". Elle détermine le contexte de planification/replanification. Ce calcul fait l'objet de la section 3.6.1. Le contexte de la mission comprend :

- la carte des objectifs de la mission ;
- l'état prévu au moment où le plan doit être prêt :
  - ⋆ le nœud vers lequel le véhicule est en train de se diriger ;
  - ⋆ la date t ;
  - ⋆ l'état des ressources ;
  - ⋆ l'état de réalisation des objectifs de la mission.
- la carte des dangers ;
- la durée de calcul maximale entre le début et la fin du calcul de replanification.

La fonction est activée soit par le niveau supérieur (événement "lancer une replanification"), soit en cas de détection de différence par la fonction "Comparer", soit par le niveau inférieur en cas de calcul de trajectoire infaisable, soit enfin par une impossibilité de calculer un plan dans le contexte courant envoyée par la fonction "Planifier/Replanifier".

Planifier/Replanifier

Cette fonction a pour entrée un contexte de planification/replanification. Elle contient les calculs de plans décrits au chapitre 2. Elle calcule également une prévision de l'évolution nominale des ressources. Elle a pour sortie un plan et les prévisions qu'elle envoie vers la base de données dédiée au niveau 2. En cas de calcul de plan impossible, elle envoie un rapport d'échec à la fonction de détermination du contexte de planification. Cette fonction est activée par la fin d'un calcul de contexte de planification/replanification.

Le séquenceur niveau 2/1

Le séquenceur a pour entrée le plan courant, issu de la base de données du niveau 2. Son rôle est de séquencer le plan : lorsqu'un segment de plan touche à sa fin, il provoque le calcul de trajectoire pour le segment suivant et met à jour la base de données du niveau 2.

3.4.3 Niveau 1 : gestion de trajectoire

Le niveau 1 effectue la gestion des trajectoires : il vérifie la conformité de la trajectoire courante avec une prévision et met en œuvre réalisation de chaque segment de plan. Il possède une base de données propre contenant la liste courante des points de trajectoire et le point de trajectoire traité courant. Les figures 3.19 et 3.20 présentent ce niveau. Il a pour entrées les cartes des dangers et des objectifs courantes de la base de données générale, l'état des ressources, la position géographique du véhicule et l'état de la charge utile issus des capteurs et la liste courante des points de trajectoire issue de la base de données du niveau 1. En sortie, il met à jour la liste des points de trajectoire et les actions datées sur la charge utile dans sa base de données. Le niveau est activé soit par une boucle de contrôle interne, soit par le niveau supérieur (demande de calcul de trajectoire), soit par le niveau inférieur (point inatteignable). Le niveau 1 active le niveau 2 par un événement de segment infaisable. Entre le niveau 1 et le niveau 0, on insère un séquenceur, dont le rôle est de gérer l'enchaînement des points de trajectoire et la mise à jour de la base de données du niveau 1. Il a pour entrée la liste des points de trajectoire pour effectuer un segment et provoque des calculs de consignes effectués par le niveau 0. Le niveau 1 est constitué de trois fonctions.

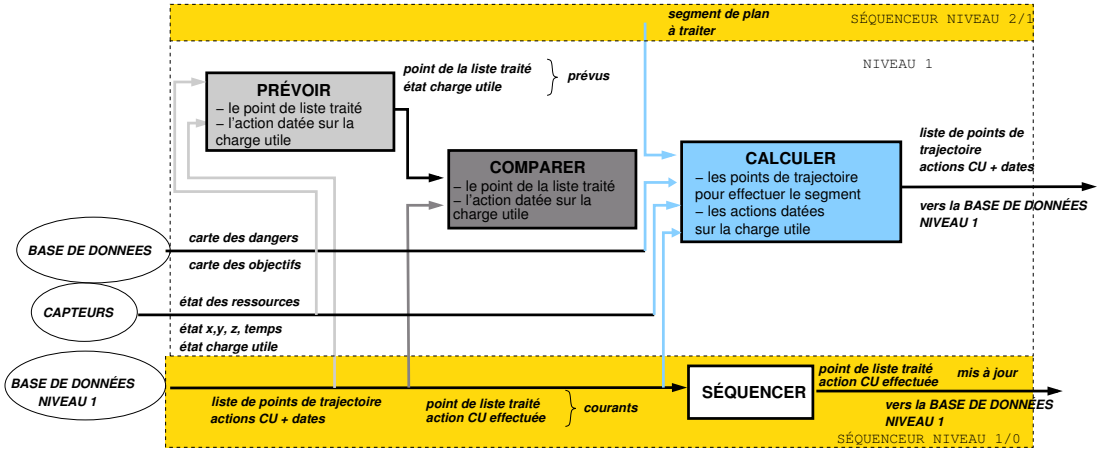


FIG. 3.19 – Niveau 1 : description des échanges de données

Prévoir

Cette fonction a pour entrées l'état des ressources, l'état (x, y, z, temps) du système de drone et l'état de la charge utile fournis par les capteurs d'une part, la liste courante des points de trajectoire et les actions datées sur la charge utile fournies par la base de données de niveau 1 d'autre part. Elle prévoit le point de la liste traité et l'état de la charge utile en fonction de la date courante.
Elle est activée régulièrement par une boucle de contrôle automatique gérée par l'architecture. Pour l'exemple applicatif de cette thèse, la boucle automatique devrait être activée toutes les secondes environ. La fin de la prévision provoque une comparaison.

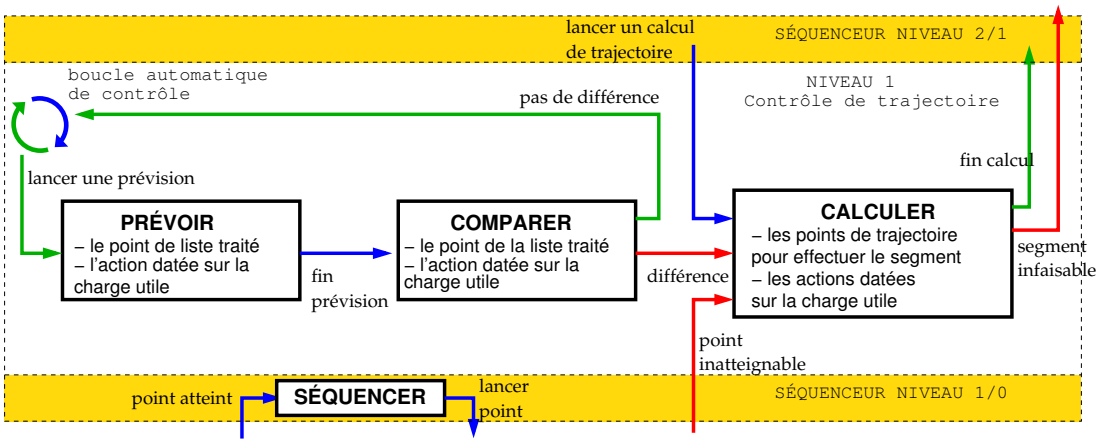


FIG. 3.20 – Niveau 1 : description des activations de tâches

Comparer

Cette fonction a pour entrées le point de la liste traité prévu et l'état prévu de la charge utile d'une part, le point de la liste traité courant reçu du séquenceur et l'état courant de la charge utile reçu des capteurs d'autre part. Cette fonction compare les données prévues et les données courantes.
La fonction est activée par la fin de la fonction "Prévoir", elle envoie un ordre d'activation de tâche en cas de différence entre les données prévues et les données courantes. Les aléas détectés à ce niveau vont être principalement des pannes de la charge utile ou des problèmes concernant le suivi de trajectoire. Les problèmes doivent être identifiés de façon à permettre un changement de contexte de planification.

Calculer

Cette fonction a pour entrées le segment de plan à traiter fourni par le séquenceur 2/1, l'état courant des cartes de danger et des objectifs issues de la base de données générale, l'état courant des ressources, du système et de la charge utile fournis par les capteurs, le point de la liste traité et l'action datée sur la charge utile fournies par la base de données du niveau 1. La fonction calcule une liste de points de trajectoire et des activations datées de la charge utile en fonction de l'action à effectuer sur le segment donné.

Ainsi pour l'exemple applicatif :
- aller-tout-droit : la fonction utilise directement le segment et le divise régulièrement afin d'obtenir une liste ordonnée de points de trajectoire. Les points de trajectoire sont des points de navigation donnés par leur coordonnées (x, y, z), le cap à suivre par le véhicule et la date à laquelle ils doivent être atteint. Il n'y a pas d'action sur la charge utile, sauf si le segment à pour point de passage de fin un point de transmission : dans ce cas, à la date de fin du segment, la transmission des données doit être activée.
- contourner : la fonction détermine la trajectoire pour contourner la zone de danger,

connue par la carte des dangers. Elle divise ensuite cette trajectoire de façon à obtenir une liste ordonnée de points de trajectoire. Il n'y a pas d'action sur la charge utile, sauf si le segment à pour point de passage de fin un point de transmission : dans ce cas, à la date de fin du segment, la transmission des données doit être activée.

- observer-une-zone : la fonction calcule les coordonnées des points afin d'effectuer un balayage en lignes parallèles décrit dans la section 1.5.4. Elle calcule également le cap à suivre pour chaque point et la date à laquelle ils doivent être atteints. Les points sont rangés par date croissante dans la liste des points de trajectoire. A la date de début du segment, un capteur permettant l'observation de la zone doit être activé. A la date de fin du segment, ce même capteur doit être désactivé. Si la transmission sur la zone est effectuée en ligne (voir la section 1.5.1 pour un rappel de cette notion), la transmission de données doit être activée en début de segment et désactivée en fin de segment. Si la transmission s'effectue en fin d'observation, la transmission doit être lancée en fin de segment. Sinon, la transmission n'est pas effectuée et l'information est conservée.

La fonction de calcul est activée soit par le niveau supérieur (événement "lancer un calcul de trajectoire"), soit en cas de détection de différence par la fonction "Comparer", soit par le niveau inférieur en cas de point inatteignable.

Le séquenceur niveau 1/0

Le séquenceur a pour entrées la liste de points de trajectoire et les actions datées sur la charge utile courants, issues de la base de données du niveau 1. Il séquence la liste des points de trajectoire et envoie les actions prévues pour la charge utile. Lorsqu'un point de la trajectoire a été atteint, il provoque un calcul de consigne pour atteindre le point suivant et met à jour la base de données du niveau 1.

3.4.4 Niveau 0 : guidage

Le niveau 0 effectue le guidage : il vérifie la conformité des consignes avec une prévision et met en œuvre le suivi de consignes et les activations de la charge utile. Il possède une base de données propre contenant les consignes courantes. Les figures 3.21 et 3.22 présentent ce niveau. Le niveau a pour entrées l'état (x, y, z) et le temps, ainsi que l'état de la charge utile donnés par les capteurs et les consignes courantes issues de la base de données dédiée au niveau. En sortie, il envoie des consignes bas niveau à la couche physique de type position/vitesse ou position/cap, et des activations/désactivations de la charge utile. Par exemple pour un drone, on peut assimiler la couche physique de déplacement à l'autopilote. Typiquement, la commande est alors une vitesse, un cap et une altitude de consigne. Il met aussi à jour les consignes. Le niveau est activé soit par une boucle de contrôle interne, soit par le niveau supérieur (demande de calcul de consignes). Le niveau 0 active le niveau 1 par un événement de point inatteignable. Le niveau de guidage est constitué de trois fonctions.

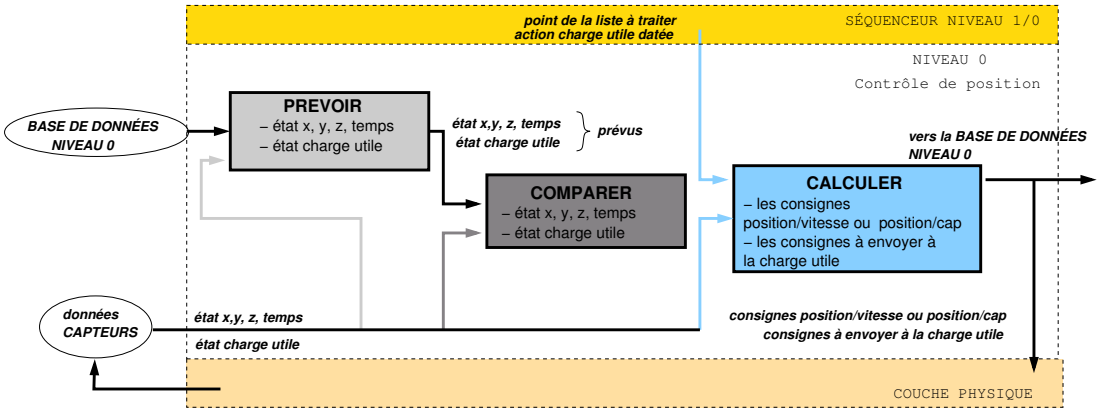


FIG. 3.21 – Niveau 0 : description des échanges de données

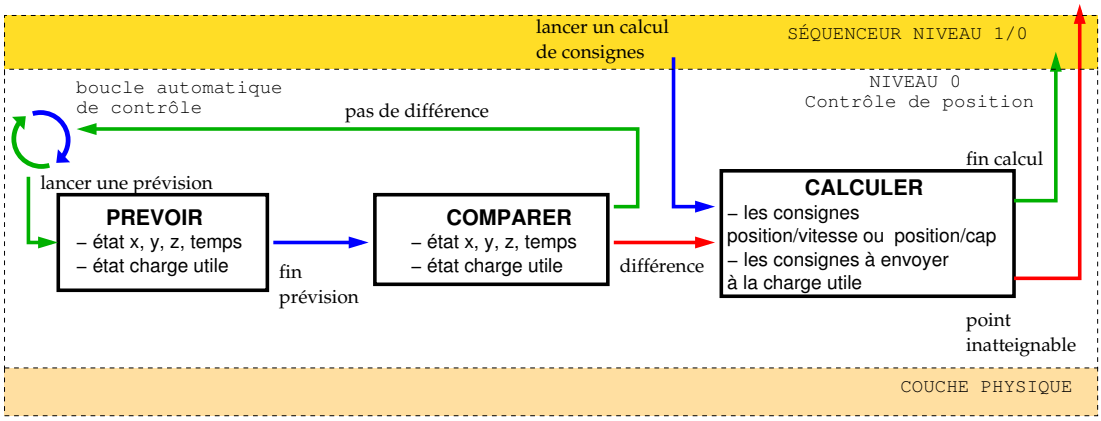


FIG. 3.22 – Niveau 0 : description des activations de tâches

Prévoir

Cette fonction a pour entrées l'état (x, y, z, temps) du système de drone et l'état de la charge utile donnés par les capteurs d'une part, les consignes courantes contenues dans la base de donnés du niveau 0 d'autre part. Elle prévoit l'état du système et de la charge utile à court terme.
Elle est activée régulièrement par une boucle de contrôle automatique gérée par l'architecture. Pour l'exemple applicatif de cette thèse, la boucle automatique devrait être activée toutes les millisecondes environ. La fin de la prévision provoque une comparaison.

Comparer

Cette fonction a pour entrées les états du système et de la charge utile prévus d'une part, l'état du système et l'état de la charge utile courants reçus des capteurs d'autre part. Cette fonction compare les données prévues et les données courantes.
La fonction est activée par la fin de la fonction "Prévoir", elle envoie un ordre d'activation

de tâche en cas de différence entre les données prévues et les données courantes. Les aléas détectés à ce niveau concernent des décalages de l'état dus au vent, aux dangers ou à des pannes de la charge utile.

Calculer

Cette fonction a pour entrées le point de la trajectoire à atteindre, les ordres à envoyer à la charge utile issus de la base de données d'une part et l'état courant du système et de la charge utile issus des capteurs d'autre part. La fonction calcule les consignes à envoyer à la couche physique de façon à atteindre le point voulu et à activer ou déactiver la charge utile correctement.

Les consignes envoyées pour l'exemple applicatif sont des consignes en position/cap ou position/vitesse et des consignes d'activation ou désactivation de la charge utile.

La fonction de calcul est activée soit par le niveau supérieur (événement "lancer un calcul de consignes"), soit en cas de détection de différence par la fonction "Comparer".

3.5 Utiliser ProCoSA pour coder et activer l'architecture

On propose de coder les différents niveaux de l'architecture avec Edipet et de gérer toutes les activations de tâches grâce au Joueur de Petri de ProCoSA. Les blocs fonctionnels correspondront à des processus externes activés par requêtes. Muni de cette partie de l'architecture embarquée, le système de drone est capable d'organiser ses tâches de raisonnement. En parallèle, il faut développer une organisation capable de superviser les tâches physiques du système.

L'association du contrôle des tâches de raisonnement et du suivi des tâches physiques constitue une architecture embarquée qui donne au système une autonomie décisionnelle et fonctionnelle de haut niveau.

3.5.1 L'organisation des tâches de raisonnement

Les bases de données

Les bases de données dont se sert l'architecture sont au nombre de six.

1. La base de données générale contient les cartes des dangers et des objectifs. Elle est mise à jour par le niveau le plus haut de la hiérarchie.
2. La base de données capteurs contient les renseignements donnés par les capteurs sur les cartes des dangers et des objectifs : elle détecte des apparitions/-disparitions/mouvements des zones de danger ou des zones objectifs. Elle contient aussi tous les renseignements sur l'état courant des ressources, sur la position de l'engin, sur la date courante et l'état de la charge utile. Elle est mise à jour automatiquement.
3. La base de données opérateur contient les renseignements donnés par le ou les opérateurs sur les cartes des dangers et des objectifs. Le ou les opérateurs peuvent

changer cette base de données à n'importe quel moment de la mission. L'avantage de cette base de données est que les connaissances opérateurs peuvent être directement prises en compte par l'architecture. L'opérateur peut ainsi "guider" la recherche, en donnant par exemple plus ou moins d'importance aux objectifs qu'il définit.

4. La base de données de niveau 2 contient le plan courant, l'évolution nominale des ressources au moment de la planification et le segment de plan traité courant. Elle est mise à jour au niveau 2 par un nouveau calcul de plan et par le séquenceur 2/1.
5. La base de données de niveau 1 contient la liste de points de trajectoire et les actions datées d'activation/désactivation de la charge utile, ainsi que le point de la liste traité et l'action courante effectuée sur la charge utile. Elle est mise à jour au niveau 1 par un calcul de point de trajectoire pour effectuer un segment de plan et d'actions datées sur la charge utile. Elle est mise à jour par le séquenceur 1/0.
6. La base de données du niveau 0 contient les consignes en position/vitesse ou position/cap et les consignes envoyées à la charge utile. Elle est mise à jour par un calcul du niveau 0.

Les tâches de raisonnement font des accès en lecture ou en écriture à cas bases de données.

Les activations de tâches de raisonnement

L'activation des tâches de raisonnement est provoquée par des requêtes d'activation associées à des transitions de réseaux de Petri. On propose de coder chaque niveau par un réseau de Petri. Les figures 3.23, 3.24, 3.25, 3.26, 3.27, et 3.28 donnent les réseaux de Petri correspondants. Cette solution a l'avantage de dissocier chaque niveau et est de ce fait très lisible. On aurait très bien pu développer un réseau unique. Ainsi, la transition "lancer-une-replanification" se retrouve dans le niveau 3 et le niveau 2, de même que l'événement "planification-impossible". On peut remarquer de la même façon des transitions identiques entre chaque niveau consécutif et les séquenceurs.
Dans ces réseaux de Petri, les places modélisent soit l'activation d'une fonction de calcul, soit l'activation d'un réseau. Ainsi la place NIVEAU1 :SEGMENT du séquenceur 2/1 décrit l'activation du réseau NIVEAU1 par la transition lancer-trajectoire. Les boucles de contrôle présentées dans la section précédente sont des activations périodiques sur les transitions de certains réseaux, envoyées par des processus externes.

Décrivons plus en détail le fonctionnement de ces réseaux. Les niveaux 3 et 2 (Figures 3.23 et 3.24) sont le reflet des descriptions d'activations de tâches décrites dans la section précédente. Les séquenceurs 2/1 et 1/0 (Figures 3.25 et 3.27) fonctionnent exactement comme on l'a décrit précédemment. Pour le niveau 1 (Figure 3.26), l'idée est que pour réaliser une trajectoire, il faut en avoir calculé les paramètres au préalable. Pour la première trajectoire du plan, le calcul de la trajectoire puis son application sont effectués en séquence. Pour les autres trajectoires, il est plus judicieux de profiter de la réalisation du segment courant pour effectuer les calculs des paramètres du segment suivant. La fin de la trajectoire dépend alors de la fin du calcul des paramètres et de

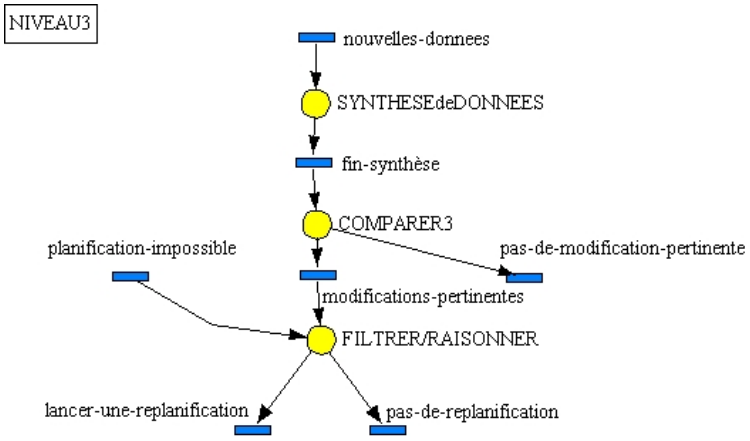


FIG. 3.23 – Réseau de Petri pour le niveau 3

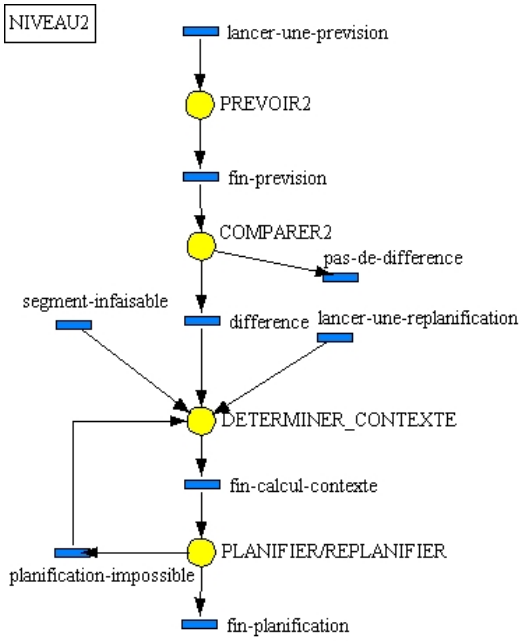


FIG. 3.24 – Réseau de Petri pour le niveau 2

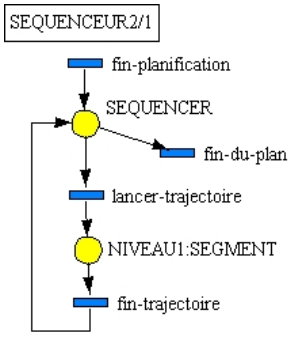


FIG. 3.25 – Réseau de Petri pour le séquenceur 2/1

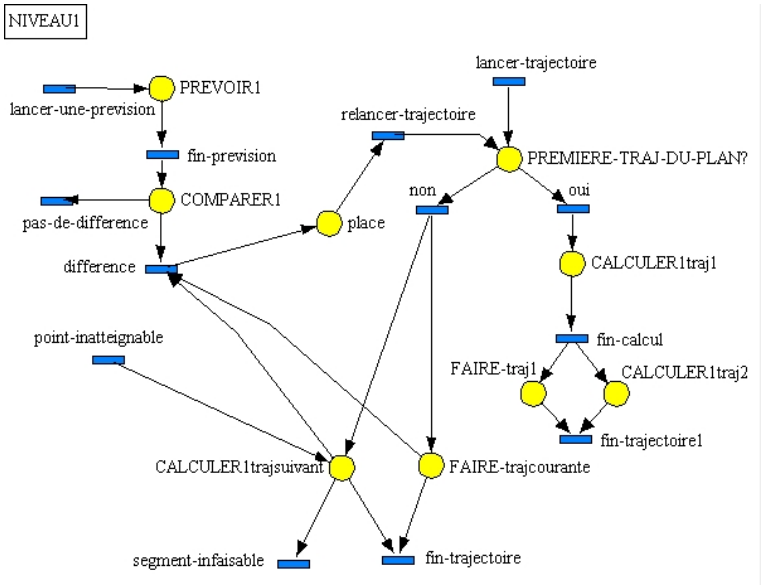


FIG. 3.26 – Réseau de Petri pour le niveau 1

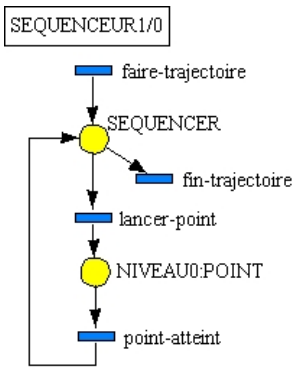


FIG. 3.27 – Réseau de Petri pour le séquenceur 1/0

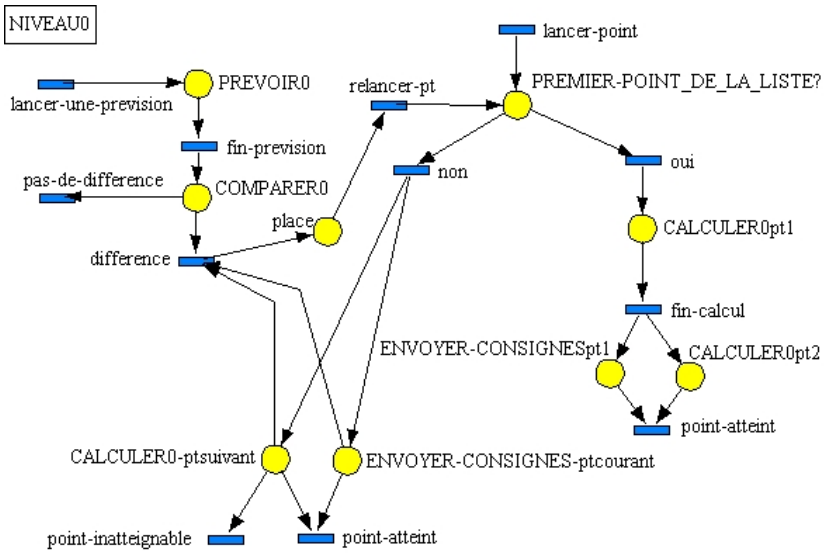


FIG. 3.28 – Réseau de Petri pour le niveau 0

la fin de la réalisation de la trajectoire courante. Si la boucle de contrôle automatique détecte une non-conformité, les places de calculs et de réalisation sont démarquées, les processus de calculs sont ainsi interrompus. Le calcul d'une trajectoire est alors relancé. Les mêmes idées sont appliquées pour la réalisation d'un point (Figure 3.28).

Mise en œuvre des fonctions

L'activation d'une fonction par une requête associée à un événement ne suffit pas : il faut aussi que la fonction ait accès aux données dont elle a besoin pour son calcul.

Il existe deux types de données :

- les données connues par les bases de données ; les fonctions qui en ont besoin font une demande à la base de données concernée : c'est par exemple le cas des cartes de dangers et d'objectifs.
- les données produites par une fonction et envoyées directement à une autre fonction. Ces données transitent par le biais des transitions des réseaux de Petri sous la forme d'informations : par exemple les données résultant d'une fonction "Prévoir".

La mise en œuvre d'une fonction sera donc la suivante : la fonction est activée par une requête d'activation associée à une transition. Cette requête est éventuellement associée à des informations résultant par exemple d'un calcul antérieur. Les informations sont les arguments d'entrée de la fonction. La fonction informatique a comme préliminaire une demande d'arguments aux bases de données existantes. Elle peut ensuite effectuer son calcul. La sortie de la fonction peut mettre à jour une ou plusieurs bases de données et envoyer un événement muni éventuellement d'informations qui vont servir d'arguments à une fonction suivante.

Exemples

Pour l'exemple applicatif, détaillons le traitement d'un aléa de type "manque de carburant". L'aléa est détecté dans la boucle automatique de contrôle du niveau 2 : le niveau des ressources prévu par la planification est différent du niveau des ressources mesuré. La transition "différence" du réseau de Petri de la figure 3.24 est donc tirée. L'observation de cette différence déclenche la détermination d'un contexte de replanification, puis une replanification. Si le calcul d'un plan est possible, le résultat est envoyé au séquenceur 2/1 par le biais de la transition "fin-planification" des réseaux des figures 3.24 et 3.25. Le rôle du séquenceur est d'appliquer le plan segment par segment en activant le réseau de Petri de niveau 1. L'aléa est donc traité. Si le calcul d'un plan est impossible, un bouclage (transition "planification-impossible" de la figure 3.24) est effectué sur la fonction de détermination du contexte jusqu'à ce qu'un calcul de plan soit possible.

Détaillons à présent le traitement d'un aléa de type "changement d'objectifs". Ce type d'aléa active la transition "nouvelles-données" du réseau de Petri de la figure 3.23. Les modifications de la carte des objectifs sont synthétisées par la synthèse de données et envoyées par la transition "fin-synthèse". La fonction "Comparer" du niveau 3 est alors activée. Si les modifications sont pertinentes et que les changements nécessitent une replanification, la transition "lancer-une-replanification" va être tirée. Elle active alors le réseau de Petri de niveau 2 : un contexte de replanification est déterminé et une replanification est lancée. La suite des activations s'effectue comme précédemment : si le calcul d'un plan est possible, le résultat est envoyé au séquenceur 2/1 par le biais de la transition "fin-planification". L'aléa est donc traité. Si le calcul d'un plan est impossible, un bouclage est effectué sur la fonction de détermination du contexte jusqu'à ce qu'un calcul de plan soit possible.

Ces exemples montrent que les aléas sont traités par l'architecture de manière hiérarchique. Un aléa détecté au niveau 3 peut déclencher des calculs successifs dans tous les niveaux inférieurs. De même, certains aléas de type "calcul-impossible" peuvent remonter et déclencher des calculs sur les niveaux supérieurs ; par exemple un événement de type "segment-infaisable" déclenché par le calcul au niveau 1 (Figure 3.26) remonte vers le niveau 2 (Figure 3.24).

3.5.2 Le suivi des tâches physiques

L'objectif de l'architecture est aussi de superviser les tâches physiques accomplies pendant la mission. Dans les réseaux de Petri qui décrivent le comportement du véhicule pendant la mission, les places modélisent les actions possibles du véhicule pendant la mission et les transitions le changement entre les états : à tout moment, l'ensemble des places marquées décrit l'état du véhicule. Quand il existe un lien de communication entre le système embarqué et un opérateur, ce dernier peut visualiser en temps-réel la mise à jour du marquage. Cela permet à l'opérateur de vérifier que le véhicule suit le comportement spécifié.

Les informations que l'on souhaite avoir à ce niveau sont :

- le type de nœud que le drone est en train de rallier ;
- l'action effectuée parmi : voler tout droit, contourner une zone de danger, effectuer un balayage ;
- des informations sur les aléas observés.

Les réseaux de Petri développés pour contrôler l'évolution des tâches physiques n'activent aucun calcul. Ils sont le reflet de l'état du système. Les événements qui déclenchent le tir d'une transition source sont des doublons des événements présentés dans les réseaux de Petri du contrôle des activations des tâches de calcul. La figure 3.29 illustre le suivi des tâches physiques de la mission. Le réseau du dessus décrit le type de nœud vers lequel se dirige le système. Le réseau du dessous décrit le type d'action effectuée entre deux nœuds. La figure 3.30 illustre le mécanisme de suivi des aléas. Le but est de pouvoir détecter jusqu'à quel niveau se propagent les défaillances par exemple, ou quelles sont les causes d'un échec.

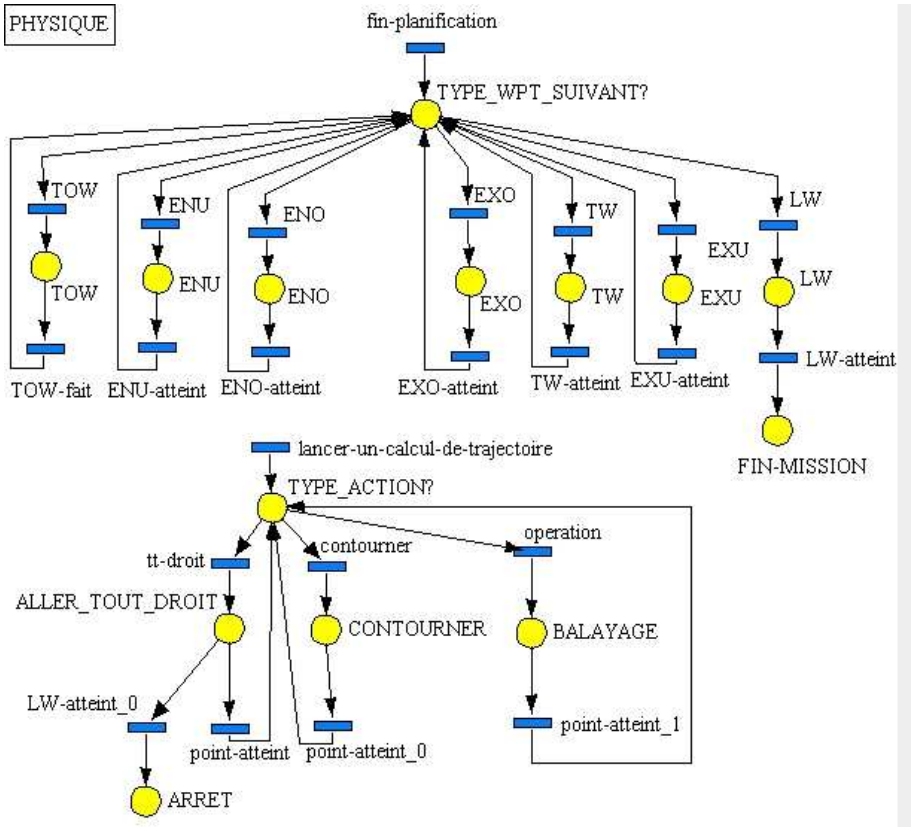


FIG. 3.29 – Réseau de Petri de supervision des tâches physiques

3.5.3 Conclusions

ProCoSA permet donc de spécifier et contrôler un comportement autonome. Vérifions que les fonctionnalités requises pour l'architecture sont bien respectées.

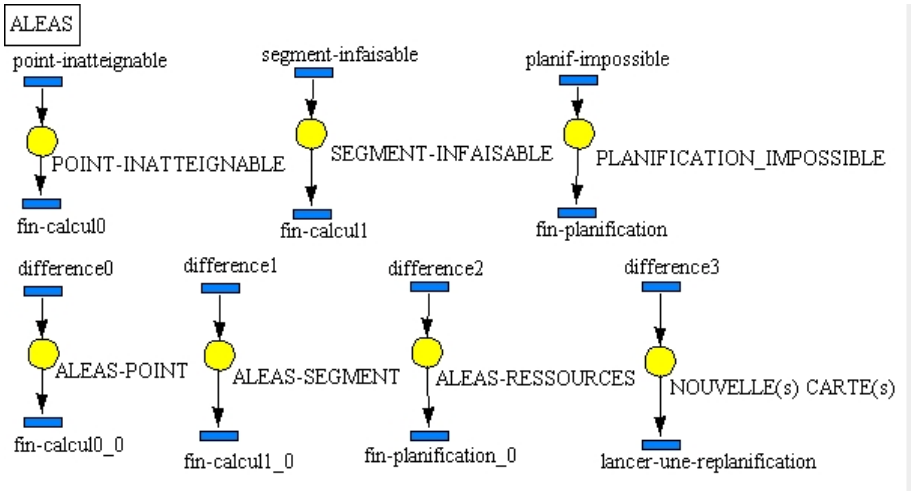


FIG. 3.30 – Réseau de Petri du suivi des aléas

1. Le niveau 3 de l'architecture permet de prendre en compte les données opérateur et les données des capteurs. Les données des capteurs sont en outre utilisées grâce à l'intégration des bases de données dans les fonctions.
2. Les niveaux 1 et 0, associés aux séquenceurs, mettent en œuvre l'enchaînement des segments définis par la planification.
3. Le niveau 0 permet d'effectuer les calculs pour rallier les points de passage et trouver les dates adéquates pour activer et désactiver la charge utile.
4. Les consignes de bas niveau sont prises en charge par le niveau 0.
5. Le suivi du déroulement des phases de calculs s'effectue au niveau des réseaux de Petri. Ils spécifient l'enchaînement des calculs.

L'architecture permet donc au système d'effectuer sa mission. Concernant la réaction aux aléas, on peut constater que :

1. Les modifications sur la carte des dangers ou sur la carte de la mission sont prises en compte au niveau le plus haut de l'architecture. Ces modifications enclenchent un calcul de plan.
2. Les impossibilités de calcul à chaque niveau sont reportées et traitées sur le niveau supérieur.
3. Les boucles de contrôle automatiques sur chaque niveau permettent de détecter une non-conformité entre un résultat prévu et un résultat courant.

L'architecture sous ProCoSA a donc toutes les fonctionnalités requises pour donner au système une autonomie décisionnelle.

L'architecture proposée n'est cependant pas spécifique à ProCoSA. Il semble possible de ré-exprimer le fonctionnement d'une telle architecture dans des langages tels que RAP ou ESL (voir section 3.2.3). Néanmoins, il faut souligner que ces langages sont habituellement utilisés pour coder un niveau de sélection de comportements réactifs.

Dans le cadre d'une réécriture de l'architecture proposée, il faudra donc coder non pas des procédures pour réaliser un but, mais des procédures pour activer des fonctions de calcul.

3.6 Criticité temporelle et critères pour le calcul de replanification

Lorsque l'architecture embarquée détecte un nouveau contexte de mission, quelles vont être les conséquences de la mise en œuvre du plan courant dans le nouveau contexte ? Faut-il replanifier et dans quelles conditions ? La solution envisagée est d'évaluer le plan courant dans le nouveau contexte et selon la valeur de cette évaluation, de juger la criticité temporelle du calcul de replanification.

3.6.1 Evaluation du plan courant dans un nouveau contexte

Pour rappel (Définition 1), le contexte de la mission est l'ensemble comprenant la carte des objectifs, l'état de la carte de l'environnement, le nœud vers lequel le véhicule est en train de se diriger, la date, l'état des ressources et l'état de réalisation des objectifs de la mission.

Soit C et C' respectivement l'ancien et le nouveau contexte de la mission. Les variables Q et Δ sont choisies de façon à optimiser J pour le contexte C.

$$(\hat{Q}, \hat{\Delta}) = \text{argmin } J(C, Q, \Delta)$$

L'évaluation du critère dans le nouveau contexte sera de la forme

$$J(C', \hat{Q}, \hat{\Delta})$$

On note cette évaluation $J_{C'}$.

La complexité du processus d'évaluation vient du fait qu'il faut considérer :
- pour le haut niveau, les objectifs réalisés, les objectifs modifiés, les nouveaux objectifs et les objectifs enlevés ;
- pour le bas niveau, les nœuds déjà traités, les nœuds dont les coordonnées ont été modifiées, les nouvelles contraintes temporelles, les nouveaux nœuds et les nœuds enlevés ;
- pour l'environnement, les déplacements de menaces, les nouvelles menaces, les menaces qui ont disparu.

L'évaluation du plan courant dans le nouveau contexte s'effectue selon l'algorithme 9.

La première étape consiste à lire la nouvelle mission (**Ligne 1**). $NouvelleMission$ contient les cartes des objectifs et des dangers mises à jour. Elle contient les renseignements sur les contraintes temporelles associées aux nœuds et toutes les données courantes concernant les objectifs et les menaces. La lecture du nom du point de passage suivant courant dans le plan (**Ligne 3**) a pour but de rechercher dans le plan le

Algorithme 9 : Évaluation du plan courant dans le nouveau contexte

```
début
    Lire NouvelleMission
    Lire PointCourant
    Lire PointDePassageSuivantCourant
    répéter
        PointLu ← lire nom du point suivant dans le plan
    jusqu'à PointLu = PointDePassageSuivantCourant
    répéter
        PointLu ← lire nom du point suivant dans le plan
        si PointLu ∈ NouvelleMission alors
            si PointLu est un point de contournement alors
                Créer le nœud de contournement correspondant à PointLu
            sinon
                Créer le nœud correspondant à PointLu
        sinon
            Chercher PointLu dans AncienneMission
            si Pointlu est une sortie d'objectif alors
                Créer le nœud correspondant à PointLu en associant un gain nul
            sinon
                Créer le nœud correspondant à PointLu
    jusqu'à PointLu = n_e
    Créer l'itinéraire de PointCourant à n_e
    Optimiser les durées et donner la valeur du critère J_C'
fin
```

point vers lequel le système se dirige au moment de la replanification (**Ligne 4**). Une fois ce point trouvé, tous les points du plan sont traités de manière itérative (**Ligne 5-13**). Si le point est un point de la nouvelle mission (**Ligne 8**), le nœud correspondant est créé, selon que le point est atteint en contournement ou pas (**Ligne 9-10**). Sinon, on crée le nœud correspondant. Si c'est un nœud de sortie d'objectif, un gain nul est associé à cet objectif, puisque l'objectif n'existe plus (**Ligne 11-13**). Enfin, l'itinéraire complet du nœud courant à un nœud fin est créé (**Ligne 15**). Cet itinéraire est optimisé, en prenant en compte les nouvelles contraintes temporelles et le nouvel emplacement des menaces (**Ligne 16**).

On peut noter les choix de traitements :
- si un objectif qui appartient à Q n'est plus dans la nouvelle mission, il est gardé dans le plan mais il ne rapporte plus de gain ;
- si un nœud de l'itinéraire évalué s'est déplacé, l'évaluation considère ses nouvelles coordonnées ;
- si un nœud de l'itinéraire évalué a disparu, il est conservé dans le plan ;
- les nouvelles contraintes temporelles sont prises en compte dans l'évaluation du critère pour le nouveau plan ;
- l'évaluation du critère prend en compte les modifications de l'environnement ;
- les durées sont ré-optimisées suivant le critère, ce qui a pour conséquence de ré-optimiser la consommation des ressources.

3.6.2 Classification des problèmes de replanification

La valeur $J_{C'}$ va être déterminante pour qualifier la replanification. En effet, si $J_{C'}$ est infiniment grand, ce la signifie que le plan courant est inadmissible pour la nouvelle mission : soit les contraintes temporelles sont violées, soit le véhicule épuise toutes ses ressources. A l'inverse, si $J_{C'}$ est négatif, cela signifie que le plan courant permet d'effectuer une mission où les récompenses sont supérieures aux coûts, et qu'elle vérifie toutes les contraintes de la mission. Le seul intérêt de la replanification est d'améliorer la fonction critère. Ainsi, plus la valeur de $J_{C'}$ est élevée, plus la mission a de risques de conduire à un échec. Soit J_{MAX} une valeur de $\mathbb{R}$ telle que le risque d'échec de la mission est considéré comme trop élevé.

La classification suivante est proposée :
- **Cas 1** : si $J_{C'} > J_{MAX}$, le plan précédent mène à l'échec de la mission. Les contraintes temporelles ou les contraintes sur les ressources ne sont plus respectées. La replanification est **critique**, elle doit absolument donner lieu à un plan admissible à exécuter.
- **Cas 2** : si $0 < J_{C'} < J_{MAX}$, le plan courant est mauvais, mais que par ailleurs une durée de replanification moyennement rapide est acceptée. La replanification est **nécessaire**, car le plan courant a un risque d'échec ou ne semble pas optimal.
- **Cas 3** : si $J_{C'} < 0$ alors le plan courant ne comporte pas de risque de destruction. La replanification est **éventuelle**. Le seul avantage d'un calcul de replanification vient du fait que le critère pourrait être amélioré.

3.6.3 Critères d'évaluation pertinents selon la classe du problème

Après avoir classifié la criticité du calcul de replanification selon la valeur de $J_{C'}$, l'idée est à présent de dégager les critères pertinents pour évaluer un calcul de planification selon les domaines de criticité. Les critères sont dépendants de paramètres liés à l'application.

Cas 1

Si $J_{C'} > J_{MAX}$, il est crucial de replanifier au plus vite. **La durée de replanification va être critique**.
Notons δ_r la durée de calcul au-delà de laquelle le module de planification doit fournir un plan. S'il ne le peut pas, il envoie un rapport d'échec vers la fonction de détermination du contexte. Sinon, dès que le module de planification a calculé un plan admissible, il est envoyé au séquenceur. Une durée δ_1 est introduite pour trouver un meilleur plan. Les critères pour évaluer les méthodes de planification seront donc le durée de calcul et la qualité de la première solution admissible, et la durée de calcul et la qualité de la meilleure solution en une durée inférieure à δ_1.

Cas 1 : replanification critique
- Une solution admissible est exigée en une durée inférieure à δ_r, sinon la planification est jugée impossible.
- Si une solution est trouvée en une durée inférieure à δ_r, elle est appliquée, et le module de planification essaye d'améliorer la solution courante en une durée de calcul inférieure à δ_1.

Les critères d'évaluation sont la durée de calcul et la qualité de la première solution admissible, et la durée de calcul et la qualité de la meilleure solution en une durée inférieure à δ_1.

Cas 2

Si $0 < J_{C'} < J_{MAX}$**, la durée du calcul de replanification peut être relâchée.**
L'idée est de ne prendre en compte que les plans qui améliorent le critère de manière significative. Ainsi, soient deux valeurs de critères J^1 et J^2, trois relations sont définies afin de les comparer : soit J^1 "est préféré" à J^2, soit J^2 "est préféré à" J^2, soit J^1 et J^2 sont quasiment identiques, ils sont dits "indifférents". De manière formelle, une fonction seuil $S(J^1) = J^1 \times k + \beta$ est introduite.

DÉFINITION 14 *Si* $|J^1 - J^2| < |S(J^1)|$ *alors* J^1 *et* J^2 *sont considérés comme* ***indifférents****. On note* $J^1 \mathcal{I} J^2$.

DÉFINITION 15 *Pour des critères au sens de minimum, si* $J^2 < J^1 - |S(J^1)|$ *alors* J^2 ***est préféré à*** J^1*. On note* $J^2 \mathcal{P} J^1$.

Supposons alors que J^1 et J^2 soient les critères associés à deux plans proposés l'un après l'autre par la planification. Tant que $J^1 \mathcal{I} J^2$, la solution de critère J^2 proposée par le module de planification n'est pas prise en compte.

Dans le cas présent, tant que le critère J^0 trouvé par la planification et $J_{C'}$ sont indifférents, le plan associé à J_0 n'est pas pris en compte. Quand $J^0 \mathcal{P} J_{C'}$ alors le plan associé à solution J^0 trouvée à l'instant t^0 est sauvegardé. Ce plan est pris en compte. Comme pour le cas 1, une durée δ_2 est introduite pour trouver un meilleur plan, cependant, seules les solutions améliorant le plan de manière significative seront prises en compte.

DÉFINITION 16 *Le critère est amélioré de* ***manière significative*** *si* $J^1 \mathcal{P} J^0$.

L'évaluation d'une méthode de planification se fera donc sur les critères suivants :
- la qualité et la durée de calcul de la première solution qui améliore le critère de façon significative (amélioration au-delà du seuil d'indifférence).
- qualité de la meilleure solution en une durée inférieure à δ_2 après l'obtention de la première solution acceptée.

Cas 2 : replanification nécessaire
- La solution courante doit être améliorée de manière significative pour que le calcul de replanification soit jugé efficace.
- Si une solution améliore de manière significative le critère courant, elle est appliquée, et le module de planification essaye d'améliorer la solution courante en une durée de calcul inférieure à δ_1.

Les critères d'évaluation sont la qualité et la durée de calcul de la première solution qui améliore le critère de façon significative et la qualité de la meilleure solution en une durée inférieure à δ_2 après l'obtention de la première solution acceptée.

Cas 3

Si $J_{C'} < 0$ alors la réalisation de la mission avec le plan courant est toujours bénéfique. La replanification peut durer tant qu'au moment de la sortie du nouveau plan, celui-ci garde un sens. On note δ_3 la durée de calcul au-delà de laquelle les calculs ne sont plus pertinents. L'évaluation d'une méthode de planification s'effectue alors sur la qualité de la meilleure solution en une durée inférieure à δ_3.

Cas 3 : replanification éventuelle
- La durée de calcul d'un plan n'est limitée que par la pertinence du résultat à l'instant où il est appliqué.
- Le module de planification essaye d'améliorer la solution courante en une durée de calcul inférieure à δ_3.

Le critère d'évaluation est la qualité de la meilleure solution en une durée inférieure à δ_3.

3.6.4 Application à l'exemple

Un contexte de mission est défini pour l'exemple par l'état et par la donnée de l'environnement :

- le nœud vers lequel le véhicule est en train de se diriger ;
- le temps t ;
- l'état des ressources ;
- l'état de réalisation des objectifs de la mission ;
- la carte des dangers.

L'évaluation du critère J se fait en connaissant la séquence Q et le vecteur des durées Δ ; à chaque durée est associée une action. Le critère est donc totalement déterminé quand le contexte de planification est connu.

La valeur de J_{MAX} est déterminée par le pourcentage de risque de destruction non compensé par des récompenses que l'opérateur est prêt à prendre si l'itinéraire courant est maintenu. Par exemple, un risque de r% de destruction est toléré, alors :

$$J_{MAX} = P_{sys} \times r$$

Cas 1

La durée δ_r doit être en accord avec la distance maximum que le drone peut parcourir sans avoir un nouveau plan. A une vitesse optimale de 60 $m.s^{-1}$, une durée acceptable sera par exemple d'une dizaine de secondes (environ 500 m).

Dans le cas critique, une fois que le module de planification a calculé un plan complet, le module de planification a une durée δ_1 pour améliorer la solution. Cette durée doit être raisonnable par rapport à la distance que parcourt le drone durant le calcul. A une vitesse optimale de 60 $m.s^{-1}$, une durée acceptable sera par exemple d'une vingtaine de secondes (environ 1 km).

Cas 2

Le seuil d'indifférence dépend des missions tests. Considérons la valeur moyenne du gain G_m sur une zone objectif, une première idée est de préférer un critère qui va prendre en compte la réalisation d'une zone de plus. On prendra donc par exemple comme valeur du seuil $S(J^1) = G_m \times 90\%$. Une seconde idée est d'accepter une amélioration du critère, par exemple de 10 %. Alors $S(J^1) = J^1 \times 10\%$. Cette solution est cependant inefficace lorsque J^1 est proche de 0.

La valeur δ_2 est choisie par l'utilisateur. On prendra par exemple 5 minutes. Une autre idée est de faire intervenir le quotient entre l'amélioration et la durée pour obtenir le nouveau plan (sorte de rapport qualité/prix). Cependant il faut garder dans l'idée que la valeur δ_2 est limitée.

Cas 3

Dans le cas de replanification éventuelle, le plan courant ne comporte pas de risque majeur d'échec. Une durée de calcul assez grande est donc autorisée. Soit δ_3 cette durée, on prendra par exemple 15 minutes.

3.6.5 Discussion

La solution envisagée ici constitue un choix déterminant dans l'évaluation des méthodes de planification. La criticité d'un contexte est donnée par la valeur du plan courant évalué dans le nouveau contexte.

D'autres solutions peuvent être envisagées. Une solution est d'évaluer la criticité en terme de durée restant avant les conséquences de l'événement : par exemple pour une mission d'observation en présence de dangers qui apparaissent en cours de mission, la durée entre la détection d'un nouveau danger sur le trajet du véhicule et la traversée de la zone de danger par le drone (Figure 3.31). Est-ce que l'événement déclenchant un calcul de replanification dénote par exemple une traversée du danger qui arrive rapidement dans le plan, ou le système a-t-il plus de temps pour replanifier ? Si le système a du temps pour replanifier, le calcul rapide d'un nouveau plan n'est pas nécessaire. Suivant l'événement, il faut alors détecter si le calcul du plan est contraint en temps ou non.

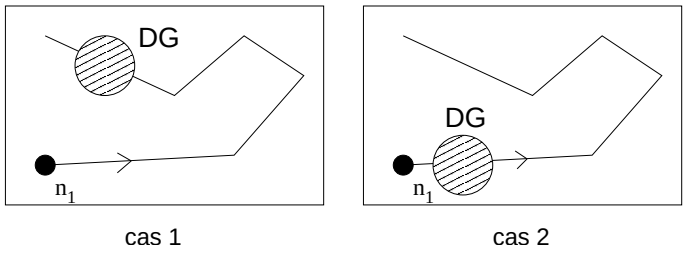


FIG. 3.31 – Deux cas de replanifications critiques - Cas 1 : le calcul de replanification pourrait durer- Cas 2 : le calcul de replanification est contraint par le temps

Une solution est de classifier les événements et de détecter ceux qui ont une conséquence immédiate sur la sécurité du vol. Néanmoins, il semble difficile d'effectuer une telle distinction.

La solution que nous avons détaillée dans les paragraphes précédents a l'avantage d'évaluer la criticité du plan courant. En cas de replanification critique, la solution doit être obtenue rapidement : quelle que soit la durée restante avant l'échec de la mission, un autre plan doit être calculé. Dans les autres cas, le plan courant n'implique pas un risque majeur pour l'appareil, de sorte que même si le système ne prend pas en compte les modifications qui ont déclenché une replanification, il y a peu de risque d'échec de la mission.

3.7 Conclusion

Dans ce chapitre, l'étude de la littérature sur les liens entre la planification et l'exécution nous a amené à cadrer ce travail de la manière suivante :

- La planification en ligne est déclenchée par des événements.
- L'architecture du système est une architecture hybride à plusieurs niveaux, dont un qui inclut le planificateur.
- ProCoSA est utilisé pour spécifier et mettre en œuvre l'architecture du système.

Une architecture hybride hiérarchique a ainsi été définie. Elle comprend quatre niveaux de la gestion de la mission jusqu'au guidage. Les mécanismes d'activation des fonctions ont été décrits grâce au logiciel ProCoSA. Cette architecture permet la mise en œuvre de la méthode de planification décrite précédemment en faisant apparaître une fonction de détermination du contexte de planification. L'étude de cette fonction montre qu'une analyse de la qualité du plan en cours vis-à-vis des données courantes permet de spécifier le temps laissé au planificateur pour donner un résultat.

Ces spécifications sont également des critères de performances permettant la comparaison des différentes méthodes de planification. Elles seront donc utilisées pour l'analyse des tests effectués sur des problèmes de planification.

4 Les tests

Résumé : Ce chapitre présente les tests effectués pour toutes les combinaisons des méthodes algorithmiques proposées. Les tests sont menés dans le cadre applicatif de missions militaires d'observation réalisées un véhicule aérien autonome. 36 scénarios sont ainsi joués. Les résultats sont analysés en tenant compte de la fréquence d'occurrence de chaque scénario. Une première analyse dégage pour chaque contexte de replanification une méthode ou un groupe de méthodes valable pour ce contexte. Une seconde analyse dégage la ou les méthodes embarquables et utilisables quel que soit le contexte de replanification.

4.1 Présentation

L'objectif de ce chapitre est d'utiliser les méthodes développées dans le chapitre 2 dans le cadre applicatif de missions d'observation menées par un véhicule aérien militaire. Ces véhicules autonomes sont appelés drones ou plus précisément "systèmes de drone". Un drone comporte un vecteur aérien ou engin, équipé d'une charge utile et de l'avionique. La charge utile est constituée par les capteurs embarqués utiles à la réalisation des objectifs de la mission (caméra, radar, désignateur laser...). L'avionique est l'informatique embarquée dont le rôle est d'assurer le pilotage du vecteur aérien et certains traitements d'informations. La station sol est composée des moyens au sol aidant à la réalisation de la mission, à la récupération et à certains traitements de l'information. L'opérateur communique avec le système via la station sol. Un drone comprend donc : un vecteur, une charge utile, une avionique et une ou plusieurs stations sol.

Parmi les nombreux types de drones, trois critères de classification sont essentiels : l'altitude de croisière, l'endurance et la longueur.

Les ordres de grandeur pour l'altitude de croisière h sont les suivants :

- moyenne altitude : 5000 m $<$ h $<$ 15000 m
- haute altitude : h $>$ 17000 m

L'endurance est définie comme la durée que peut passer l'engin en vol. En prenant $C = -\frac{dm}{dt}$ la réduction de masse m définissant la consommation de carburant exprimée en $kg.s^{-1}$, on peut définir l'endurance par E :

$$E = \int_{m_i}^{m_f} \frac{dm}{C}$$

avec m_i la masse initiale de l'avion et m_f la masse finale, c'est à dire à l'atterrissage. La différence entre m_i et m_f correspond à la masse de carburant. Un ordre de grandeur d'une endurance dite longue est de 10 à 50 heures.

Le domaine technique des drones se décompose en trois segments : les drones tactiques, les drones de moyenne altitude et longue endurance (MALE) permettant d'utiliser une charge utile de l'ordre de 100 kg et les drones de haute altitude et longue altitude (HALE).

Un drone de type MALE est choisi pour effectuer les missions test. Ce sont les engins les plus adaptés au contexte militaire de missions d'observation et de prises d'informations. Citons pour exemple quelques MALE existants de nos jours :

- le Predator (USA) (Figure 4.1), altitude de croisière 7600m, endurance 40h, mission de surveillance optique/radar, écoute électronique, utilisé en Irak, Bosnie, Kosovo et en Afghanistan ;
- le RQ-5 Hunter (USA), altitude de croisière 5000m, endurance 11h, applications terrestres et navales avec possibilité de transmettre les informations en quasi temps-réel, utilisé en Macédoine pour soutenir les opérations dans les Balkans en 1999, 2000, 2001, et 2002 ;
- les drones Eagle 1 et 2 (France), altitude de croisière 15240m, endurance 24h, missions de renseignement, de surveillance et de reconnaissance, premier vol en juin 2003 pour le prototype de Eagle 1.

FIG. 4.1 – Le Predator

L'annexe C propose une liste non exhaustive des principaux projets de recherche utilisant les drones. Ces projets sont souvent liés à des applications civiles.

4.2 Les scénarios

4.2.1 Le système de drone

Le système de drone choisi est de type MALE. La table 4.1 résume ses principales caractéristiques, choisies proches de celles du Predator.

altitude de croisière	5000 m
vitesse optimale	58,7 $m.s^{-1}$
masse à vide	1000 kg
masse de carburant (kg)	suivant la mission
Cl_{max}	2,85
Cd_o	0,13
poussée maximum au sol	3924 N
S	11,25
K	0,016
VNE	200 $m.s^{-1}$
CSR	2,77778e-6 $kg.s^{-1}.N^{-1}$
largeur des raies de balayage (m)	suivant la mission
P_{sys}	1000000 $
P_{carbu}	0,94 $\$.kg^{-1}$
P_{vol}	0,056 $\$.s^{-1}$

TAB. 4.1 – Caractéristiques du système de drone utilisé pour les tests

On suppose que les stations sol sont situées dans la zone amie, sans danger. L'ordre de grandeur de la portée des communications est de 140 km. Les communications sont supposées immédiates et sans échec.

4.2.2 Les missions et les environnements

Quatre missions dans quatre environnements différents sont définies de façon à tester un large panel de réactions du module de planification. Pour chaque mission, l'instant $t = 0$ correspond au décollage de l'engin.

La première mission comporte six zones objectif dont le rayon est de 10 km. La carte de la mission globale est donnée figure 4.2.

L'observation d'un objectif peut rapporter $G_o = 1000$ \$ au maximum. Les probabilités d'observation sont fixées à 1 pour toutes les zones. On définit deux entrées et deux sorties par zone objectif : elles sont réparties régulièrement sur le contour de la zone de manière à ce que les deux entrées soient diamétralement opposées (Figure 4.3), ainsi que les deux sorties : si le véhicule arrive droit devant une porte de sortie, la porte d'entrée la plus proche est au plus à 15 km. La transmission des informations s'effectue lors du passage par la porte de sortie des objectifs pour les zones $O1$ et $O2$. Pour les autres zones objectif, il est nécessaire de passer sur un des points de transmission pour envoyer l'information.

Deux modes de navigation sont envisagés (Section 1.5.3) : la ligne droite et le contournement de zone de danger. A ces deux modes de navigation s'ajoute l'action de balayage. En effet, lors du traitement d'une zone objectif, l'engin survole la zone en effectuant un balayage simple avec une largeur entre les raies de 1 km (Figure 4.3). La masse de carburant embarquée est de 400 kg.
L'environnement comporte quatre zones de danger de rayon égal à 40 km. Pour toutes

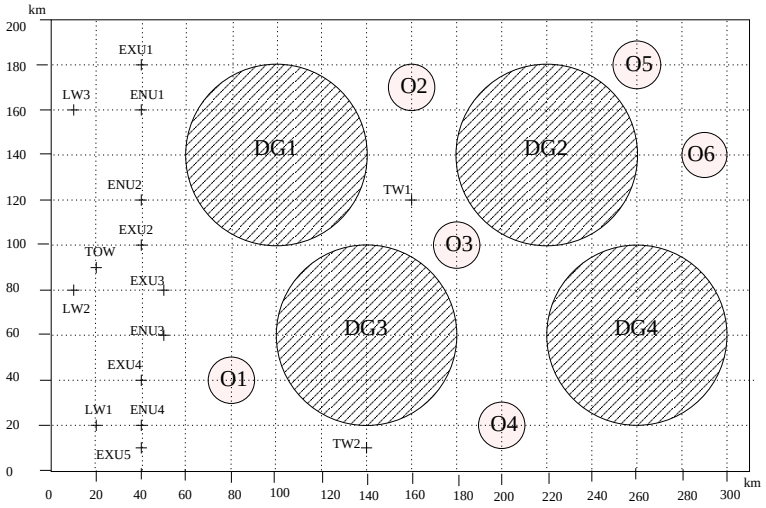


FIG. 4.2 – Mission 1

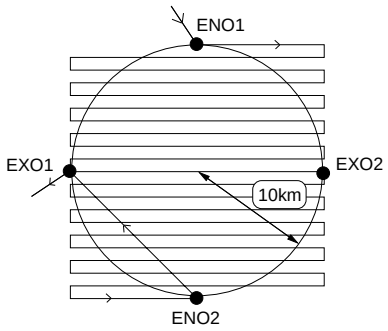


FIG. 4.3 – Disposition des entrées/sorties pour une zone objectif de rayon 10 km

les zones de danger, la probabilité d'existence est de 0,2. La probabilité de détection du drone par la menace est donnée par l'équation 1.17 avec $D_{min} = 15s$ et $D_{max} = 25s$. La probabilité de touche est la même pour toutes les menaces, elle est prise égale à 0,9.

La deuxième mission comporte quatre zones objectif, dont le rayon est de 10 km. La carte de la mission globale est donnée figure 4.4. L'observation des zones rapporte un gain différent selon l'importance de l'objectif. La zone O1 a un gain maximum de 500 $, les zones O2 et O4 de 1000 $ et la zone O3 a un gain de 2000 $. Les probabilités d'observation sont fixées à 1 pour toutes les zones. On définit deux entrées et deux sorties par zone objectif. La transmission des informations s'effectue lors du passage par une porte de sortie de l'objectif pour la zone $O1$. Pour les autres zones objectif, il est nécessaire de passer sur un des points de transmission pour envoyer l'information.
Deux modes de navigation sont envisagés, la ligne droite et le contournement de zone de danger. A ces deux modes de navigation s'ajoute l'action de balayage avec une largeur entre les raies de balayage de 1 km. La masse de carburant embarquée est de 300 kg.
L'environnement comporte quatre zones de danger. $DG1$ et $DG4$ ont un rayon égal à

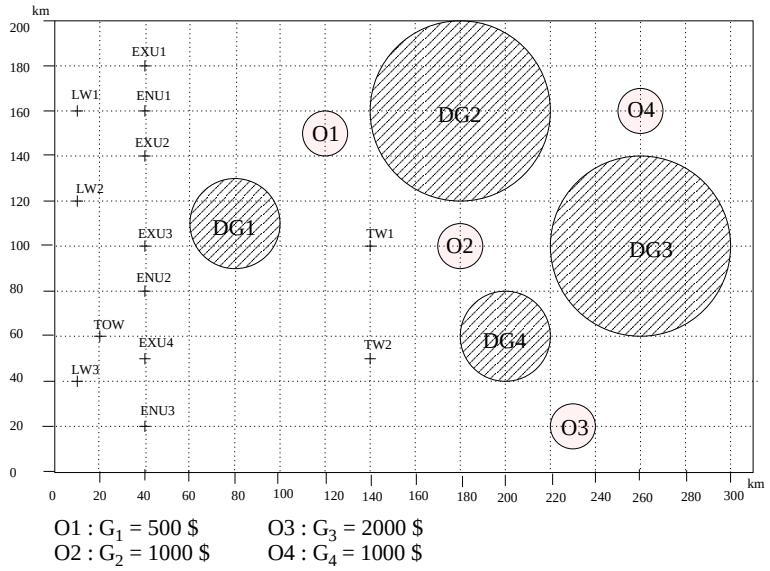


FIG. 4.4 – Mission 2

20 km. $DG2$ et $DG3$ ont un rayon égal à 40 km. Pour toutes les zones de danger, les probabilités d'existence, de détection du drone par la menace et de touche sont identiques à celles de la première mission.

La troisième mission comporte six zones objectif, dont le rayon est de 20 km. La carte de la mission globale est donnée figure 4.5. Chaque objectif a un gain de 1000 \$. Les probabilités d'observation sont fixées à 1 pour toutes les zones. La figure 4.6 montre qu'on définit quatre entrées et quatre sorties par zone objectif réparties régulièrement sur le contour de la zone : si le véhicule arrive droit devant une porte de sortie, la porte d'entrée la plus proche est au plus à 15 km. La transmission des informations s'effectue lors du passage par une porte de sortie de l'objectif pour les zones $O1$ et $O2$. Pour les autres zones objectif, il est nécessaire de passer sur un des points de transmission pour envoyer l'information.
Deux modes de navigation sont envisagés, la ligne droite et le contournement de zone de danger. A ces deux modes de navigation s'ajoute l'action de balayage avec une largeur entre les raies de 2 km. La masse de carburant embarquée est de 510 kg.

L'environnement comporte trois zones de danger. $DG1$ a un rayon égal à 20 km. $DG2$ et $DG3$ ont un rayon égal à 40 km. Pour toutes les zones de danger, les probabilités d'existence, de détection du drone par la menace et de touche sont identiques à celles de la première mission.

La quatrième mission comporte six zones objectif, dont le rayon est de 10 km. La carte de la mission globale est donnée figure 4.7. La réalisation de chaque objectif peut rapporter une récompense de 1000 \$. Les probabilités d'observation sont fixées à 1 pour toutes les zones. On définit deux entrées et deux sorties sur le contour de chaque zone objectif. La transmission des informations s'effectue lors du passage par la porte

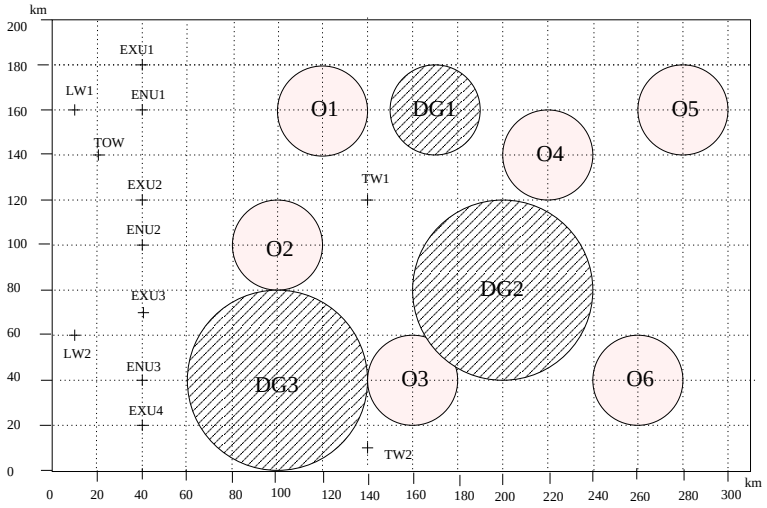


FIG. 4.5 – Mission 3

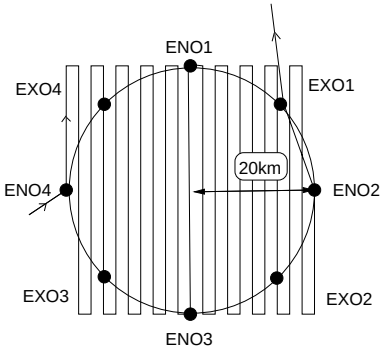


FIG. 4.6 – Disposition des entrées/sorties pour une zone objectif de rayon 20 km

de sortie des objectifs pour les zones $O1$ et $O2$. Pour les autres zones objectif, il est nécessaire de passer sur un des points de transmission pour envoyer l'information.
Le seul mode de navigation autorisé est la ligne droite auquel s'ajoute le balayage simple d'une zone objectif avec un rayon de balayage de 1 km. La masse de carburant embarquée est de 400 kg.
L'environnement comporte quatre zones de danger identiques à celles de la première mission.

4.2.3 Planification globale et replanifications

Une planification initiale est appliquée à chaque mission. Elle permet de déterminer des fenêtres temporelles appropriées pour les points d'entrée et de sortie de la zone ennemie.

Pour chaque mission, trois instants de replanification sont définis : un en début de mission, un en milieu de mission et le troisième en fin de mission. Trois événements déclenchants sont étudiés pour ces trois instants :

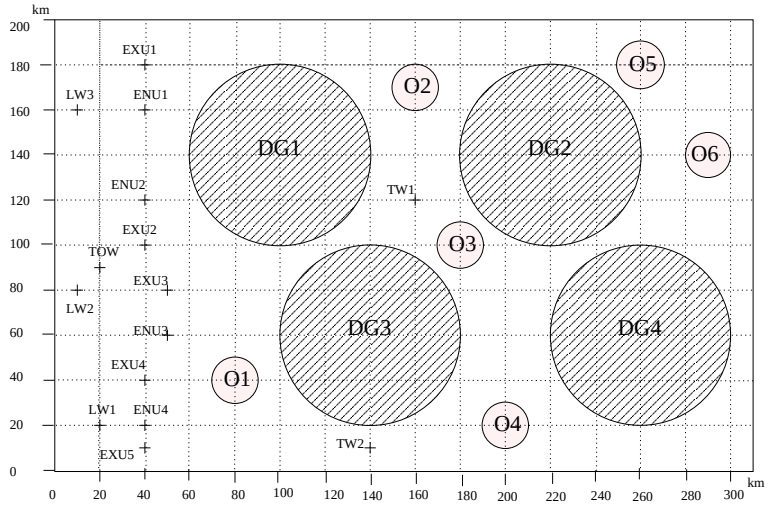


FIG. 4.7 – Mission 4

- Evt_1 fuite de carburant : le véhicule n'a plus assez de carburant pour réaliser le plan courant ;
- Evt_2 changement dans la carte des zones de danger ;
- Evt_3 changement dans la carte des zones objectif : disparition ou apparition de zones objectif, changement de gain, contraintes temporelles modifiées, ...

La combinaison des trois instants de replanification et des trois événements déclenchants permet de développer neuf scénarios de replanification par mission.

On indique ici le choix des scénarios tests sur l'exemple de la mission 1. Tous les scénarios testés sont présentés en Annexe D, ils ont tous la même structure : une planification initiale, trois instants de replanification et sur ces trois instants, test des trois types d'événements. En tout, 9 scénarios sont mis en place pour chaque mission, soit 36 scénarios au total.

Planification initiale (cas de la mission 1)

La figure 4.8 donne le meilleur itinéraire trouvé par la planification après deux jours de calculs pour la mission 1. Le meilleur plan trouvé par la planification passe dans l'ordre par $O1$, $O3$, transmet les informations pour la zone $O3$ en $TW2$, puis effectue $O4$, $O6$, $O5$, $O2$, transmet des informations en $TW1$, et sort de la zone ennemie par la porte $EXU3$ pour atterrir en $LW2$. Toutes les zones de danger sont contournées, sauf $DG1$.

Les instants d'application des calculs de replanification en début (D), milieu (M) et fin (F) de mission sont choisis sur cet itinéraire. Ils sont indiqués sur la carte respectivement par les points $n_1(D)$, $n_1(M)$ et $n_1(F)$ après les zones $O1$, $O4$ et $O5$. A la replanification D, l'objectif $O1$ a donc été réalisé. Pour la replanification M, les objectifs $O1$ et $O3$ ont été réalisés, $O4$ est traité, mais l'information qui lui est associée n'a pas été transmise. A la replanification F, $O1$ et $O3$ ont été réalisés, $O4$, $O6$ et $O5$ ont été

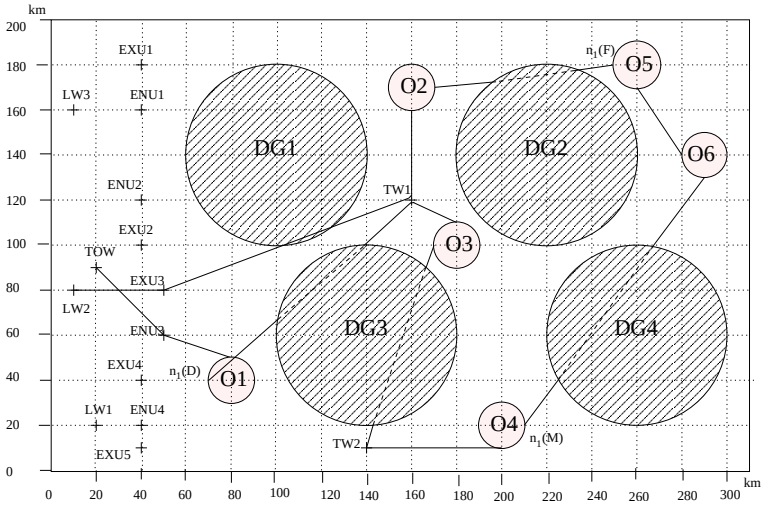


FIG. 4.8 – Mission 1 : Planification globale et choix des points de replanification

traités, mais les informations qui leur sont associées n'ont pas été transmises.

Les replanifications (mission 1)

Au point $n_1(D)$, l'événement Evt_1 est due à un manque de carburant pour terminer la mission. Le véhicule ne possède plus que 250 kg pour achever sa mission. Les objectifs et les dangers restent inchangés.

L'événement Evt_2 est dû à une mise à jour de la carte des dangers. Le danger $DG1$ est moins étendu et s'est déplacé vers le nord. $DG2$ et $DG4$ se sont déplacés vers le nord. $DG3$ s'est déplacé vers l'ouest et s'est scindé en deux zones de danger. La carte 4.9 résume ces changements.

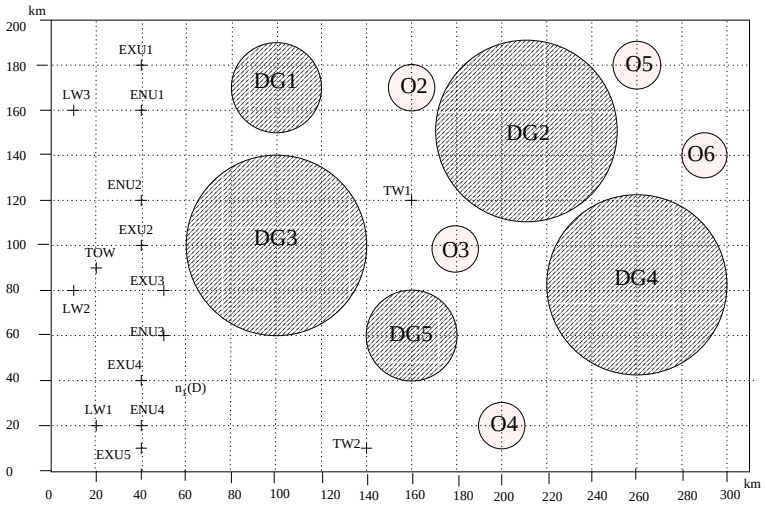
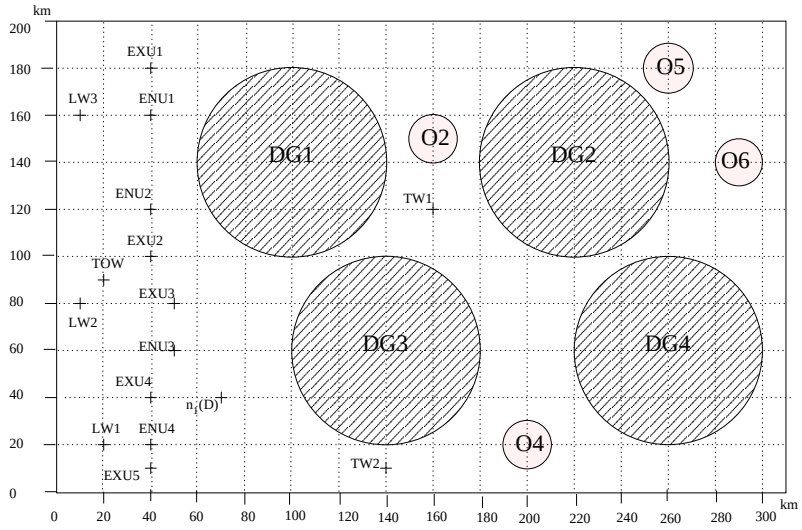


FIG. 4.9 – Mission 1 : replanification D2, mise à jour des dangers

L'événement Evt_3 est déclenché par une mise à jour des objectifs. $O2$ se déplace vers le sud et perd de l'importance, $O3$ disparaît, $O4$ et $O5$ ont des contraintes temporelles. $O6$ reste inchangé. La carte 4.10 illustre ces changements.



O2 : G_2 = 900 $, pas de contraintes temporelles
O4 : G_4 = 100 $, t_{min} = 5h , t_{max} = 8h30
O5 : G_5 = 2000 $, t_{min} = 5h , t_{max} = 8h30
O6 : G_6 = 1000 $, pas de contraintes temporelles

FIG. 4.10 – Mission 1 : replanification D3, mise à jour des objectifs

Au point $n_1(M)$, l'événement Evt_1 correspond à un carburant réduit à 100 kg. Les cartes des objectifs et du danger restent inchangées.

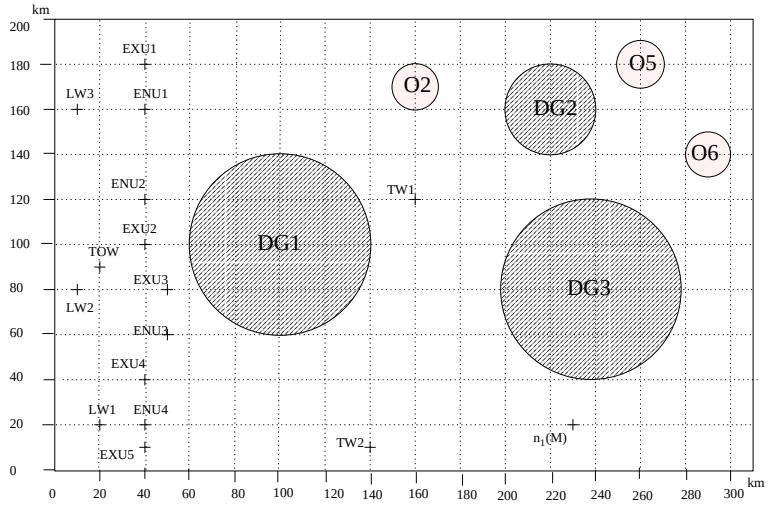


FIG. 4.11 – Mission 1 : replanification M2, mise à jour des dangers

L'événement Evt_2 est dû à la mise à jour de la carte des dangers : la zone de danger 1 se déplace vers le sud, la zone de danger 2 est réduite, la zone de danger 3 se déplace

vers l'ouest alors que la zone de danger 4 disparaît totalement. La carte 4.11 illustre les changements d'environnement.

L'événement $Evt3$ est une remise à jour des objectifs illustrée par la figure 4.12. L'objectif $O2$ prend de l'importance et possède des contraintes temporelles dures. L'objectif $O5$ perd de l'importance. Un nouvel objectif ($O7$) apparaît, avec des contraintes temporelles dures.

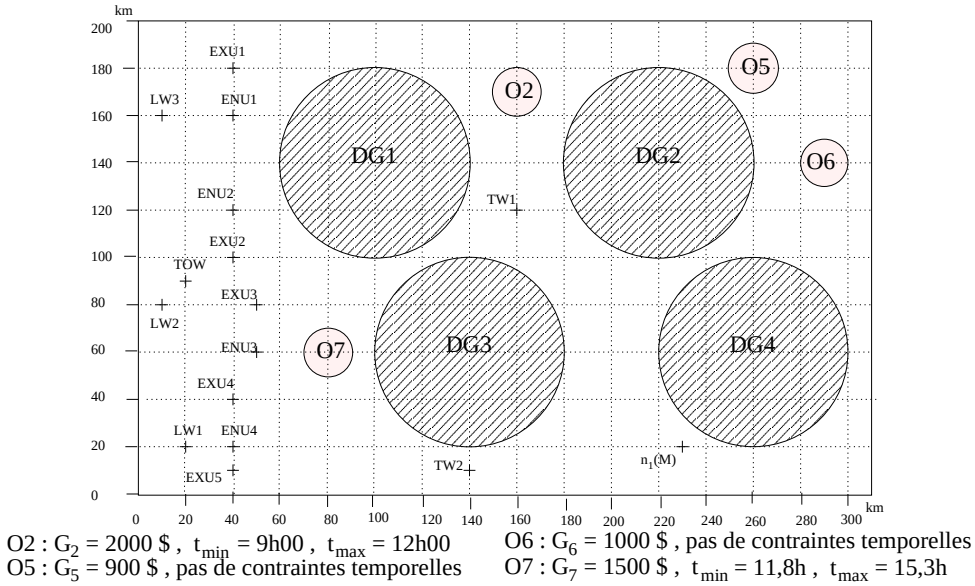


FIG. 4.12 – Mission 1 : replanification M3, mise à jour des objectifs

Au point $n_1(F)$, l'événement Evt_1 correspond à un carburant réduit à 70 kg. Les cartes des objectifs et du danger restent inchangées.

L'événement Evt_2 est dû à la mise à jour de la carte des dangers (Figure 4.13). Les zones de danger 1 et 3 fusionnent, la zone de danger 4 se déplace vers le nord.

L'événement $Evt3$ est une remise à jour des objectifs (Figure 4.14). Deux nouvelles zones objectif apparaissent sur la carte de la mission. $O7$ possède un gain de 2000 $ et des contraintes dures, $O8$ possède un gain de 1000 $ et pas de contraintes temporelles.

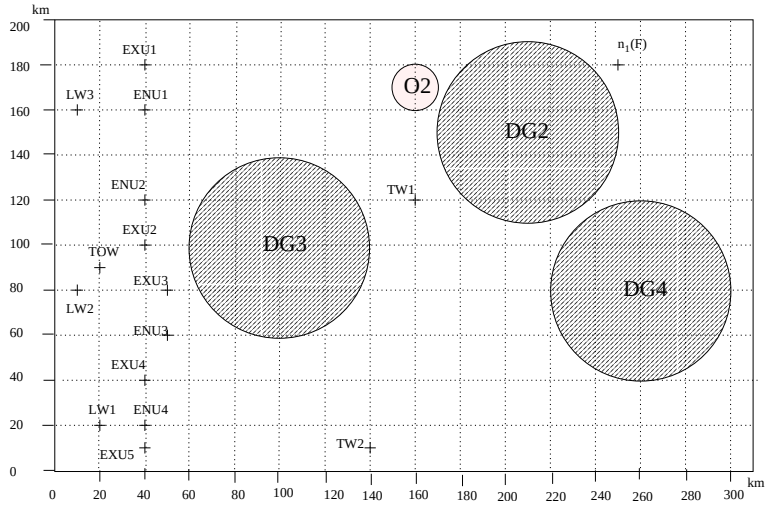
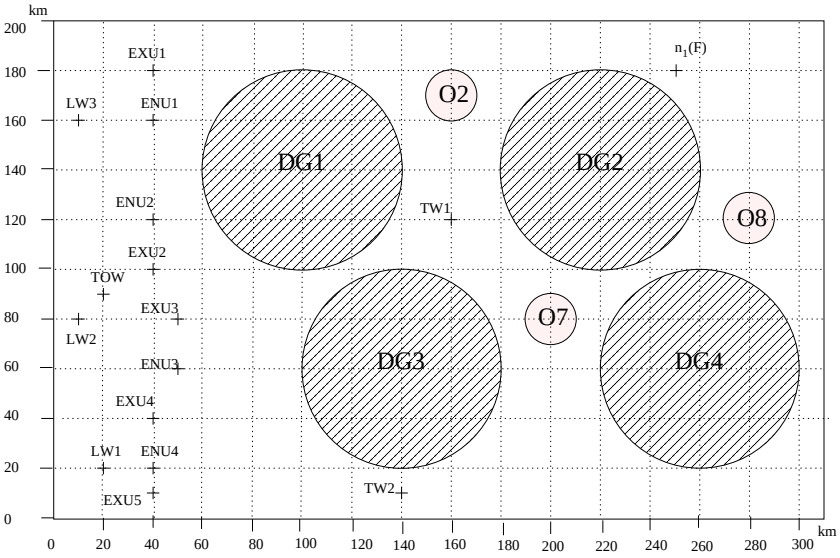


FIG. 4.13 – Mission 1 : replanification F2, mise à jour des dangers



O2 : G_2 = 1000 $, t_{min} = 17,8h , t_{max} =20h
O7 : G_7 = 2000 $, t_{min} = 13,5h , t_{max} = 18h
O8 : G_8 = 1000 $, pas de contraintes temporelles

FIG. 4.14 – Mission 1 : replanification F3, mise à jour des objectifs

4.3 Résultats et Analyses

4.3.1 Mise en œuvre des tests

Les tests sont effectués sur un processeur SunBlade150. Les 16 combinaisons de méthodes sont testées sur chaque scénario sur une durée maximale de 15 minutes.

Pour rappel, les méthodes testées sont les quatre méthodes d'évaluation du coût du nœud courant à un nœud fin (Section 2.4.1) :
- H_1 évalue les récompenses non encore obtenues sans tenir compte de l'utilisation des ressources jusqu'au nœud courant ;
- H_i évalue les récompenses non encore obtenues en tenant compte de l'utilisation des ressources jusqu'au nœud courant ;
- H_r est basé sur un problème relaxé : le problème est résolu sans optimiser le critère partiel à chaque développement de nœud, la vitesse est constante ;
- H_W s'inspire du formalisme proposé dans cette thèse. Une recherche arrière est effectuée sur un haut niveau de description pour affecter à chaque nœud une valeur heuristique utilisée lors du développement.

Quatre rangements (Section 2.4.3) sont testés :
- R_1 rangement en profondeur ordonnée guidée par g, la valeur optimale du critère partiel jusqu'au nœud courant ;
- R_2 rangement en profondeur ordonnée guidée par $g+h$ la somme de la valeur optimale du critère partiel jusqu'au nœud courant et de la valeur donnée par l'évaluation du coût du nœud courant à un nœud fin choisie ;
- R_3 rangement au meilleur g d'abord ;
- R_4 rangement au meilleur $g + h$ d'abord.

Les paramètres de l'algorithme sont identiques pour tous les tests.
Le nombre maximum de nœuds développés lors de la recherche d'une première solution (Section 2.3.2), noté cpt_{max}, est fixé à 100. Le choix de la valeur de ce paramètre dépend de la mission définie. Il s'agit d'effectuer un compromis entre la qualité de la première solution et la durée de calcul pour la trouver. Le nombre de nœuds maximum dans un itinéraire étant de $4 + 3.\overline{E_o}$, $\overline{E_o}$ correspondant au nombre d'objectifs réalisés par le plan, une mission réelle avec 5 ou 6 objectifs mènent à un plan contenant une vingtaine de nœuds. La recherche dans l'arborescence limitée au développement de cent nœuds paraît correcte. Développer moins de nœuds est une solution pour obtenir *a priori* un plan plus rapidement, de qualité moindre. Le risque est de ne pas développer assez de nœuds et de n'obtenir aucun plan admissible.
Les paramètres pour l'optimisation du critère (Section 2.3.4) sont les suivants : le nombre d'itérations maximum possibles I_{max} est fixé à 1000, la différence entre deux valeurs du critère optimisé ϵ_{max} est fixée à 1. L'expérience montre que le nombre d'itérations maximum n'est jamais atteint. Le choix de la différence entre deux valeurs du critère optimisé dépend de l'ordre de grandeur des critères optimisés. Pour les missions choisies, une différence entre deux critères d'une valeur de 1 est négligeable.

Les paramètres des méthodes utilisées dans l'algorithme sont choisis de la même façon. Ainsi le nombre de nœuds développés dans la méthode H_r est fixé à 100, pour les mêmes raisons que précédemment. Le paramètre γ de la règle 2 (méthode d'élagage, Section 2.4.2) a pour valeur 0,001. Cette valeur indique le degré de confiance dans la fonction d'évaluation du nœud courant vers un nœud but. Plus la valeur est petite, plus la confiance est grande. Il faut ici effectuer un compromis entre la confiance que l'on a dans la fonction d'évaluation du nœud courant vers un nœud but et la durée de calcul. En effet, si on a peu confiance en l'heuristique, il y aura peu de nœuds élagués : l'arborescence va donc être probablement explorée quasiment entièrement. D'un autre côté, si on accorde trop de confiance à l'heuristique, et qu'elle n'est pas efficace, alors l'arborescence va être peu explorée car beaucoup de nœuds potentiellement intéressants seront élagués. Des tests préliminaires montrent que la valeur 0,001 donne des résultats corrects.

La table 4.2 récapitule les différentes missions.

Mission	Nbre zones objectif	Nbre E/S	Navigation
1	6	2	ligne droite/ contournement
2	4	2	ligne droite/ contournement
3	6	4	ligne droite/ contournement
4	6	2	ligne droite

TAB. 4.2 – Récapitulatif des missions tests

Les 36 scénarios sont construits sur la base d'un choix de mission parmi les 4 proposées, un choix d'instant par D, M et F, et d'un choix d'événement par Evt_1, Evt_2 ou Evt_3.

Pour chaque scénario, le plan initial est évalué selon l'algorithme 9.

Ensuite, au cours du calcul de planification, à chaque amélioration du critère pour un plan allant de n_1 à n_e et satisfaisant les contraintes temporelles, de carburant et de danger imposées par le cahier de charges de la mission, un fichier solution est créé. Ce fichier texte contient :

- la méthode de calcul employée ;
- la valeur prévue pour le critère ;
- la durée de calcul pour obtenir le plan ;
- le plan (succession de points de mission, action associée et durée) ;
- la quantité de carburant utilisée.

Les critères d'évaluation des méthodes retenus sont ceux de la section 3.6.4 : ils varient selon le contexte de replanification.

Les résultats des tests suivant ces critères sont présentés en Annexe D.

4.3.2 But de l'analyse

L'analyse a pour but de répondre à deux problèmes. Le premier est de dégager une ou plusieurs méthodes à embarquer pour répondre à un contexte de replanification particulier. Le second est de dégager la ou les méthodes embarquables dans le système de drone et à intégrer dans l'architecture de contrôle pour répondre à tous les contextes de replanification envisagés. Cette ou ces méthodes doivent donc obtenir un plan de manière (quasi) systématique dans tous les cas, et donner la meilleure espérance de gains possible.

4.3.3 Moyens employés

La ou les méthodes dégagées doivent permettre d'obtenir la meilleure espérance de gains possible sur la totalité des scénarios de replanification. Les tests effectués ont été construits de manière systématique sur trois classes d'instants de replanification et sur trois classes d'événements possibles.

L'idée est de comparer les algorithmes uniquement sur la qualité du critère de manière homogène. En effet, les contraintes temporelles de la replanification sont déjà prises en compte dans les critères retenus pour qualifier les méthodes (Section 3.6.4). L'homogénéité découle du principe d'espérance mathématique. Ainsi, considérons le problème simplifié de k scénarios $S_1, \ldots, S_k$.

PROPRIÉTÉ 5 *Soient $f_1, \ldots, f_k$ leurs fréquences d'occurrence respectives, et $J^i_1, \ldots, J^i_k$ la valeur du critère obtenu pour chaque scénario par la méthode i, alors chaque méthode peut être évaluée selon la valeur d'un* ***critère généralisé*** *J^i.*

$$J^i = \sum_{j=1}^{k} f_j . J^i_j$$

Le but est de définir les valeurs des f_j de la façon la plus réaliste possible. Pour nos scénarios, il s'agit de définir quelles sont les fréquences d'occurrence pour les trois instants de replanification, et pour les trois classes d'événements possibles. Notons f_D, f_M, f_F les fréquences d'occurrence des trois instants de replanification et f_{Evt1}, f_{Evt2} et f_{Evt3} celles des trois événements.

Sans perte de généralité, il est possible de dire que les trois instants de replanification ont la même fréquence d'occurrence. Le calcul du critère généralisé peut donc ne prendre en compte que la fréquence d'occurrence des trois événements.

A l'opposé, il est difficile d'évaluer les fréquences d'occurrence des trois événements. Afin d'effectuer une étude de robustesse, on propose les hypothèses suivantes :

- L'événement $Evt1$ correspond à une perte brutale de carburant. On suppose que cet événement a une fréquence d'occurrence faible. L'événement se produit soit une fois sur 1000 missions, soit une fois sur 100 missions, soit une fois sur 10 missions.

$$f_{Evt1} \quad \in \quad \{\tfrac{1}{1000}, \tfrac{1}{100}, \tfrac{1}{10}\}$$

- L'événement $Evt2$ correspond à la mise à jour de la carte des dangers. On suppose que l'événement possède une fréquence d'occurrence moyenne. L'événement se produit soit une fois toutes les deux missions, soit une fois par mission, soit deux fois par mission.
$$f_{Evt2} \in \{\tfrac{1}{2}, 1, 2\}$$
- L'événement $Evt3$ est une mise à jour de la carte des objectifs. On suppose que l'événement possède une fréquence d'occurrence moyenne. L'événement se produit soit une fois toutes les deux missions, soit une fois par mission, soit deux fois par mission.
$$f_{Evt3} \in \{\tfrac{1}{2}, 1, 2\}$$

On étudie donc les 27 combinaisons d'occurrences possibles. Ceci permet d'étudier la robustesse des analyses vis à vis de la fréquence d'occurrence des trois événements, qui est mal connue.

4.3.4 Choix d'une méthode pour chaque contexte de replanification

Evaluation des méthodes selon la criticité

Pour chaque scénario $S_1, \ldots, S_k$, on définit une criticité pour chaque scénario, indiquant une replanification critique, nécessaire ou éventuelle. On évalue alors les critères généralisés pour les scénarios de même criticité, et ce pour les 27 combinaisons de fréquences d'événements. Pour les replanifications critiques et nécessaires, on calcule deux critères généralisés : la qualité de la première solution admissible et la qualité de la meilleure solution calculée en moins de 20 s pour le cas critique, la qualité de la première solution améliorant de manière significative le plan courant et la qualité de la meilleure solution en durée limitée à 5 min (sous réserve qu'elle améliore de manière significative la solution précédente) pour le cas nécessaire. Pour les replanifications éventuelles, seul le critère généralisé correspondant à la meilleure solution trouvée en moins de 15 min est étudié.

Au niveau des criticités, les tests montrent que les replanifications critiques sont au nombre de 22 sur 36 scénarios. On compte 6 replanifications nécessaires et 8 replanifications éventuelles. Ceci représente une bonne représentation des cas de replanification réels. En effet, il sera rare de lancer une replanification alors que le plan courant ne mène pas vers une destruction ou une grande dégradation du plan.

Robustesse des résultats

Le but de cette section est d'étudier les variations du classement des méthodes si la combinaison de fréquence varie. Pour cela, on trace le profil des critères généralisés pour chaque méthode, en fonction de la combinaison des fréquences d'occurrence des événements.

La figure 4.15 montre l'étude de la première solution obtenue en cas de replanification critique. On distingue six groupes de méthodes. En considérant la valeur du critère

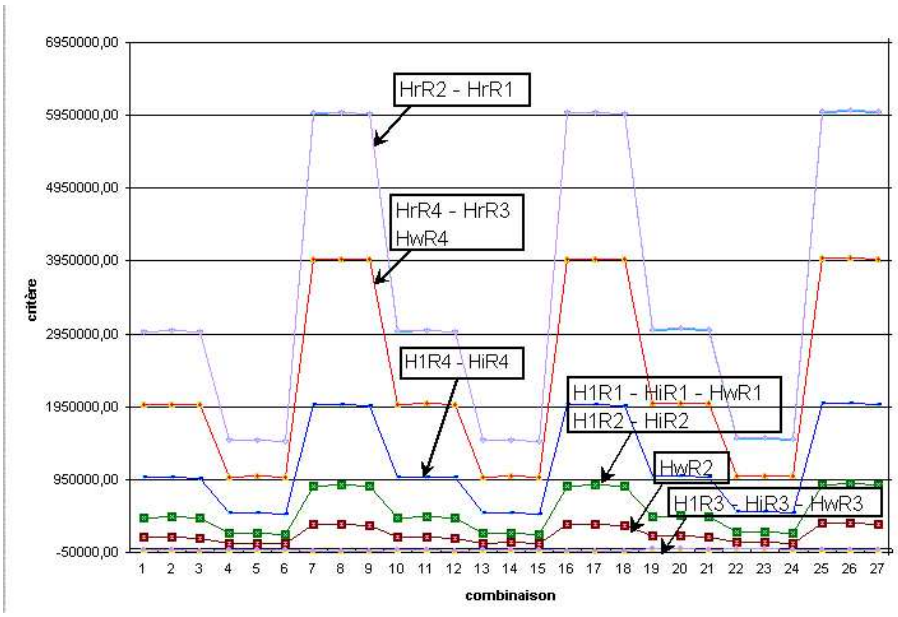


FIG. 4.15 – Replanification critique - Qualité de la première solution admissible : critère généralisé fonction des combinaisons de fréquences d'occurrence

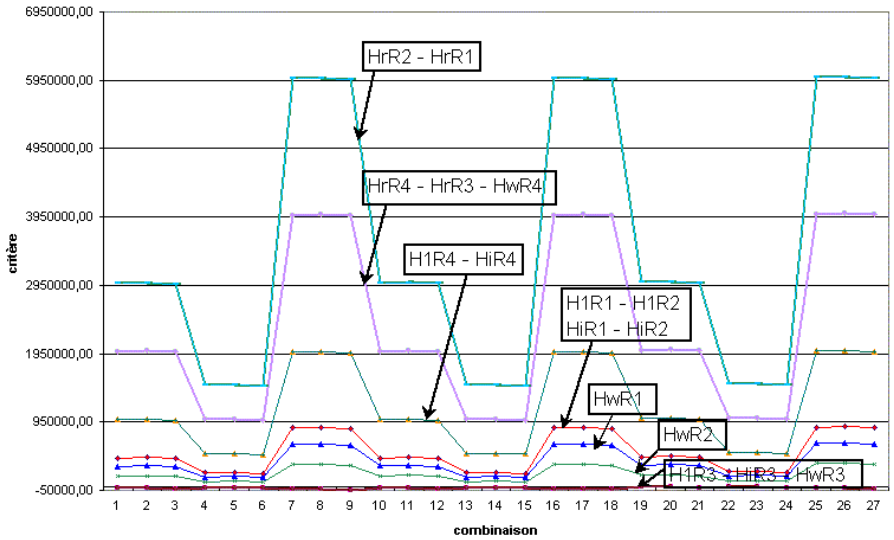


FIG. 4.16 – Replanification critique - Qualité de la meilleure solution en temps limité à 20 s : critère généralisé fonction des combinaisons de fréquences d'occurrence

généralisé, l'ordre des groupes est le même pour toutes les combinaisons d'algorithmes. Cela signifie que l'on peut se limiter à l'étude des meilleurs groupes seulement et que la fréquence des événements a peu d'incidence sur l'analyse.

La figure 4.16 montre l'étude de la meilleure solution obtenue en durée limitée à 20 s pour une replanification critique. Comme précédemment, on distingue des groupes de méthodes se comportant de manière identique. L'ordre des groupes est le même pour toutes les combinaisons d'algorithmes.

On conclut donc que pour le cas de la replanification critique, l'étude peut se limiter à une combinaison unique de fréquences d'occurrence des événements.

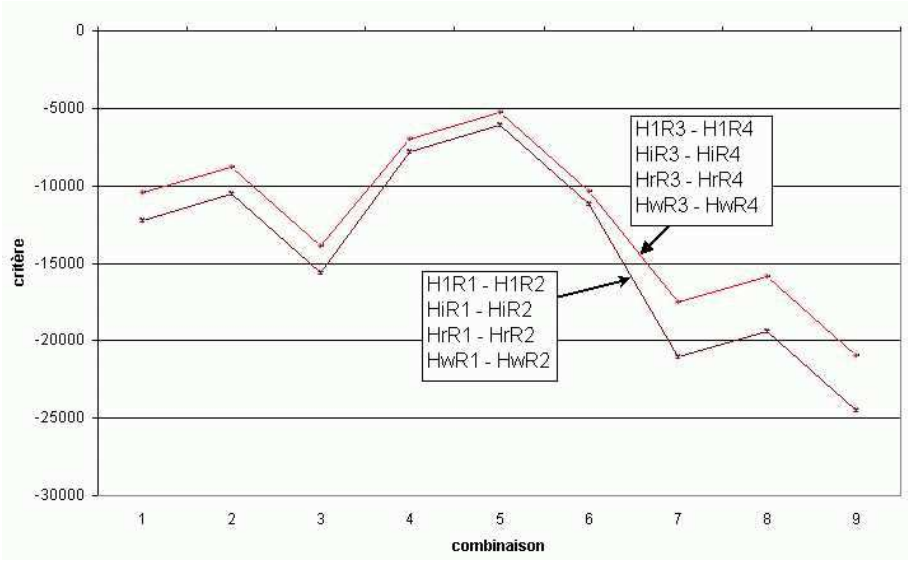


FIG. 4.17 – Replanification nécessaire - Qualité de la première solution admissible améliorant le critère : critère généralisé fonction des combinaisons de fréquences d'occurrence

L'étude de la première solution obtenue pour une replanification nécessaire (Figure 4.17) permet de distinguer deux groupes de méthodes se comportant de manière identique. L'ordre des groupes est le même pour toutes les combinaisons d'algorithmes.

Pour la meilleure solution obtenue en durée de calcul limitée à 5 min (Figure 4.18), l'étude de robustesse permet de dégager trois groupes de méthodes se comportant de manière similaire. Toutefois, l'étude montre aussi que dans un même groupe, selon les combinaisons étudiées, il existe des différences de classement ; par exemple dans le groupe composé de H_1R_3, H_iR_3, H_WR_3, H_rR_1, H_rR_2 et H_WR_4, les courbes se croisent. On s'efforcera donc lors de l'analyse des résultats de ne pas distinguer les disparités entre les méthodes d'un même groupe.

L'étude de la meilleure solution obtenue en durée limitée à 15 min pour une replanification éventuelle (Figure 4.19) ne permet pas de distinguer des groupes de méthodes se comportant de manière identique. A priori, toutes les méthodes donnent des résultats voisins pour toutes les combinaisons de fréquences d'événements considérées.

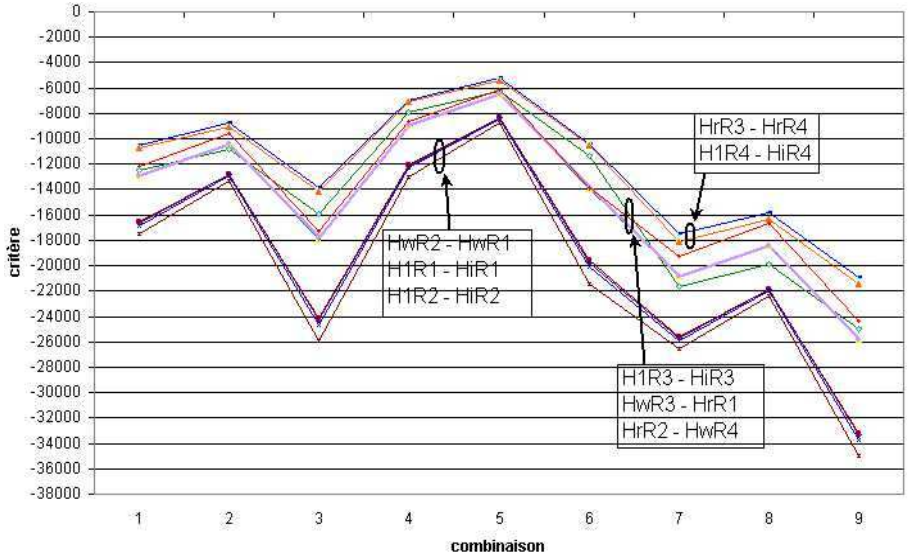


FIG. 4.18 – Replanification nécessaire - Qualité de la meilleure solution en temps limité à 5 min : critère généralisé fonction des combinaisons de fréquences d'occurrence

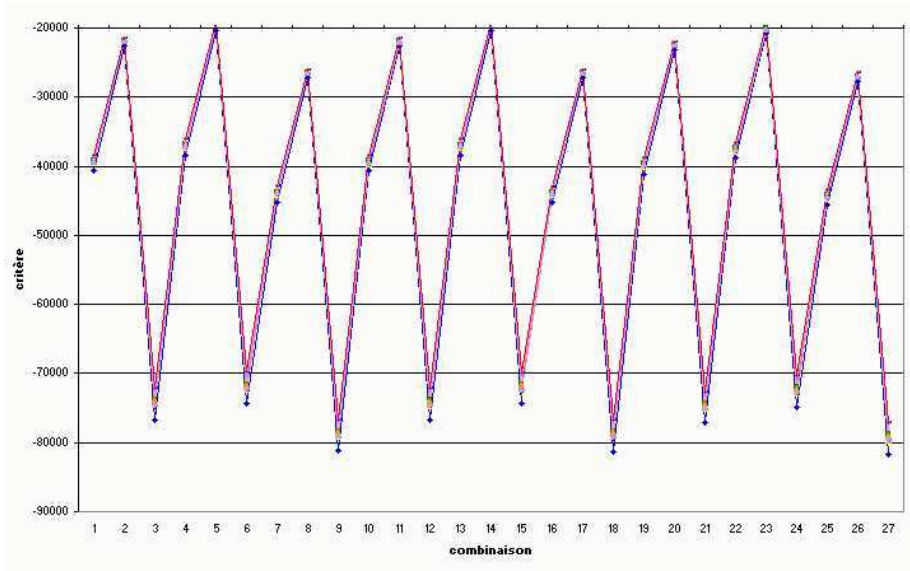


FIG. 4.19 – Replanification éventuelle - Qualité de la meilleure solution en temps limité à 15 min : critère généralisé fonction des combinaisons de fréquences d'occurrence

L'étude de robustesse a permis de montrer que, pour toutes les combinaisons de fréquences d'occurrence des événements, l'analyse des résultats aboutit à des conclusions comparables.

Analyse du contexte de replanification critique

Etudions tout d'abord la qualité de la première solution admissible obtenue. La figure 4.15 permet de distinguer six groupes de méthodes.

1. Les méthodes H_1R_3, H_iR_3 et H_WR_3 ont des résultats parfaitement identiques. Sur le panel de tests effectués, ce groupe présente l'avantage majeur de toujours calculer une solution menant à un risque de destruction de l'appareil inférieur à 18%. Ce risque est calculé en fonction du critère de valeur maximale obtenue et du prix de l'appareil. Dans 95% des cas, ces méthodes calculent un plan pour lequel le risque de destruction est inférieur à 0,01%.
2. La méthode H_WR_2 permet de toujours calculer une solution menant à un risque de destruction de l'appareil inférieur à 18%. Elle permet de limiter ce risque à 0,01% dans environ 90% des cas.
3. Les méthodes H_1R_1, H_iR_1, H_WR_1, H_1R_2 et H_iR_2 calculent dans tous les cas des solutions menant à un risque de destruction de l'appareil pouvant atteindre 50%. Néanmoins, elles permettent de limiter ce risque à 0,01% dans environ 90% des cas. Ce sont encore de bonnes méthodes.
4. Les méthodes H_1R_4 et H_iR_4 mènent le système à l'échec dans un cas sur 22, soit environ 5% de risque d'échec très probable de la mission en cas de replanification critique. Elles calculent des plans pour lesquels le risque de destruction est inférieur à 0,01% dans environ 90% des cas.
5. Les méthodes H_rR_4, H_rR_3, et H_WR_4 mènent le système vers un échec très probable dans environ 10% des cas. Elles calculent des plans pour lesquels le risque de destruction est inférieur à 0,01% dans environ 85% des cas.
6. Les méthodes H_rR_2 et H_rR_1 sont à priori les plus mauvaises méthodes pour le calcul d'un plan lors d'une replanification dans un contexte critique. Ces méthodes mènent le système à l'échec dans trois cas sur 22, soit 14% de risque d'échec très probable de la mission en cas de replanification critique. Elles calculent des plans pour lesquels le risque de destruction est inférieur à 0,01% dans environ 80% des cas.

Etudions à présent la qualité de la meilleure solution obtenue pour une durée de calcul inférieure à 20 s. En règle générale, les courbes n'ont pas évolué par rapport au cas précédent. C'est le signe que les méthodes développées ne permettent pas ou très peu d'améliorer très rapidement la première solution admissible. La figure 4.16 montre que seule la méthode H_WR_1 a amélioré ses performances de manière significative en terme de qualité. Cependant, certaines améliorations ne sont pas visibles.

1. Pour H_1R_3, H_iR_3 et H_WR_3, la valeur du plan ne subit une amélioration que dans 4,5% des cas. L'amélioration est alors d'environ 30%. La recherche d'une meilleure solution semble donc inutile.
2. La méthode H_WR_2 ne permet aucun amélioration.
3. Pour H_WR_1, deux améliorations significatives sur 22 cas testés sont observées.
4. Les méthodes H_1R_1, H_iR_1, H_1R_2 et H_iR_2 présentent la particularité de subir de nombreuses améliorations : dans 24% des cas testés, les plans améliorent leur

qualité. Sur la totalité des tests, les plans s'améliorent en moyenne de 30%. Cette amélioration n'est pas visible sur la figure 4.16 à cause de l'importance des échecs dans le critère généralisé. L'amélioration est illustrée sur la figure 4.20. En abscisse, on porte tous les scénarios traités, en ordonnée, la moyenne de l'amélioration sur ce groupe de méthodes.

5. Les méthodes H_1R_4 et H_iR_4 ne mènent jamais à une amélioration de la qualité de la solution.
6. Les méthodes H_rR_4, H_rR_3 et H_WR_4 ne mènent quasiment jamais à une amélioration. Les améliorations ont une fréquence d'environ 7% et n'apporteront qu'un gain faible : la meilleure amélioration observée est de 20%.
7. Le dernier groupe est toujours composée des méthodes H_rR_2 et H_rR_1. La qualité de la solution est améliorée dans environ 20% des cas. Cette amélioration n'est pas très importante. En moyenne sur tous les cas traités, l'amélioration du critère est de 3%.

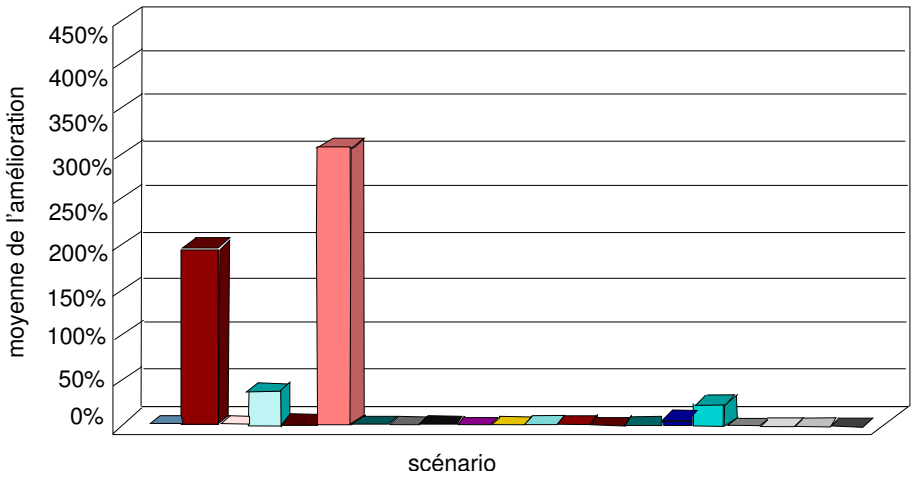


FIG. 4.20 – Amélioration de la qualité du plan pour les méthodes H_1R_1, H_iR_1, H_1R_2 et H_iR_2

On constate que le rangement au meilleur g d'abord (R_3) donne les meilleurs résultats. C'est une méthode de rangement qui ne prend pas en compte l'évaluation du critère futur. *A priori*, cette méthode n'aurait pas dû donner pas de meilleurs résultats que le rangement en profondeur ordonnée guidée par g. En effet, en réponse critique, il semble préférable de trouver un plan au plus vite, ce que permettent les descentes ordonnées au meilleur d'abord. Les tests effectués contredisent cet *a priori*. Pour les mêmes méthodes d'évaluation, le rangement en profondeur ordonnée guidée par g (R_1) donne de moins bons résultats. Une première conclusion pour le cas de replanification en contexte critique est qu'il est préférable d'utiliser la méthode de rangement au meilleur d'abord.

On peut donner deux interprétations au fait que la méthode de rangement au meilleur $g + h$ d'abord ne donne pas les meilleurs résultats. Tout d'abord, les méthodes

d'évaluation des coûts futurs peuvent être de mauvaise qualité. De fait, le guidage dans l'arbre d'exploration est mal effectué, et les méthodes n'aboutissent pas. Ensuite, on peut supposer que ces méthodes prennent trop de temps pour l'évaluation, et ne peuvent donc pas explorer l'arbre aussi rapidement qu'une méthode de guidage simple. Dans le cas critique, les méthodes de guidage doivent donc être très rapides. D'après les tests, ce sont les méthodes les plus simples en termes de calculs qui ont les meilleurs résultats. C'est ce qui explique que la méthode d'évaluation H_r ait de si mauvais résultats par exemple.

On remarque aussi que les méthodes d'évaluation H_1 et H_i sont équivalentes. Associées au même rangement, elles font toujours parties des mêmes groupes. Pour les rangements en profondeur ordonnée, la méthode H_W est la plus efficace, quelle que soit la méthode de rangement à laquelle elle est associée : le guidage et l'élagage produits par cette méthode sont meilleurs.

Pour le cas critique, on préférera les méthodes d'évaluation les plus simples : le groupe de méthodes H_1R_3, H_iR_3 et H_WR_3, utilisant un rangement au meilleur g d'abord et des méthodes d'évaluation rapides, sera le meilleur. Dans ce cas, l'heuristique ne sert qu'à élaguer.
A l'opposé, les méthodes utilisant l'évaluation H_r, qui utilise plus de durée de calcul, sont à proscrire.
Enfin, dans le cas d'une replanification critique, l'amélioration de la solution est très peu fréquente. Il semble donc inutile d'essayer d'améliorer la solution rapidement.

Analyse du contexte de replanification nécessaire

L'étude de la qualité de la première solution lors d'une replanification en contexte nécessaire permet de dégager deux groupes de méthodes (Figure 4.17).

1. Le premier groupe contient toutes les méthodes effectuant un rangement en profondeur ordonnée guidée par g ou $g + h$.
2. Le second groupe contient toutes les méthodes effectuant un rangement au meilleur g ou $g + h$ d'abord.

Les deux groupes apportent des solutions avec un risque de destruction de l'appareil négligeable. Seul un scénario sur six permet de dégager les méthodes de rangement en profondeur ordonnée guidée par g ou $g + h$ comme les meilleures. Pour les cinq autres tests, les résultats sont identiques ou légèrement à l'avantage du second groupe de méthodes. On peut donc se demander si les tests sont révélateurs ou pas du comportement des algorithmes.

Pour l'étude de la meilleure solution significative obtenue en moins de 5 min de calcul (Figure 4.18), on distingue trois groupes de méthodes par leur faculté à améliorer la première solution trouvée. Ainsi, on peut voir que les courbes du premier groupe ont baissé de la figure 4.17 à la figure 4.18.

1. Le premier groupe est composé des méthodes H_WR_2, H_WR_1, H_1R_1, H_iR_1, H_1R_2 et H_iR_2. Une première observation permet de constater que toutes ces méthodes

sont dirigées par des méthodes de rangement en profondeur ordonnée guidée par g ou $g + h$. Ces méthodes améliorent la qualité du plan dans 3 cas sur 6. Les améliorations ne sont pas homogènes, elles vont de 0,3% à plus de 380%.

2. Le second groupe est composé des méthodes H_1R_3, H_iR_3, H_WR_3, H_rR_1, H_rR_2 et H_WR_4. Le groupe est composé de deux sous-groupes de méthodes. Le premier est composé des méthodes H_rR_1 et H_rR_2, qui appartenaient au premier groupe lors du calcul de la première solution. Ces méthodes n'ont pas calculé de plan de meilleure qualité en temps voulu. On peut supposer que cela vient du fait que la durée du calcul est limitée à 5 min. Le deuxième sous-groupe de méthodes a amélioré sa première solution en moyenne de plus de 70%. Ce sont en majorité des méthodes utilisant le rangement au meilleur g d'abord.
3. Le dernier groupe comporte les méthodes utilisant les rangements au meilleur g ou $g+h$ d'abord qui n'ont pas ou très peu amélioré leur première solution, soit H_rR_3, H_rR_4, H_1R_4 et H_iR_4. L'amélioration la plus grande observée dans ce groupe est une amélioration de 3% de la première solution. Ce sont en majorité les méthodes utilisant le rangement au meilleur $g + h$ d'abord qui appartiennent à ce groupe.

Comme pour le cas critique, les méthodes d'évaluation les moins gourmandes en durée de calcul donnent les meilleurs résultats. Par contre, ce sont les rangements en profondeur ordonnée guidée par g ou $g + h$ qui prédominent. Ces méthodes trouvent les plans de meilleure qualité lorsque la durée du calcul dépasse 10 s. Etudions la façon dont les méthodes améliorent la qualité du plan. Les tests significatifs (c'est-à-dire ceux pour lesquels le plan est amélioré de plus de 10%, et pour lesquels les méthodes ne réagissent pas de manière identique), sont les scénarios Mission 1-D2 et Mission 3-D3. Pour ces deux scénarios et pour chaque méthode, on trace la qualité du meilleur plan trouvé en fonction de la durée de calcul. Les figures 4.21 et 4.22 illustrent cette étude.

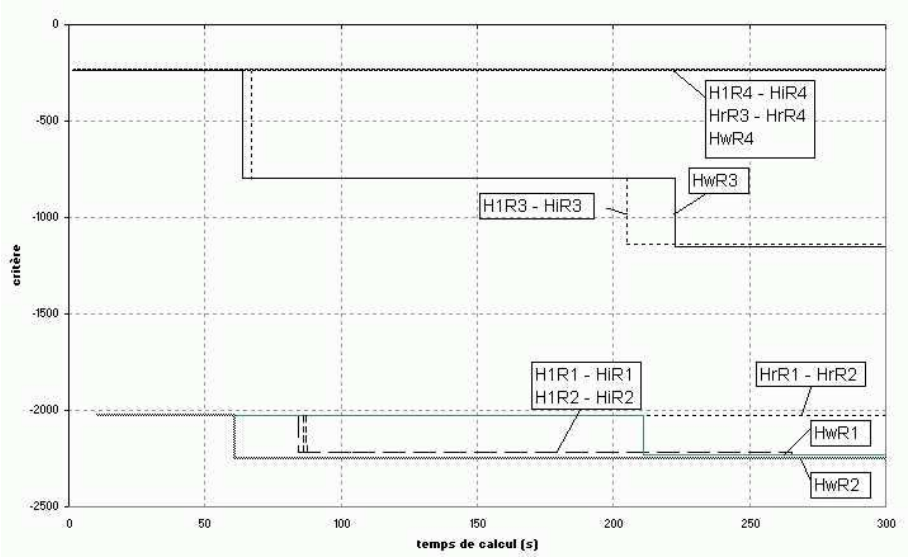


FIG. 4.21 – Evolution du critère en fonction de la durée de calcul - Mission 1-D2

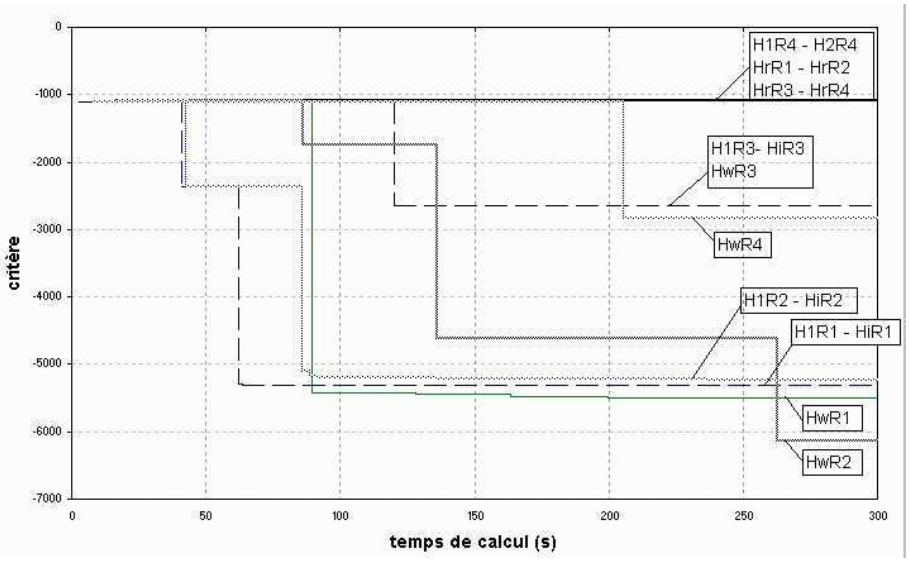


FIG. 4.22 – Evolution du critère en fonction de la durée de calcul- Mission 3-D3

On constate que toutes les évolutions significatives du critère s'effectuent après 50 s de calcul. En règle générale, ce sont comme prévu les méthodes de rangement en profondeur ordonnée qui trouvent les meilleures améliorations. Celles-ci surviennent moins de 2 min après le début des calculs. En cas de replanification nécessaire, il sera donc judicieux d'attendre cette durée pour mettre à jour le plan.
On peut penser que les méthodes au meilleur d'abord se "perdent" dans l'arbre sans parvenir à déterminer un nouveau plan complet, alors que les méthodes de rangement en profondeur ordonnée, par nature plus efficaces pour développer un plan complet, parviennent à améliorer le plan courant. Cependant, les méthodes de rangement en profondeur ordonnée ont l'inconvénient d'être très concentrées sur une branche de l'arbre. Si cette branche contient beaucoup de branches filles qui n'apportent pas de résultat, ces méthodes peuvent se révéler inefficaces si l'élagage n'a pas eu pour effet de couper cette branche inutile.
Etant donné que toutes les méthodes utilisant H_r sont parmi les groupes les moins bons, on conclut que les méthodes H_1, H_i et H_W permettent un élagage efficace de l'arbre de recherche. De plus, il existe très peu de différence entre les méthodes guidées par g et celles guidées par $g + h$ pour toutes les méthodes, sauf H_W. Cela signifie que les méthodes de guidage développées ne sont pas efficaces. Néanmoins, la différence entre les résultats de $H_W R_1$ et $H_W R_2$ permet de dire que la méthode H_W a une action en guidage non négligeable. Cependant il est difficile de dire si cette action de guidage est positive ou non.

Pour les replanifications en contexte nécessaire, on constate que les méthodes de rangement en profondeur ordonnée guidée par g ou $g + h$ donnent les résultats les plus satisfaisants. La méthodes d'évaluation H_r ne semble pas être efficace pour améliorer une solution, et ce quel que soit le rangement utilisé. Les méthodes d'évaluation dites "rapides" (H_1, H_i et H_W) donnent les meilleurs résultats.

Analyse du contexte de replanification éventuelle

Les analyses de qualité pour les replanifications en contexte éventuel (Figure 4.19) ne permettent pas de dégager des méthodes meilleures que les autres, si ce n'est H_rR_2 qui est souvent la meilleure méthode en terme de qualité. On va donc dans un premier temps étudier la durée de calcul pour obtenir ces solutions, puis on étudiera l'évolution de la solution en fonction de la durée du calcul.

Comme pour les analyses de qualité, on étudie les différentes combinaisons de fréquences d'occurrence des événements. Pour cela, on trace les durées de calcul généralisées pour chaque méthode, en fonction de la combinaison des fréquences d'occurrence des événements (Figure 4.23).

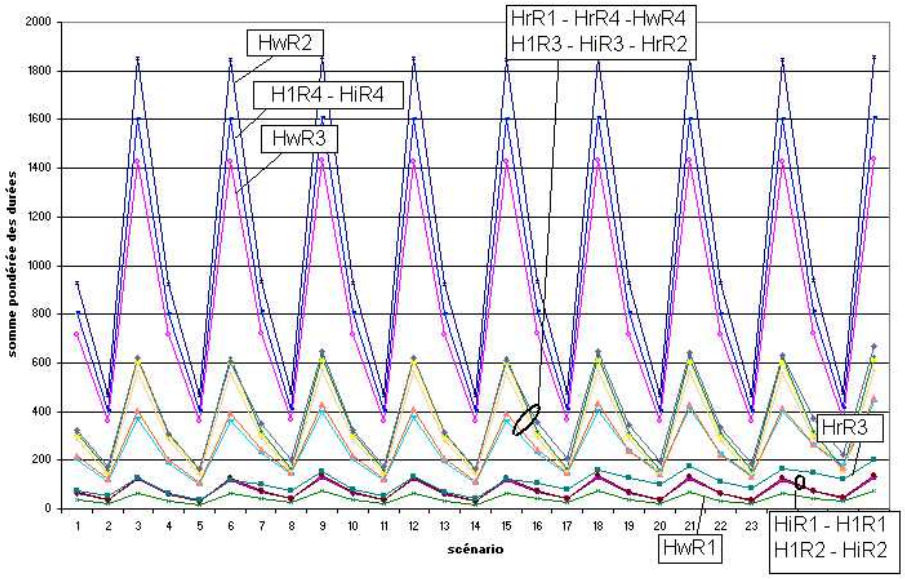


FIG. 4.23 – Etude de la robustesse par rapport à la durée de calcul - contexte de replanification éventuelle

Le tracé des courbes permet de dégager cinq groupes de méthodes.

1. Le premier groupe est composé de la méthode H_WR_1. Elle permet d'obtenir la meilleure solution en une dizaine de secondes. Cette méthode permet d'obtenir dans 90% des cas une solution d'une qualité très proche de la meilleure solution obtenu par l'ensemble des algorithmes (environ 7% de différence maximum). Par

contre dans les 10% restants, on trouve une différence d'environ 30% avec la solution la meilleure, obtenue par H_rR_2.

2. Le second groupe est composé des méthodes H_iR_1, H_1R_1, H_1R_2 et H_iR_2. On constate que ces méthodes donnent leur meilleur résultat en environ 10 s. La plus longue des réponses est de 30 s. La qualité des résultats est très bonne. Dans tous les cas, le meilleur résultat a moins de 10% de différence avec le meilleur résultat obtenu sur l'ensemble des méthodes.
3. Le troisième groupe est composée de la méthode H_rR_3. Cette méthode permet d'obtenir la meilleure réponse en une moyenne de 75 s. Lorsqu'on étudie les résultats de cette méthode dans le temps (amélioration de la solution), on s'aperçoit qu'elle n'obtient pas la même qualité de solution que les autres méthodes dans la même durée : une solution de même qualité est obtenue en une durée de calcul plus longue.
4. Le quatrième groupe est composé des méthodes H_rR_1, H_rR_4, H_WR_4, H_1R_3, H_iR_3 et H_rR_2.
5. Les trois derniers groupes sont composés respectivement des méthodes H_WR_3, puis H_1R_4 et H_iR_4 qui ont exactement le même comportement enfin H_WR_2. Il faut remarquer que les solutions obtenues sont pour la majorité des cas les meilleures solutions. En outre, si ces méthodes obtiennent les meilleures solutions en une durée plus longue que les autres, elles ont obtenu les mêmes résultats en termes de qualité en une durée comparable.

En règle générale, pour le cas de replanification en contexte éventuel, les méthodes de rangement en profondeur ordonnée sont plus performantes que celles au meilleur d'abord en ce qui concerne la rapidité d'obtention de la meilleure solution. La méthode présentant le plus court temps de réponse est une méthode guidée par H_W et rangée de manière ordonnée au meilleur d'abord guidée par g. On peut donc en conclure que l'élagage effectué avec l'aide de la méthode H_W est efficace. En ce qui concerne la qualité de la solution, la méthode H_r donne les meilleurs résultats dans la plupart des cas. On en conclut donc que cette méthode guide bien la recherche vers les solutions les meilleures, même si elle n'est pas efficace pour élaguer l'arbre de recherche.

Etudions par ailleurs l'amélioration des plans en fonction de la durée de calcul pour chaque scénario significatif. En effet, les méthodes mal classées en terme de temps de réponse ont peut-être trouvé un plan de qualité équivalente aux autres en un temps de réponse équivalent. Les scénarios significatifs sont Mission 2-M3 et Mission 2-F3. Les figures 4.24 et 4.25 donnent l'amélioration des plans pour chaque méthode en fonction de la durée de calcul.

La figure 4.26 donne un zoom de la figure 4.25 centré sur le début des courbes.

Il est clair que les méthodes de rangement en profondeur ordonnée (R_1 et R_2) donnent les meilleurs résultats : la qualité des solutions est comparable voire souvent meilleure que celle des méthodes de rangement au meilleur d'abord, et les améliorations surviennent plus rapidement.
Les méthodes H_1 et H_i obtiennent des résultats semblables : la prise en compte de l'évaluation du danger dans H_i est négligeable.

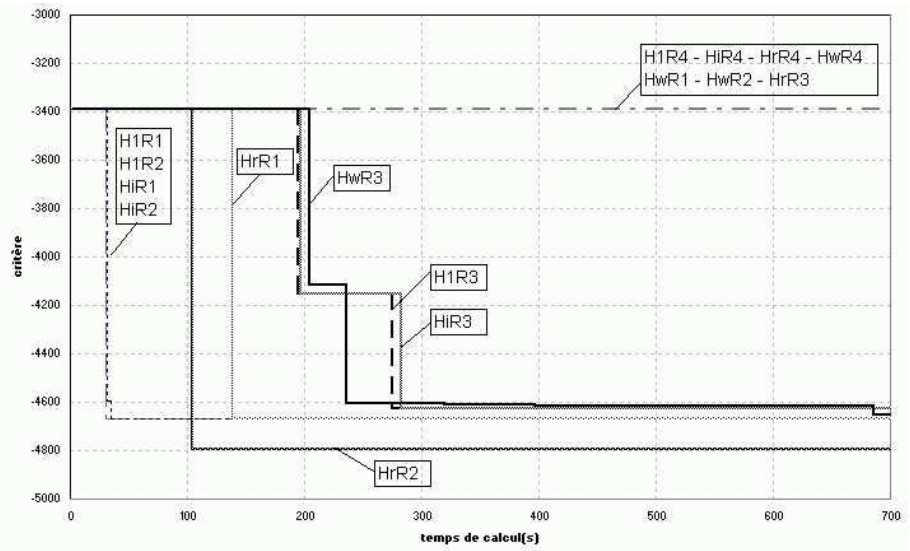


FIG. 4.24 – Evolution du critère en fonction de la durée de calcul - Mission 2-D3

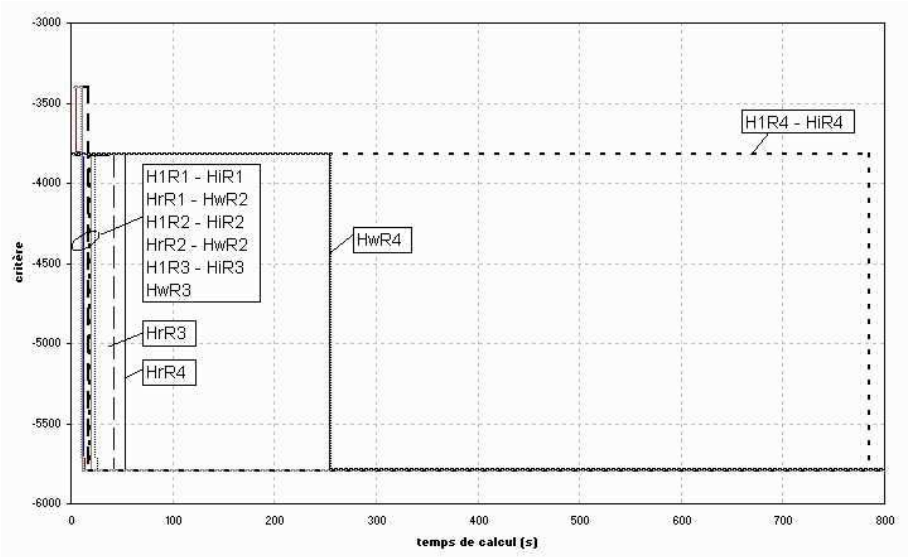


FIG. 4.25 – Evolution du critère en fonction de la durée de calcul - Mission 2-F3

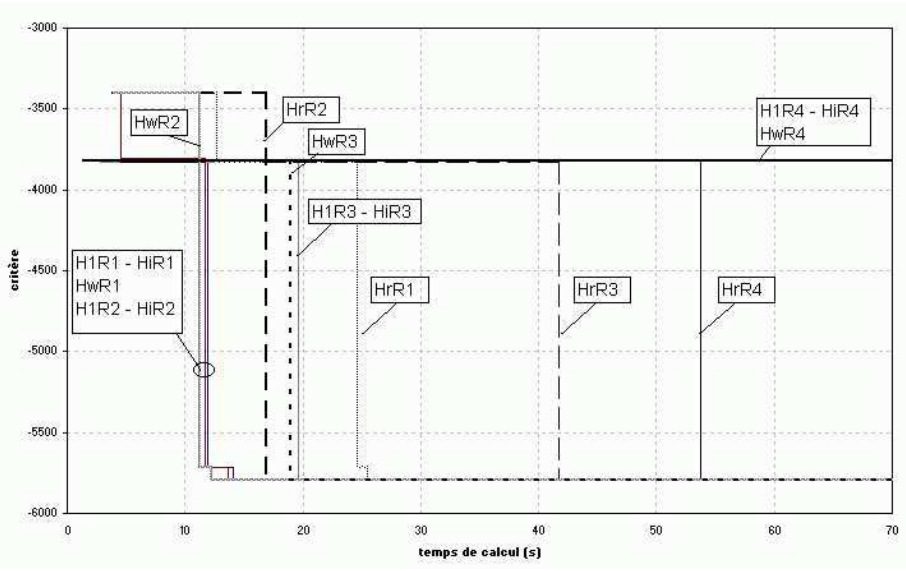


FIG. 4.26 – Evolution du critère en fonction de la durée de calcul entre 0 et 70 s - Mission 2-F3

Au niveau du guidage, on peut avancer que le guidage de H_1 et H_i est peu efficace. En effet, on constate sur les courbes qu'associées à R_1, ces méthodes donnent les mêmes résultats qu'avec R_2. Pire encore, dans certains cas, elles donnent de meilleurs résultats associées avec R_3 qu'avec R_4. Par ailleurs, H_r semble être une bonne méthode de guidage, mais à long terme, c'est-à-dire au-delà de 20 s. En effet, les figures 4.24 et 4.25 montrent qu'associée à R_2, elle obtient de meilleurs résultats qu'avec R_1. On remarque la même chose pour les rangements R_4 etR_3. Enfin, H_W apparaît comme une méthode de guidage peu efficace puisque l'on constate (Figures 4.24 et 4.25) que $H_W R_3$ obtient de meilleurs résultats que $H_W R_4$.
En contrepartie, H_r apparaît comme une mauvaise méthode d'élagage par rapport à H_1 et H_i. En effet, si on compare les courbes de $H_r R_1$ avec celles de $H_1 R_1$, $H_i R_1$ ou $H_W R_1$ sur les figures 4.24 et 4.26, ou encore les courbes de $H_r R_3$ avec celle de $H_1 R_3$, $H_i R_3$ ou $H_W R_3$, on constate qu'associée aux méthodes guidées uniquement par g (R_1 et R_3), H_r donne de plus mauvais résultats que les autres méthodes associées au même rangement.

Pour les replanifications en contexte éventuel, les méthodes de rangement en profondeur ordonnée sont les plus efficaces en terme d'amélioration de la solution. Cependant, il est difficile de trouver une méthode meilleure que les autres en terme de qualité et de durée de calcul. Si on ne considère que la qualité des solutions trouvées, la méthode $H_r R_2$ est la meilleure. Si l'on ne considère que le durée de calcul, la méthode $H_W R_1$ est la meilleure. Il est par conséquent nécessaire d'effectuer une étude multicritère pour ce problème.

Choix d'une méthode d'analyse multicritère

Il existe deux grandes approches pour la décision multicritère. L'approche de l'utilité multi-attribut, appelée aussi utilitarisme, suppose qu'il existe une relation de préférence sur les alternatives. La méthode est la suivante : une fonction d'utilité globale agrège toutes les utilités, puis la valeur de cette utilité globale est comparée pour toutes les alternatives. Le désavantage majeur est que cette solution peut avantager des alternatives extrêmes, c'est ce qu'on a vu pour le cas de replanification éventuelle. Elle convient à des applications où les critères quantitatifs sont mesurables objectivement et commensurables. Ce n'est pas le cas dans notre application où les critères sont des durées en secondes et des termes exprimant une qualité en dollars.

Les méthodes de surclassement posent des hypothèses moins fortes. Le but est d'élaborer des concepts et des algorithmes dans une perspective d'aide à la décision. L'imprécis et l'incertain sont pris en compte grâce au concept de seuils. La notion de surclassement est utilisée pour modéliser l'idée de préférence au sens large. Ces méthodes sont plus adaptées à notre problème. Les méthodes ELECTRE [Roy 1968], [Roy 1991], [Figueira, Mousseau, & Roy 2005] semblent tout à fait convenir : elles permettent de décrire un système de préférences conduisant à la formulation d'une recommandation.

ELECTRE I et ELECTRE IS La méthode ELECTRE I relève de la problématique "procédure de sélection", le problème est posé en terme de choix de la "meilleure" action. La méthode ELECTRE IS est une généralisation de la méthode ELECTRE I. Etant donné un ensemble fini d'actions évaluées sur une famille cohérente de critères quantitatifs ou qualitatifs (pseudo-critères), la méthode a pour objet d'aider à comparer les actions en vue du choix final d'une action ou d'un sous-ensemble d'actions. La méthode agrège les préférences partielles en une relation de surclassement nette qu'elle analyse en termes de graphe. Le sous-ensemble recherché est constitué par le noyau du graphe.

ELECTRE TRI ELECTRE TRI est un outil d'aide multicritère à la décision spécialement conçu pour traiter des problèmes de tri. C'est une méthode intéressante dans la mesure où elle permet une comparaison différente des actions potentielles, non plus entre elles, mais par rapport à une référence stable. Elle permet d'utiliser des valeurs de référence, lorsqu'elles existent.

ELECTRE III et ELECTRE IV La méthode ELECTRE III relève de la problématique "procédure de classement" : son but est de classer les actions potentielles, depuis les "meilleures" jusqu'aux "moins bonnes", en tolérant les ex æquo. Dans un préordre (relation réflexive et transitive), des ex æquo sont possibles. Dans un ordre (relation réflexive, transitive et antisymétrique), il n'y pas d'ex æquo. Un préordre total est un préordre dans lequel les éléments sont toujours comparables (incomparabilité exclue). Un préordre partiel est un préordre dans lequel l'incomparabilité est permise.

Il faut remarquer que dans une procédure de classement, il n'est pas tenu compte de la valeur intrinsèque de chaque action mais seulement de sa valeur relative par rapport aux autres actions. Cette méthode utilise, tout comme la méthode ELECTRE I, la relation de surclassement. Cependant, la distinction est faite entre deux sortes de

surclassements : les surclassements forts qui reposent sur des bases solides et sont donc avancés avec une grande certitude, les surclassements faibles qui concernent ceux des surclassements qui sont sujets à caution.

L'exploitation de ces deux graphes (l'un fort, l'autre faible) s'opère selon un algorithme qui permet de classer les actions. Cet algorithme permet d'obtenir deux classements ou préordre différents. Le premier préordre est obtenu de façon descendante, c'est à dire en sélectionnant tout d'abord les alternatives les meilleures, puis les suivantes, jusqu'au plus mauvaises. Le second préordre est obtenu de façon ascendante, c'est à dire en sélectionnant en premier lieu les alternatives les plus mauvaises pour finir par les meilleures. Ces deux préordres étant le plus souvent différents, c'est leur intersection, un préordre partiel, qui constituera le classement le plus fiable.

Le préordre intersection met en relief les comparaisons entre les alternatives que la méthode a permis d'obtenir et souligne les incomparabilités éventuelles.

L'alternative a sera considérée meilleure que l'alternative b si, dans l'un au moins des deux préordres, a est classée avant b et si, dans l'autre, a est au moins aussi bien classée que b.
L'alternative a sera jugée équivalente à b si les deux alternatives appartiennent à la même classe dans chacun des deux préordres.
Les alternatives a et b seront incomparables si, par exemple, a est en meilleure position que b dans le classement ascendant et si b est meilleur que a à l'issue de la distillation descendante.

Après le déroulement des calculs, l'ensemble des informations est contenu dans un graphe représentant le préordre partiel : toutes les alternatives sont rangées des meilleures aux plus mauvaises. Pour deux alternatives données a et b, quatre cas sont possibles :

1. a est meilleure que b : en terme de graphe, il existe un chemin de a vers b ;
2. b est meilleure que a : il existe un chemin de b vers a ;
3. a et b sont équivalentes ;
4. a et b sont incomparables.

La méthode ELECTRE IV permet la construction de plusieurs relations (emboîtées) de surclassement nettes lorsqu'il n'est pas possible d'affecter un poids à chaque pseudo-critère, ce qui est notre cas pour le contexte de replanification éventuelle. Le décideur doit toutefois admettre qu'aucun critère n'est négligeable ni prépondérant face à un regroupement quelconque d'une moitié des pseudo-critères. On construit, à partir de ces relations emboîtées, une relation de surclassement floue.

C'est cette méthode que nous utiliseront par la suite pour analyser les résultats des tests. On se reportera à [Vallée & Zielniewicz 1994] pour une description complète des moyens et des méthodes utilisés par ELECTRE III et IV.

Analyse multicritère pour le contexte de replanification éventuelle

On considère les deux critères suivants : la qualité généralisée, résultante de la somme des qualités pondérées par les fréquences d'occurrence des évènements, et la durée gé-

néralisée, résultant de la somme des durées de calcul pondérées par les fréquences d'occurrence des événements. Les alternatives sont les 16 méthodes proposées.

Le graphe représentant le préordre partiel pour le contexte de replanification éventuelle est donné figure 4.27. D'après la matrice du préordre final, on constate que les méthodes H_1R_3 et H_rR_4 sont indifférentes, de même pour H_iR_3 et H_WR_3. Enfin, H_1R_4, H_iR_4 et H_WR_2 sont aussi indifférentes. Sur le graphe, l'incomparabilité entre a et b est représentée par une absence de chemin allant de a vers b. Par exemple, on peut dire que H_1R_1 est incomparable avec H_rR_1, H_rR_2, H_WR_1 mais qu'elle est préférée à H_WR_4.

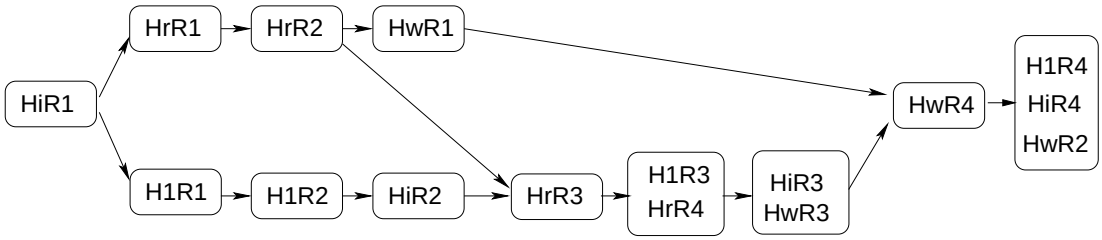


FIG. 4.27 – Graphe représentant le préordre partiel

L'étude multicritère confirme que les méthodes H_rR_2 et H_WR_1, préférées respectivement pour la qualité de la solution, et pour le temps de réponse, ne sont pas les meilleures. La meilleure méthode d'après cette étude est H_iR_1. Elle correspond au meilleur compromis qualité, temps de réponse. Les deux suivantes sont H_rR_1 et H_1R_1. On constate que H_rR_1 et H_1R_1 appartiennent au groupe G_2 défini pour le classement selon le temps de réponse. Cela confirme le fait que les différences entre les durées de calcul influencent le plus la décision.

Les méthodes qui obtiennent le meilleur compromis qualité/temps de réponse sont les méthodes utilisant le rangement R_1 en profondeur ordonnée guidée par g. Pour ce rangement, les méthodes H ne sont utiles que pour l'élagage. Cela confirme le fait que H_1 et H_i sont peu efficaces en guidage. De façon générale, pour ce cas, les méthodes de rangement au meilleur d'abord obtiennent les plus mauvais compromis qualité/temps.

Pour les replanifications en contexte éventuel, l'analyse multicritère permet de trouver les méthodes obtenant le meilleur compromis en terme de rapport qualité/temps de calcul. La meilleure méthode est la méthode H_iR_1. De façon générale, pour ce cas, les méthodes de rangement au meilleur d'abord obtiennent les plus mauvais compromis qualité/temps.

4.3.5 Choix d'une méthode unique pour tous les cas de replanification

L'objectif de ce paragraphe est de trouver une ou plusieurs méthodes qui trouve le meilleur compromis de réponse quel que soit le contexte de replanification. D'après les études effectuées dans les sections précédentes, aucune méthode ne se dégage comme la meilleure méthode. Il est donc nécessaire d'effectuer une analyse multicritère pour effectuer un choix. On choisit d'effectuer l'étude avec ELECTRE IV.

On considère les critères suivants :
- la qualité généralisée de la première solution pour les cas de contexte critique ;
- la qualité généralisée de la meilleure solution pour les cas de contexte critique ;
- la qualité généralisée de la première solution pour les cas de contexte nécessaire ;
- la qualité généralisée de la meilleure solution pour les cas de contexte nécessaire ;
- la qualité généralisée de la meilleure solution pour les cas de contexte éventuel ;
- la durée de calcul généralisée pour l'obtention de la meilleure solution pour les cas de contexte éventuel ;

Les alternatives considérées sont les méthodes algorithmiques.

Le graphe représentant le préordre partiel est présenté figure 4.28.

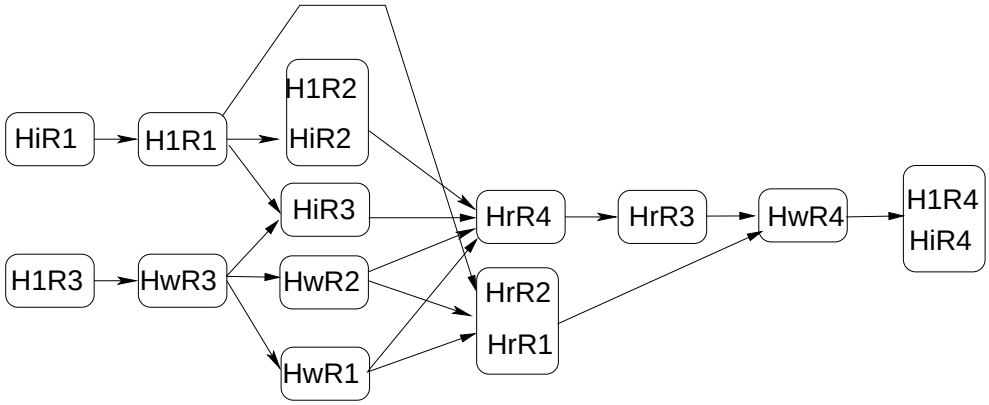


FIG. 4.28 – Graphe représentant le préordre partiel de l'analyse du problème global

Les méthodes qui semblent être les meilleurs compromis pour tous les cas de replanification sont les méthodes H_1R_3 et H_iR_1. Etudions leurs résultats.

La méthode H_1R_3 fait partie du premier groupe pour la qualité des solutions en réponse de type critique. Pour rappel, ce groupe permet de toujours calculer une solution menant à un risque de destruction de l'appareil inférieur à 18%. Dans 95% des cas, H_1R_3 calcule un plan pour lequel le risque de destruction est inférieur à 0,01%. Elle fait partie du deuxième groupe de méthodes dans le classement pour les réponses de type nécessaire. Elle améliore la qualité du plan de manière significative (en moyenne 85%) entre la première solution et la meilleure solution en durée de calcul limitée. En ce qui concerne le cas de replanification éventuelle, H_1R_3 fait partie d'un groupe de méthodes qui calculent des plans de bonne qualité. Dans l'analyse multicritère, on constate que, mis à part H_rR_3, elle est préférée à toutes les autres méthodes utilisant un rangement au meilleur d'abord.

La méthode H_iR_1 fait partie du troisième groupe pour le cas critique. Pour rappel, cette méthode calcule des solutions menant à un risque de destruction de l'appareil inférieur à 50% dans tous les cas. Néanmoins, elle permet de limiter ce risque à 0,01% dans environ 90% des cas. Sur la totalité des tests, les plans s'améliorent en moyenne de 26%. Pour le cas de replanification en contexte nécessaire, H_iR_1 appartient au premier groupe de méthodes. L'amélioration des plans n'est pas homogène, elle va de 0,3% à plus de 380%.

Pour le cas de replanification en contexte éventuel, H_iR_1 fait partie du second groupe. Elle donne sa meilleure solution en environ 10 s. Dans tous les cas, le meilleur résultat a moins de 10% de différence avec le meilleur résultat obtenu sur l'ensemble des méthodes. Concernant l'étude multicritère, H_iR_1 constitue le meilleur compromis entre la durée de calcul et la qualité de la solution. Elle domine toutes les autres méthodes.

En conclusion, on préférera H_1R_3 pour des missions où il est plus probable de replanifier en cas critique qu'en cas éventuel puisqu'elle mène à un risque de destruction moindre. A l'inverse, on préférera H_iR_1 pour des missions où il est plus probable de replanifier en cas éventuel ou nécessaire.

Les conclusions avancées dans les sections précédentes sont confirmées :

- Les méthodes de recherche guidées par g seul sont meilleurs que celle guidées par $g + h$: R_1 et R_3 dominent R_2 et R_4 ;
- La méthode H_r associée à R_3 donne le plus souvent les plus mauvais résultats ;
- Les méthodes d'évaluation "rapides", H_1, H_i et H_W donnent d'excellents résultats.

Les méthodes dominantes sont H_1R_3 et H_iR_1. Elles obtiennent un excellent compromis entre la durée de calcul et la qualité des solutions. Cependant, la méthode H_iR_1 calcule des solutions menant à un risque de destruction de l'appareil limité uniquement à 50% contre 18% pour la méthode H_1R_3. C'est un critère qui n'a pas été pris en compte dans l'analyse multicritère. Pour réduire les risques d'échecs, il est donc préférable d'utiliser la méthode H_1R_3.

4.4 Conclusions

L'analyse des tests a permis de dégager un certain nombre de principes.

Dans le cadre d'un choix de rangement pour chacun des contextes de replanification. On préférera prendre un rangement au meilleur g d'abord (R_3) pour les replanifications critiques, alors qu'on préférera un rangement en profondeur ordonnée (R_1 ou R_2) pour les replanifications de type nécessaire. En replanification de type éventuelle, il semble que les meilleurs compromis durée de calcul/qualité soient obtenus pour les rangements sans évaluation (R_1 ou R_3).

Au niveau des méthodes d'évaluation des coûts du nœud courant à un nœud fin, on constate que l'évaluation des récompenses non obtenues avec ou sans ressources (H_1 et H_i) sont des méthodes équivalentes. La prise en compte du danger dans H_i est négligeable.

Pour le **guidage**, les rangements basés sur H_1 et H_i sont peu efficaces. Les rangements basés sur H_r sont efficace uniquement en contexte éventuel.

Pour l'**élagage**, les méthodes dites simples H_1, H_i et H_W sont les plus efficaces. Par contre l'élagage effectué par H_r semble être de mauvaise qualité.

En conclusion, il semble clair qu'il faut **dissocier la méthode de guidage de la méthode d'élagage**. On pourra par exemple tester un rangement au meilleur $g+h$ d'abord guidé par H_r et élaguer par H_W, H_1 ou H_i. Il faudra le comparer à un rangement au meilleur g d'abord guidé par une méthode simple (H_1, H_i ou H_W).

5 Conclusions et Perspectives

Conclusions

Le domaine de recherche sur les véhicules aériens autonomes intelligents connaît actuellement un rapide développement et offre un grand nombre de défis en ce qui concerne la planification de mission.

Le but de cette thèse est de résoudre le problème de planification de mission pour un véhicule aérien autonome. Ce problème s'énonce ainsi :

> Un véhicule autonome doit choisir et ordonner un sous-ensemble d'objectifs à réaliser parmi tous les objectifs de sa mission. Il doit optimiser le choix de ses actions, connaissant ses ressources, l'environnement, la récompense maximum associée à chaque objectif et les contraintes temporelles qui y sont associées.

Ce problème présente de nombreuses difficultés. Le chapitre 1 dégage dans un premier temps les modèles et méthodes de planification relatifs au problème pour en tirer des idées de résolution. Les problèmes de recherche de chemins mettent à jour la nécessité de **discrétiser l'espace géométrique**. Sur cette discrétisation, il est alors possible de développer un graphe et d'optimiser la longueur d'un chemin en suivant une arborescence décrivant les chemins possibles. L'optimisation dans les graphes nous apprend l'importance de l'utilisation d'**algorithmes informés**, comme l'algorithme A^*. Cependant, les algorithmes de la littérature sont inutilisables tels quels. En effet, ils se basent très souvent sur la propriété de l'inégalité triangulaire qui n'est pas une propriété du problème de planification de mission. Néanmoins, des problèmes comme le voyageur de commerce international ont de grandes similitudes avec le problème de planification de mission, et leur résolution en deux sous-problèmes est à exploiter. Le formalisme état-action permet d'utiliser l'approche logique dans un cadre dynamique. Les problèmes de sélection de la littérature permettent de dégager les aspects non classiques du problème : **les heuristiques classiques doivent être adaptées**, car le nombre d'objectifs n'est pas fixé *a priori*, l'**adaptation de la fonction d'élagage** doit aussi être étudiée à cause des variations non monotones du critère lors de l'exploration de l'arborescence. Enfin, l'étude des Processus Décisionnels de Markov souligne la **difficulté de prendre en compte les incertitudes pour des problèmes de taille réelle**.

Sans être directement applicables, les modèles et méthodes de planification existants donnent une excellente base de travail pour formaliser et résoudre le problème de planification de mission pour un véhicule aérien autonome.

Le premier apport de cette thèse est la **proposition d'un cadre formel pour le problème de planification de mission** (Chapitre 1). L'idée est de tirer parti du **principe d'abstraction** pour organiser la description du problème dans une hiérarchie à deux niveaux. Le haut niveau décrit la réalisation des objectifs : il est générique vis à vis des problèmes de gestion de mission. Le bas niveau décrit la mise en œuvre des actions dans le temps et l'utilisation des ressources : il dépend de l'application. Ce formalisme permet de décrire un **problème où le nombre d'objectifs dans le plan n'est pas fixé** *a priori*. Les incertitudes sur le déroulement du plan sont intégrées dans le critère d'optimisation. L'application de ce formalisme à un exemple de mission militaire d'observation pour un véhicule aérien autonome illustre le processus de réflexion à suivre pour résoudre un tel problème :

1. discrétisation de l'espace ;
2. description haut niveau : quelles sont les étapes nécessaires à la réalisation d'un objectif de la mission ?
3. description bas niveau : quelles sont les ressources ? Quel est le critère ? Quelles sont les contraintes ? Quelles sont les actions possibles ?

Il est possible de suivre cette méthode pour formaliser un problème de planification de mission pour un grand nombre d'applications.

Ce travail de thèse propose un cadre formel pour les problèmes de planification de mission. Le formalisme proposé est basé sur la décomposition du problème en deux niveaux : le haut niveau correspond à la réalisation des objectifs de la mission, le bas niveau décrit cette réalisation en fonction du temps et des ressources. Il permet de formaliser un problème avec incertitudes et où le nombre d'objectifs du plan n'est pas fixé *a priori*.

Le deuxième apport de cette thèse est le développement d'**algorithmes de planification, adaptés au problème de planification de mission et à la replanification** (Chapitre 3). Le cadre algorithmique est basé sur l'algorithme A^*, adapté aux spécificités du problème : optimisation du critère à chaque développement de nœud, rendue obligatoire par la nature non monotone de critère, prise en compte de contraintes non linéaires, dues aux ressources, et de contraintes linéaires sur les durées, optimisation d'un critère non linéaire prenant en compte des récompenses et des coûts dépendant du mode de réalisation des objectifs. Des **méthodes dédiées aux spécificités du problème** sont proposées : nouvelles méthodes d'évaluation du coût d'un nœud courant à un nœud fin et méthodes d'élagage spécifiques. Enfin, différentes méthodes de rangement de la liste des nœuds pendants sont proposées.

La résolution du problème de planification est rendue possible grâce au cadre algorithmique proposé dans ce travail. Basé sur le A^*, l'algorithme proposé est adapté aux spécificités du problème : il optimise un critère non linéaire par rapport aux durées prenant en compte des récompenses et des coûts dépendant du mode de réalisation des objectifs. Différentes méthodes d'évaluation de coût permettent d'informer la recherche, des méthodes d'élagage dédiées visent à améliorer l'efficacité du calcul. Différentes méthodes de rangement sont aussi envisagées.

Le troisième point important de cette thèse concerne l'**intégration de la planification dans une architecture embarquée** (Chapitre 3). Ce travail est nécessaire aux vues des exigences d'autonomie visées. Dans la première partie du chapitre, les liens entre planification et exécution sont étudiés. Il est en effet crucial de bien positionner notre problème sur les échelles existantes de définition de l'autonomie d'un système. Ce travail vise le niveau d'autonomie appelée **replanification embarquée** : le véhicule est totalement autonome, mais n'a pas la capacité de coordonner ses décisions avec d'autres engins. Un tel niveau d'autonomie dans un environnement où les occurrences des événements sont imprévisibles suppose un **calcul de plan en ligne sur événement, appelé replanification**, à opposer aux méthodes de calcul de plan hors ligne, effectuées avant l'exécution de la mission. Le concept de replanification implique l'adaptation des architectures embarquées pour prendre en compte ce module. L'étude des diverses architectures existantes montre l'importance de développer une **architecture hybride, incluant des éléments délibératifs et réactifs**, permettant le contrôle des tâches décisionnelles et pouvant effectuer un suivi de situation.

Le niveau d'autonomie décisionnelle visé est celui de la replanification embarquée. Il nécessite un calcul de plan en ligne, appelé replanification. La mise à jour du plan en ligne implique le développement d'une architecture hybride, intégrant le déclenchement de calculs de nouveaux plans en cas d'aléa et la prise en compte du résultat de ce calcul.

Comme le calcul de plan s'effectue en ligne sur événement, il est crucial de déterminer comment et pourquoi un tel calcul est déclenché et comment le résultat est appliqué à l'exécution. L'architecture embarquée d'un système muni d'une telle autonomie décisionnelle va devoir d'une part **permettre au système d'effectuer sa mission**, et d'autre **prendre en compte les aléas, dont les instants d'occurrence sont par définition imprévisibles**. L'architecture hybride proposée comporte quatre niveaux hiérarchisés, tous inspirés du principe d'asservissement en boucle fermée de l'Automatique. On propose d'utiliser le logiciel ProCoSA pour concevoir et activer l'architecture. Le logiciel est utilisé à la fois pour **organiser les tâches de raisonnement** et **suivre les tâches physiques**. L'intégration du calcul de plan dans l'architecture met à jour l'importance de la détermination de la **criticité temporelle du calcul de replanification**. Ceci fait l'objet de la dernière partie du chapitre 3. La solution proposée dans cette thèse est d'évaluer le plan courant dans le nouveau contexte et d'en tirer des conclusions sur la criticité temporelle du calcul de replanification.

L'architecture proposée permet de mettre en place une planification en ligne sur événement. Elle est hiérarchisée sur quatre niveaux (de la gestion de la mission jusqu'au guidage), tous inspirés du principe d'asservissement en boucle fermée de l'Automatique. L'activation et le contrôle des tâches de raisonnement et le suivi de la situation des tâches physiques s'effectue grâce au logiciel ProCoSA. On propose enfin de juger de la criticité temporelle du calcul de replanification par l'évaluation du plan courant dans le nouveau contexte.

La dernière partie de ce manuscrit (Chapitre 5) propose des tests effectués sur ordinateur. Les scénarios sont des missions militaires d'observation pour un drone MALE en présence de dangers pouvant apparaître en cours de mission. 36 scénarios sont testés sur les 16 combinaisons de méthodes possibles. L'analyse des tests permet de dégager les performances des algorithmes en fonction de critères découlant de la méthode de détermination de la criticité temporelle du calcul de replanification. La première conclusion est que le cadre algorithmique proposé permet de résoudre le problème, souvent en temps limité. Les analyses précises dégagent deux conclusions : il faut **dissocier les méthodes d'élagage et de guidage**. Enfin, la méthode la plus proche du formalisme développé, H_W, donne de très bons résultats en terme de temps de calcul : elle est adaptée au contexte de replanification critique.

Les tests effectués montrent que les méthodes développées permettent de résoudre le problème de planification de mission, souvent en temps limité. Les analyses permettent de dégager deux principes importants :
- il faut dissocier les méthodes d'élagage et de guidage ;
- la méthode la plus proche du formalisme développé, H_W semble adaptée au contexte de replanification critique.

Ce travail propose donc un cadre formel et algorithmique pour formaliser et résoudre le problème de planification de mission et enfin embarquer cette solution dans un véhicule autonome grâce à une architecture hiérarchique.

Discussion

Pour autant, ce travail ne constitue qu'une partie des recherches sur les véhicules aériens autonomes intelligents. Tout au long de ce manuscrit, des hypothèses sur le problème ou sa résolution sont posées. Il serait intéressant de les remettre en cause pour des développements à venir.

L'hypothèse la plus importante de ce travail de thèse est que la planification ne prévoit pas de replanification dans le plan (Section 1.3.1). Lever cette hypothèse revient à calculer des plans partiels et replanifier en ligne pour l'état courant. Une autre solution pourrait être de développer une planification d'anticipation comme dans Propice-Plan.

La deuxième hypothèse est que “la vitesse est constante entre deux nœuds” (Section 1.4.4). Cette hypothèse implique une discontinuité sur une variable qui de toute évidence ne peut qu'être que continue pour des applications réelles. On peut néanmoins espérer qu'entre deux nœuds, cette vitesse de consigne moyenne sera respectée et que les valeurs de consommation et de durée seront respectées en moyenne. Dans tous les cas, les boucles de contrôle automatique développées dans l'architecture devraient permettre de corriger en ligne ces décalages dus à la différence entre le modèle et la réalité.
Etudions les conséquences si on ne posait pas les hypothèses énoncées avec la formalisation du problème (Section 1.4.4).
Lever l'hypothèse 3 revient à dire que les fonctions de récompense sur les objectifs sont infinies. Ceci a pour conséquence de mener à des itinéraires de même valeur de critère quelles que soient les actions choisies. Ces itinéraires de critère infiniment négatif peuvent s'affranchir des contraintes sur les ressources non décomposables ou décomposables dont la fonction de coût est non linéaire, puisque ces contraintes sont prises en compte par des fonctions de pénalisation dans le critère. Un objectif avec une fonction de récompense infinie signifie physiquement que cet objectif est si important que quoi qu'il se passe, il faut que le système le réalise.
L'hypothèse 4 suppose que la fonction de coût des ressources est décroissante avec les ressources. Cette hypothèse sert à définir un coût partiel, utilisé dans la résolution. Elle ne peut pas être levée.
L'hypothèse 5 suppose que toutes les ressources sont consommables et non renouvelables. Imaginons maintenant une ressource réutilisable (à opposer avec une ressource consommable). Il n'y a pas lieu de minimiser le coût relatif à cette ressource. Par contre une ressource réutilisable mais limitée implique une contrainte, puisqu'on ne peut pas utiliser plus que ce que l'on a à disposition. Dans notre problème, cela signifie que pour un tel type de contrainte, l'inégalité 1.5 doit être non seulement vérifiée au nœud n_e, mais aussi pour tous les nœuds de l'itinéraire. Par contre le coût relatif à la variable est supprimé. La résolution du problème reste identique. En ce qui concerne les ressources renouvelables, dont la quantité reste constante quelle que soit l'utilisation de la ressource, cela revient dans notre problème à considérer une ressource illimitée : il n'y a aucune contrainte dessus, ni aucun avantage à minimiser son utilisation. On pourrait éventuellement penser à des ressources disponibles seulement sur une fenêtre temporelle ou à des ressources exclusives (de type capteur/émetteur par exemple). Les contraintes sur ces ressources pourraient être prises en compte dans l'espace de définition de l'ensemble des actions possibles entre deux nœuds. Cet espace de définition serait alors fonction de l'utilisation des ressources jusqu'au nœud courant et du temps.

Perspectives

Les algorithmes de planification

Le fonctionnement des algorithmes de planification est dépendant d'un certain nombre de paramètres de réglage fixés par l'utilisateur qui les détermine de manière empirique. Ces paramètres sont les suivants :

- le nombre maximum cpt_{max} de nœuds développés lors de la recherche d'une première solution admissible (Section 2.3.2) ;
- la vitesse par défaut du système le long de l'itinéraire lorsqu'il n'y a pas d'optimisation des vitesses (Section 2.3.2) ;
- lors de l'optimisation du critère (Section 2.3.4), les deux tests d'arrêt utilisés : nombre d'itérations maximum possible I_{max} et précision ϵ pour considérer que l'optimum est atteint ;
- le nombre de nœuds développés et vitesse fixée lors de la méthode H_r (Section 2.4.1), identiques par défaut aux valeurs choisies pour la recherche d'une première solution ;
- la valeur du γ pour le second élagage proposé (Section 2.4.2).

Une solution envisageable pour déterminer ces paramètres de manière automatique est l'utilisation de méthodes d'apprentissage ou de règles. Les paramètres doivent varier en fonction de la taille du problème de planification (notamment le nombre d'objectifs à réaliser), et en fonction des valeurs des récompenses et des coûts, qui influencent l'ordre de grandeur du critère à optimiser. Il faut aussi envisager une étude de sensibilité de la solution à ces paramètres, aussi bien en terme de temps de calcul qu'en terme de qualité de la solution. Une première étude montre que le nombre maximum de nœuds développés lors de la recherche d'une première solution a de l'influence si le problème est très contraint, car tous les itinéraires proposés seront inadmissibles : développer un nombre de nœuds trop grand augmente considérablement le temps de calcul sans donner de résultat. L'étude des variations de γ sur quelques exemples montre que ce paramètre a une influence sur la rapidité des calculs et sur la qualité de la solution. Sa valeur doit dépendre d'une étude sur l'efficacité des méthodes d'évaluation du critère d'un nœud courant à un nœud fin.

Plusieurs améliorations sont envisagées en ce qui concerne les méthodes. La première touche la méthode d'évaluation H_r (Section 2.4.1). En effet, il faudrait, comme pour la méthode de recherche d'une première solution admissible, optimiser les vitesses en prenant en compte toutes les contraintes du problème pour les meilleures séquences trouvées. Ceci améliorerait considérablement l'efficacité de cette méthode car elle prendrait en compte les coûts, les contraintes temporelles et l'utilisation des ressources pour guider la recherche.
La deuxième amélioration vise à dissocier les méthodes h du guidage et celles utilisées pour l'élagage. Il faut tester différentes combinaisons de ces méthodes sur un large panel de mission de façon à étudier leur comportement.
Enfin, pour répondre au problème de limitation forte du temps de calcul, une perspective proposée est de développer des plans partiels, éventuellement en utilisant un développement en largeur d'abord. Cette solution permettrait d'exécuter le meilleur plan partiel. Une autre solution consiste à développer précisément le début d'un plan, et d'effecteur une évaluation plus grossière de la fin du plan, pour exécuter un plan "possiblement le meilleur" par la suite. C'est ce qui se passe si l'on interrompt l'algorithme de base (Algorithme 3) lorsqu'il n'a pas encore trouvé de plan allant jusqu'à un nœud but. Si les méthodes de rangement sont en $g + h$, le premier élément de la liste des pendants permet d'exécuter le meilleur plan en prenant en compte une évaluation de la suite du plan.

En ce qui concerne le choix de la méthode ou des méthodes à embarquer, une perspective intéressante serait de développer une fonction permettant de choisir automatiquement la meilleure méthode à appliquer pour résoudre le problème. Ce choix devrait s'effectuer en fonction de la connaissance de la mission et de la criticité du calcul de replanification. La fonction de choix pourrait utiliser une méthode d'apprentissage, des méthodes floues, ou plus simplement des règles de choix.

Enfin, l'algorithme et les méthodes proposés permettent de résoudre un problème hybride de choix de points de passage et d'optimisation d'un critère continu sous contraintes. Il serait intéressant d'envisager l'application de cet algorithme à d'autres problèmes que la planification de mission pour un véhicule autonome. Une proposition est d'imaginer un problème d'organisation du temps et de l'utilisation des machines de production dans une usine. Le problème s'énonce ainsi :

Un contremaître doit faire usiner différentes pièces et les vendre à ses clients, sachant que :

- l'usinage de chaque pièce peut s'effectuer sur des machines différentes, il faut donc choisir la machine qui va usiner chaque pièce ;
- les pièces sont usinées les unes après les autres ;
- toutes les pièces ne sont pas forcément usinées.

Le contremaître doit choisir un sous-ensemble de pièces à usiner, ordonner leur fabrication et optimiser son choix en optimisant un critère qui comprend : le coût de l'utilisation des machines, les profits dus à la vente des pièces.
Les contraintes sont dues :

- au temps total qui est imparti au contremaître pour usiner et vendre ses pièces ;
- aux fenêtres temporelles où les machines sont disponibles.

Le coût de l'utilisation des machines peut varier en fonction de leur date d'utilisation (jour/nuit), de leur durée d'utilisation.

En effectuant l'analogie suivante, "usinage de pièce $\longleftrightarrow$ objectif", "plusieurs machines possible pour usiner une pièce $\longleftrightarrow$ actions", et en considérant chaque machine comme une ressource, ce problème peut se formaliser et être résolu par nos algorithmes. On imagine sans mal que d'autres problèmes peuvent être envisagés comme des problèmes de planification de mission.

Les perspectives sur les algorithmes de planification portent sur :
- l'automatisation du choix des paramètres des algorithmes ;
- l'amélioration des méthodes algorithmiques ;
- l'automatisation du choix de la meilleure méthode à utiliser pour résoudre le problème selon le contexte de planification ;
- l'application des algorithmes sur des domaines d'application variés.

L'architecture

Le travail le plus important dans l'avenir est de mettre en œuvre l'architecture pour tester la prise en compte et le traitement des aléas. Ainsi, un premier travail concernant le développement d'un banc test d'un véhicule aérien autonome sous ProCoSA a été effectué.

Des travaux futurs sont envisageables en ce qui concerne le haut niveau de décision dans l'architecture. La connaissance opérateur pourrait ainsi être mise à contribution. En effet, l'opérateur peut ne pas se limiter à donner des modifications sur les cartes des objectifs et des dangers. Il pourrait aller jusqu'à donner quelques informations sur les graphes de haut et bas niveaux, par exemple des objectifs à effectuer en suivant, ou des actions déjà choisies : ainsi le processus de planification aurait plus de facilité à trouver un plan. Par exemple, si l'opérateur sait qu'il faut effectuer l'objectif $O3$ puis tout de suite après réaliser $O6$, il va l'indiquer dans la structure du graphe de haut niveau en ne mettant en sortie de $W_{r(3)}$ que l'ensemble $W_{s(6)}$. De même, si l'opérateur sait que pour aller de la sortie de $O2$ à l'entrée de $O4$ la meilleure action est la ligne droite, il pourra indiquer que l'ensemble $A_{2,4}$ contient uniquement l'action de ligne droite.

Un problème important lors du lancement du calcul de planification est la détermination du nœud d'application de la planification, et toutes les données qui lui sont rattachées : ressources r_1, masse du véhicule m_1, position. Une solution pour le système serait de se mettre en position d'attente si le véhicule n'est pas en danger immédiat. Il suffirait alors de prévoir l'utilisation des ressources, notamment la consommation en carburant jusqu'à la date prévue d'obtention du plan. Cette date serait calculée en fonction du contexte de planification.

Une autre perspective intéressante porte sur des travaux récents concernant la prise en compte de l'accumulation des aléas pendant la replanification. La solution envisagée se base sur un classement des aléas selon leur type. Quatre types d'aléas sont étudiés : des aléas conduisant à l'abandon de la mission, des aléas touchant la sécurité du véhicule, des aléas modifiant les objectifs de la mission et enfin des aléas menant à l'autonomie totale du véhicule (perte de tous les moyens de communication). Des règles d'agrégation sont proposées et des traitements adaptés sont envisagés. Le point intéressant est que si un aléa nécessitant une replanification survient alors qu'un calcul de plan est en cours, la replanification est relancée en tenant compte des deux aléas à traiter.

Enfin, l'architecture proposée présente une structure hiérarchisée générique à toutes les architectures des systèmes d'autonomie de niveau 4 dans la classification [OSD 2002]. Elle pourrait donc être appliquée à divers systèmes autonomes, tels que des robots terrestres, des satellites, ou des engins sous-marins.

Les perspectives sur l'architecture portent sur :
- la mise en œuvre de l'architecture ;
- l'utilisation des connaissances opérateur ;
- la prise en compte de l'accumulation des aléas pendant la replanification ;
- l'utilisation de l'architecture pour d'autres systèmes autonomes.

Les tests

En ce qui concerne les tests, l'idée principale serait de développer un générateur automatique de mission pour compléter l'ensemble des tests déjà réalisés. Ce générateur devrait prendre en compte les propriétés des problèmes traités, notamment :

- les points géographiques représentant les points d'entrée de la zone ennemie forment une frontière de part et d'autre de laquelle se situent d'une part les points de début et de fin de mission, d'autre part les points objectif et les zones de danger ;
- les points d'entrée et de sortie des zones objectif sont globalement concentrés ;
- les contraintes temporelles entre les points d'entrée et de sortie de la zone ennemie, et celles sur les objectifs doivent être compatibles ;
- le nombre de zones objectif constituant l'objet de la mission est limité.

D'autres types de scénarios peuvent aussi être envisagés, par exemple des scénarios civils dont les objectifs sont du type : survoler l'école primaire à l'heure du déjeuner pour voir si les voitures roulent à la vitesse réglementaire, surveiller une route pour détecter du verglas à une certaine heure, ...

Les perspectives sur les tests portent sur :
- le développement d'un générateur automatique de mission ;
- le développement d'autre types de scénarios, notamment civils.

L'application

Le problème résolu pour l'application à un véhicule aérien autonome peut être étendu. Une première perspective vient du fait que le choix des points de passage n'est pas issu d'une analyse automatique de l'espace. Les points sont donnés *a priori* par un expert du domaine. L'automatisation de la discrétisation pourrait être effectuée par la technique de choix de points aléatoirement placés dans l'espace aérien, ou bien par une approche tridimensionnelle des graphes de Voronoï.

Une hypothèse du problème écarte tous les aspects portant sur le relief de l'environnement. Dans un cadre plus général, il serait intéressant d'étudier comment prendre en compte cette donnée. Une première piste consiste à dire que le traitement des nœuds dans l'algorithme ne dépend pas de leur composante d'altitude. Ainsi, si une analyse du modèle numérique de terrain permet de fixer des ensembles de nœuds de passage à des altitudes correctes, il n'y a aucun problème pour la résolution. Cette première approche conviendrait pour des engins effectuant leur mission avec un environnement comprenant

des reliefs. Une autre piste consiste à proposer un ensemble d'altitudes possibles pour pouvoir par exemple avoir une meilleure visibilité d'un objectif en cas de nuages. Cette solution implique une multiplication du nombre de nœuds à développer et une explosion rapide du nombre d'itinéraires possibles. Il faudrait donc trouver un compromis entre le nombre de nœuds permettant de réaliser un objectif et le temps de calcul d'un plan. C'est un problème analogue au choix du nombre de points d'entrée et de sortie pour un objectif étendu.

Les transmissions sont *a priori* instantanées. Il faudrait envisager de prendre en compte des transmissions effectuées sur un intervalle de temps ou bien des transmissions partielles. Une solution pourrait être de voir les transmissions comme des zones de transmission, avec un début de transmission et une fin de transmission, tout comme le traitement des objectifs. Une autre modélisation possible des transmissions est de les considérer comme instantanées en imaginant des relais satellites par exemple. Ces transmissions seraient uniquement contraintes par des intervalles de temps où les transmissions seraient interrompues.

Une amélioration pertinente de la modélisation serait d'envisager la dégradation de la récompense en fonction de l'intervalle de temps entre le traitement d'un objectif et la transmission des informations : il est en effet réaliste d'imaginer qu'une fois le gain emmagasiné (traitement d'un objectif terminé), il soit plus avantageux de transmettre au plus tôt. Ainsi l'information donnée est la plus récente possible. Il suffirait pour cela de développer une fonction pondérant la récompense obtenue en fonction de la date de fin de traitement de l'objectif et de la date de transmission des informations.

Enfin, le module de planification actuel ne traite que du problème de l'itinéraire et des actions. Il faudrait par la suite prendre en compte plus précisément la gestion de la charge utile durant la mission. Ainsi pour l'exemple illustratif, on pourrait envisager des limitations en terme de mémoire, des pannes de capteurs ou des choix possibles entre deux capteurs.

Les perspectives sur le problème de mission portent sur :
- l'automatisation du choix des points de passage ;
- le traitement du relief ;
- la levée de l'hypothèse de l'instantanéité des transmissions ;
- la dégradation des récompenses obtenues en fonction de l'intervalle de temps entre le traitement d'un objectif et la transmission des informations ;
- la gestion détaillée de la charge utile.

ANNEXES

A Rappels sur la construction et l'optimisation dans les graphes

A.1 Construction de graphe

Le graphe est construit le plus souvent par une discrétisation de l'environnement. Il existe deux catégories de méthodes [Latombe 1991] pour discrétiser l'espace de recherche. Les méthodes de la première catégorie font appel à des décompositions en cellules, de différents types, qui permettent de reproduire la topologie de l'espace (Figure A.1). Les nœuds du graphe obtenu sont alors des cellules et les arcs correspondent aux frontières entre cellules. Un chemin est alors défini par une suite de cellules adjacentes.

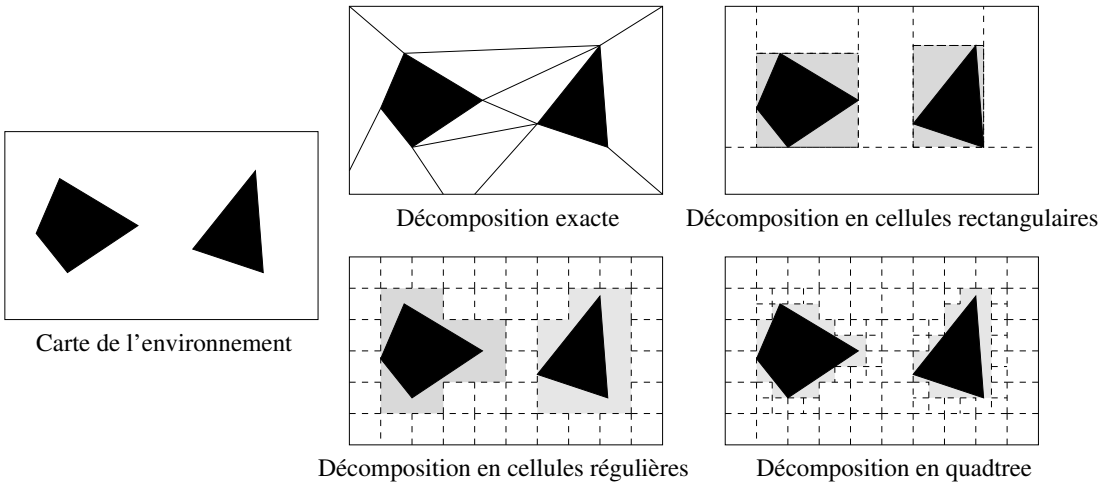


FIG. A.1 – Exemples de décomposition en cellules. La décomposition exacte permet de représenter l'espace libre à l'aide de cellules de formes irrégulières. La décomposition en cellules rectangulaires ne représente qu'un sous-ensemble de l'espace libre suffisant pour la planification. Ces cellules peuvent être de taille régulière ou non. Une représentation hiérarchique de type "quadtree" permet d'utiliser des cellules de taille variable en fonction de la complexité locale de l'environnement.

Les méthodes de la seconde catégorie effectuent un pré-calcul de chemins entre des points répartis dans l'environnement. Les chemins possibles sont donnés par un graphe, dont les nœuds sont les points de l'environnement et les arcs sont les chemins reliant les points.

Le graphe issu du diagramme de Voronoï consiste à découper l'espace en cellules de points équidistants de chaque obstacle. Les déplacements se font alors le long des fron-

tières entre cellules, ce qui permet de générer des chemins passant le plus loin possible des obstacles. En deux dimensions, les frontières sont des courbes correspondant aux arcs du graphe et les points de l'espace à égale distance de trois obstacles sont associés aux nœuds du graphe et donc aux états. Un exemple avec des obstacles ponctuels est donné Figure A.2 : les courbes correspondent au diagramme de Voronoï. Des arcs supplémentaires sont définis pour permettre la connexion du diagramme avec l'origine et les destinations.

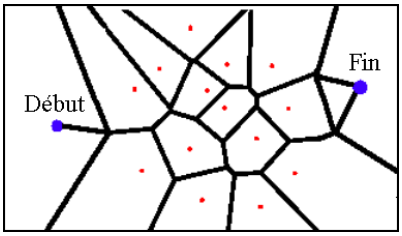


FIG. A.2 – Exemple de graphe de Voronoï

Le graphe de visibilité (Figure A.3) consiste à considérer comme états des points de l'espace tangents aux obstacles ou nœuds d'obstacles, extremums sur l'obstacle à partir de l'origine, d'une des destinations ou d'un point déjà défini.

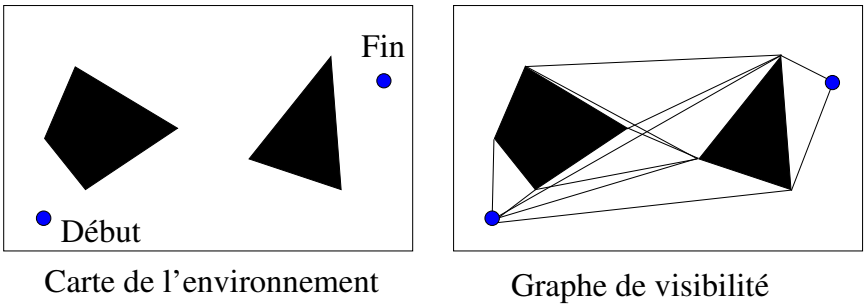


FIG. A.3 – Exemple de graphe de visibilité

Le graphe de Voronoï et le graphe de visibilité sont très répandus en robotique mobile [Blaer & Allen 2003], [Rao & Iyengar 1990], mais aussi pour résoudre des problèmes de missions multi-véhicules [Chandler, Patcher, & Rasmussen 2001], [Beard *et al.* 2002].

A.2 Optimisation dans les graphes

Des présentations générales des algorithmes de chemins figurent dans plusieurs ouvrages de graphes ou d'algorithmique. On pourra se reporter au livre de Gondran et Minoux, *Graphes et Algorithmes*, publié aux éditions Eyrolles pour la première fois en 1979, et dont la dernière édition date de 1995 [Gondran & Minoux 1995].

A.2.1 Définitions

DÉFINITION 17 *Un **graphe orienté** (figure A.4) est défini par le couple (N, A) où :*
- *N est l'ensemble des nœuds ;*
- *A est l'ensemble des arcs du graphe.*

L'ensemble des arcs est un sous-ensemble du produit cartésien $N \times N$. Un arc est une ligne joignant un nœud de départ n_i à un nœud d'arrivée n_j. L'arc (n_i, n_j), s'il existe, est différent de l'arc (n_j, n_i).

DÉFINITION 18 *Pour un arc (n_i, n_j), n_j est le **successeur** de n_i, et n_i est le **prédécesseur** de n_j. On dit que n_i et n_j sont des nœuds **voisins** ou **adjacents**.*

DÉFINITION 19 *Un **chemin** est une séquence d'arc $\{a_1, \ldots, a_k\}$ telle que pour tout i de 1 à $k-1$ les arc a_i et a_{i+1} ont un nœud commun, ce nœud étant de type arrivée pour a_i et de type départ pour a_{i+1}. Dans un graphe orienté, un **circuit** est un chemin dans lequel le nœud d'arrivée de a_k coïncide avec le nœud de départ de a_1.*

DÉFINITION 20 *Un graphe orienté admet une **racine** ou **nœud origine** n_1 si pour tout n_j appartenant à N il existe un chemin de n_1 à n_j.*

DÉFINITION 21 *Une **arborescence** est un graphe orienté admettant un unique nœud origine et dans lequel il n'existe pas deux chemins ayant au moins un nœud commun.*

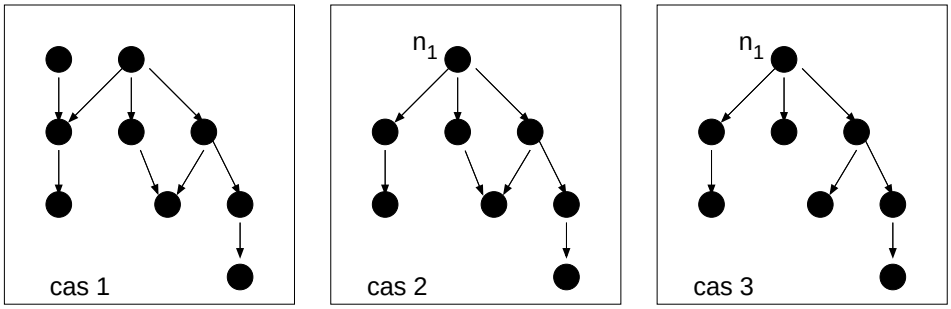


FIG. A.4 – Exemples de graphes orientés - cas 1 : sans nœud origine ; cas 2 : avec un nœud origine n_1 ; cas 3 : arborescence

DÉFINITION 22 *La **profondeur** d'un nœud dans une arborescence est le nombre d'arcs du chemin qui va du nœud origine à ce nœud. Le nœud origine est à la profondeur 1, et la profondeur d'un nœud est égale à la profondeur de son prédécesseur plus 1. Si un nœud est à une profondeur p, tous ses successeurs sont à la profondeur p+1. Tous les nœuds d'une arborescence de même profondeur sont dits au même **niveau**.*

A.2.2 Problèmes de recherche de moindre coût

Ce paragraphe présente les principales idées permettant de construire des algorithmes de recherche de moindre coût ainsi que quelques algorithmes de la littérature.

Les problèmes de recherche de moindre coût visent à trouver le chemin le moins coûteux d'un nœud de départ à un ensemble de nœuds destination. On définit le nœud de départ n_1 comme le nœud origine d'un graphe, et un ensemble de nœuds but ou nœuds destination. Chaque transition possible correspond à un arc du graphe. A chaque transition, on associe un coût (ou poids) : l'arc ayant pour nœud de départ n_i et pour nœud d'arrivée n_j est pondéré par le coût $c(n_i, n_j)$.

☞ Par la suite, on appellera **arborescence** ou **arbre d'exploration** l'arborescence de nœud origine n_1 décrivant les chemins possibles de ce graphe. L'arborescence peut donc contenir plusieurs fois un nœud appartenant à des chemins différents du même graphe.

Pour pouvoir résoudre un tel problème, un algorithme de recherche doit réaliser une exploration d'une manière systématique et contrôlée. On distingue deux classes d'algorithmes de recherche :

- Les algorithmes non-informés ou "aveugles" qui réalisent une recherche exhaustive dans l'arborescence, sans utiliser aucune information concernant la structure du graphe pour optimiser la recherche ;
- Les algorithmes informés ou guidés qui utilisent des informations pour optimiser cette recherche dans l'arbre d'exploration. Ces algorithmes parviennent ainsi à des performances meilleures.

Les algorithmes de recherche de plus court chemin sont des méthodes itératives essentiellement basées sur le principe du déroutement.

Définition 23 *A chaque nœud n_i, on associe un **potentiel** $g_i \in \mathbb{R}$. En cours d'algorithme, les différentes valeurs de g_i représentent des coûts des chemins entre le nœud origine et le nœud n_i considéré. En fin d'algorithme, g_i représente le coût d'un plus court chemin du nœud origine au nœud n_i considéré.*

La figure A.5 illustre le positionnement des variables sur le graphe.

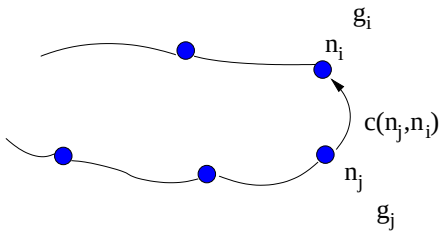


Fig. A.5 – Principe du déroutement

L'**algorithme de Ford-Bellman** se base directement sur l'idée du déroutement. Les potentiels sont initialisés à 0 pour le nœud origine, noté n_1, et à $+\infty$ pour les autres. On effectue des déroutements tant qu'il est possible d'en effectuer. Il est peu structuré dans la mesure où l'ordre de traitement des nœuds est quelconque.

Principe du déroutement

si $g_j + c(n_j, n_i) < g_i$ **alors**
  Déroutement par n_j :
  $g_i \leftarrow g_j + c(n_j, n_i)$
  prédécesseur de $n_i \leftarrow n_j$

Algorithme de Ford-Bellman

début
  Initialisation :
  Marquer chaque nœud n_i avec un potentiel g_i, $g_1 = 0$ et $g_i = +\infty$;
  Amélioration des potentiels :
  tant que *Il existe un arc* (n_j, n_i) *tel que* $g_j + c(n_j, n_i) < g_i$ **faire**
    Dérouter par n_j
fin

Définition 24 *Chaque étape de l'exploration correspond à un* ***développement de nœud*** *: le nœud a été explicitement engendré. On appelle* ***nœud pendant*** *un nœud non développé dont le prédécesseur a été développé. L'ensemble des nœuds pendants est aussi appelé* ***frontière de recherche****.*

Notations : *listeP* est la liste des nœuds pendants ; *listeQ* est la liste des nœuds déjà développés ; $\hat{u}$ est le premier élément de *listeP* et $S(\hat{u})$ est l'ensemble des successeurs de $\hat{u}$ dans le graphe.

Le **parcours en largeur d'abord** part du nœud origine du graphe. Les nœuds sont ensuite développés niveau par niveau et cela jusqu'à ce que tous les nœuds de l'arborescence aient été développés. On développe donc tous les successeurs de n_1, puis tous les successeurs de ses successeurs ...

Le **parcours en profondeur d'abord** part du nœud origine du graphe. Le principe de cet algorithme est de descendre le plus profondément dans l'arborescence avant de se déplacer en largeur. On effectue ainsi un parcours "de haut en bas" et "de gauche à droite". L'inconvénient d'un tel parcours est que l'on risque de s'enfoncer indéfiniment dans l'arborescence sans trouver de nœud destination.

La Figure A.6 illustre ces deux types de parcours de l'arborescence lors de l'exploration.

Algorithme en largeur d'abord

début
- Initialisation :
- Placer n_1 dans *listeP*
- **tant que** *listeP n'est pas vide* **faire**
  - Traitement :
  - Traiter $\hat{u}$ (dans le graphe, effectuer tous les déroutements pour atteindre $\hat{u}$ par exemple)
  - Rangement des suivants dans *listeP* :
  - Mettre les éléments de $\mathrm{S}(\hat{u})$ qui ne sont pas dans *listeP* en queue de *listeP*
  - Mettre $\hat{u}$ dans *listeQ*
  - Effacer $\hat{u}$ de *listeP*

fin

Algorithme en profondeur d'abord

début
- Initialisation :
- Placer n_1 dans *listeP*
- **tant que** *listeP n'est pas vide* **faire**
  - Traitement :
  - Traiter $\hat{u}$ (dans le graphe, effectuer tous les déroutements pour atteindre $\hat{u}$ par exemple)
  - Rangement des suivants dans *listeP* :
  - Mettre les éléments de $\mathrm{S}(\hat{u})$ qui ne sont pas dans *listeP* en tête de *listeP*
  - Mettre $\hat{u}$ dans *listeQ*
  - Effacer $\hat{u}$ de *listeP*

fin

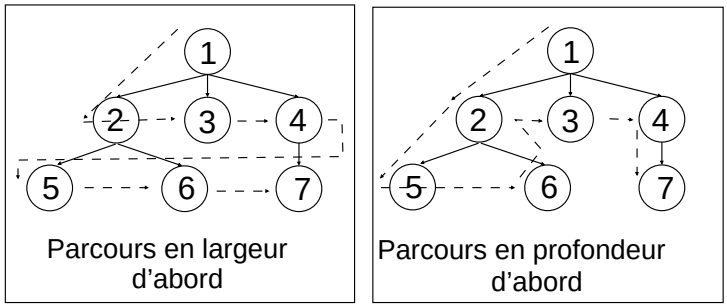


FIG. A.6 – Parcours en largeur d'abord - ordre de parcours : 1, 2, 3, 4, 5, 6, 7 - Parcours en profondeur d'abord, ordre : 1, 2, 5, 6, 3, 4, 7

L'**algorithme de Dijkstra** résout un problème de recherche de plus court chemin pour un graphe avec des coûts positifs ou nuls sur les arcs. Le principe est proche de l'algorithme en profondeur d'abord. L'algorithme part du nœud origine puis calcule le potentiel des nœuds suivants. Il cherche le nœud pendant de meilleur potentiel puis effectue ce même traitement itérativement sur le point courant. L'inconvénient majeur de cet algorithme est que toutes les directions dans l'arborescence sont équivalentes, il n'y a pas d'évaluation des coûts futurs.

Algorithme de Dijkstra

début
- Initialisation :
- Placer n_1 dans $listeP$
- **tant que** $listeP$ *n'est pas vide* **faire**
  - Traitement :
  - Traiter $\hat{u}$ (dans le graphe, effectuer tous les déroutements pour atteindre $\hat{u}$ par exemple)
  - Rangement des suivants dans $listeP$:
  - Mettre les éléments n_i de $\mathrm{S}(\hat{u})$ qui ne sont pas dans $listeP$ en rangeant la liste des nœuds en g_i croissant
  - Mettre $\hat{u}$ dans $listeQ$
  - Effacer $\hat{u}$ de $listeP$

fin

B Algorithme du simplexe

B.1 Définition - Notations

B.1.1 Problème

Le problème linéaire est le suivant :
Trouver les $x_1, x_2, \ldots, x_n$ qui optimisent la fonction :

$$c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

et vérifient les contraintes :

$$\begin{gathered} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\ \cdots \\ a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kn}x_n \leq b_k \\ \cdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m \\ x_1 \geq 0; x_2 \geq 0; \ldots x_n \geq 0 \end{gathered}$$

avec $c_1, \ldots, c_n, a_{11}, \ldots, a_{mn}, b_1, \ldots b_m$ connus.

En notation condensée :

$$\begin{gathered} \min z = \mathbf{C}^\top \mathbf{X} \text{ sous } \mathbf{A}.\mathbf{X} \leq \mathbf{b} \text{ et } \mathbf{X} \geq 0 \\ \text{avec } \mathbf{C}(n \times 1), \mathbf{A}(m \times n) \text{ et } \mathbf{b}(m \times 1) \text{ connus et } \mathbf{X}(n \times 1) \end{gathered} \tag{B.1}$$

B.1.2 Forme standard

La forme standard est un problème d'optimisation sous la forme :

$$\begin{gathered} \min z = \mathbf{C}^\top \mathbf{X} \text{ sous } \mathbf{A}.\mathbf{X} = \mathbf{b} \text{ et } \mathbf{X} \geq 0 \\ \text{avec } \mathbf{C} \text{ et } \mathbf{A} \text{ connus} \end{gathered} \tag{B.2}$$

B.1.3 Mise sous forme standard

On peut toujours remplacer les inégalités :

$$a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kn}x_n \leq b_k$$

par des égalités, en introduisant des *variables d'écart.*

$$a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kn}x_n - e_k = b_k \text{ avec } e_k \text{ positif ou nul.}$$

Le problème devient :

$$\min z' = [\mathbf{C}^\top \quad \mathbf{0}^\top].\begin{bmatrix}\mathbf{X}\\ \mathbf{e}\end{bmatrix} \text{ sous } \quad [\mathbf{A} \quad \mathbf{1}].\begin{bmatrix}\mathbf{X}\\ \mathbf{e}\end{bmatrix} = \mathbf{b} \text{ et } \begin{bmatrix}\mathbf{X}\\ \mathbf{e}\end{bmatrix} \geq 0 \qquad \text{(B.3)}$$
$$\text{avec } \mathbf{C}, \mathbf{A} \text{ et } \mathbf{b} \text{ connus}$$

B.1.4 Définitions

Définition 25 *On appelle* ***solution*** *tout vecteur* $\boldsymbol{X}$ *tel que* $\boldsymbol{A}.\boldsymbol{X} = \boldsymbol{b}$.

Définition 26 *Une solution* $\boldsymbol{X}$ *qui vérifie*

$$\begin{cases} \boldsymbol{A}.\boldsymbol{X} = \boldsymbol{b} \\ \boldsymbol{X} \geq 0 \end{cases}$$

est une solution ***admissible***

Définition 27 $\boldsymbol{A}$ *étant une matrice* $(m \times n)$ *de rang* m. *Une* ***base*** *est une matrice* $\boldsymbol{B}(m \times m)$ *extraite de* $\boldsymbol{A}$ *dont le déterminant est non nul.*

On remarque que le système à résoudre possède m égalités et $n+m$ inconnues. Donc la valeur de n variables **peut être fixées arbitrairement**, par exemple à zéro. On dit qu'elles sont mises **hors base**

Définition 28 *On appelle* ***variables hors base*** *les* n *variables de* $\mathbb{R}^{n+m}$ *fixées à zéros. Les* m *variables restantes sont appelées* ***variables de base***

Définition 29 *On appelle* ***solution de base*** *une solution où en ayant choisi* n *variables hors base, on obtient une solution unique en résolvant les* m *contraintes d'égalité obtenues en ajoutant les variables d'écart.*

Notons x_B les variables de base correspondant aux m variables positives ou nulles et x_H les variables hors base, correspondant aux n variables nulles.

Définition 30 *Une solution de base telle que* $x_B > 0$ *est dite* ***admissible***.

Définition 31 *Si* $x_B \geq 0$, *alors la solution est dite* ***solution de base admissible dégénérée***.

B.2 Résultats fondamentaux

B.2.1 Définitions

DÉFINITION 32 *Soient X et Y deux points d'un domaine K, le segment qui joint X et Y est défini par l'ensemble des U tels que $U = \lambda X + (1-\lambda)Y$ avec $0 \leq \lambda \leq 1$.*

DÉFINITION 33 *Soient X et Y deux points intérieurs de K, le domaine K sera **convexe** si le segment joignant X et Y est dans K.*

DÉFINITION 34 *Une fonction f **est convexe** dans un domaine convexe K si*

$$f(\lambda X + (1-\lambda)Y) \leq \lambda f(X) + (1-\lambda) f(Y)$$

DÉFINITION 35 *U est un **sommet** du domaine K ssi*

$$\begin{cases} U = \lambda X + (1-\lambda)Y \\ 0 < \lambda < 1 \end{cases} \Longrightarrow U = X = Y$$

Autrement dit un point d'un polyèdre est un sommet s'il ne peut pas être vu comme la combinaison linéaire convexe de deux autres points du polyèdre. Tout point d'un polyèdre peut être vu comme une combinaison linéaire convexe des sommets du polyèdre.

B.2.2 Théorèmes

THÉORÈME 1 *S'il existe une solution admissible au problème $\boldsymbol{A.X} = 0, \boldsymbol{X} \geq 0$ alors il existe une solution de base admissible.*
S'il existe une solution optimale admissible, il existe une solution de base admissible optimale

THÉORÈME 2 *L'ensemble des solutions admissibles forme un ensemble convexe K.*

Preuve : Soient $\mathbf{X}$ et $\mathbf{Y}$ deux solutions dans K.

$$\begin{cases} \mathbf{AX} = \mathbf{b} & \mathbf{X} \geq 0 & \mathbf{X} \in K \\ \mathbf{AY} = \mathbf{b} & \mathbf{Y} \geq 0 & \mathbf{Y} \in K \end{cases}$$

Alors :

$$\begin{cases} \lambda \mathbf{AX} = \lambda \mathbf{b} \\ (1-\lambda)\mathbf{AY} = (1-\lambda)\mathbf{b} \end{cases} \Longrightarrow \mathbf{A}[\lambda \mathbf{X} + (1-\lambda)\mathbf{Y}] = \mathbf{b} (\text{en sommant})$$

Soit $\mathbf{U} = \lambda \mathbf{X} + (1-\lambda)\mathbf{Y}$ alors $\mathbf{AU} = \mathbf{b}$ donc $\mathbf{U} \in K$ et $\mathbf{U} \geq 0$.

□

THÉORÈME 3 *Si K est un ensemble convexe, un point de K solution admissible de $\boldsymbol{A.X} = 0, \boldsymbol{X} \geq 0$ est un sommet de K ssi $\boldsymbol{X}$ est une solution de base admissible.*

THÉORÈME 4 *Toute solution de* $\boldsymbol{A}.\boldsymbol{X} = 0, \boldsymbol{X} \geq 0$ *est une combinaison linéaire convexe des sommets. La solution optimale est donc une combinaison linéaire convexe des sommets.*

THÉORÈME 5 *L'optimum de z, s'il existe, est atteint en au moins un sommet de K.*

Preuve : (Par l'absurde)
Soit $\mathbf{X}^*$ un optimum de z qui ne soit pas un sommet de K.

$$\mathbf{C}^\top.\mathbf{X}^* < \mathbf{C}^\top.\mathbf{X}^i \qquad \forall i \text{ tel que } 1 \leq i \leq p$$

$$\Longrightarrow \sum_{i=1}^{p} \lambda_i \mathbf{C}^\top.\mathbf{X}^* < \sum_{i=1}^{p} \lambda_i \mathbf{C}^\top.\mathbf{X}^i$$

$$\text{or } \sum_{i=1}^{p} \lambda_i = 1 \text{ car c'est une somme linéaire convexe}$$

$$\text{D'où } \mathbf{C}^\top.\mathbf{X}^* < \textstyle\sum_{i=1}^{p} \lambda_i \mathbf{C}^\top.\mathbf{X}^i$$

Choisissons les λ_i tels que $\mathbf{X}^* = \sum_{i=1}^{p} \lambda_i \mathbf{C}^\top.\mathbf{X}^i$ (Théorème 4). Donc $\mathbf{C}^\top < \mathbf{C}^\top$ absurde, $\mathbf{X}^*$ est donc un sommet.

□

THÉORÈME 6 *Si z atteint son optimum pour plusieurs sommets, il l'atteint aussi pour toute combinaison linéaire convexe de ces sommets.*

Preuve :

$$z_{opt} = \mathbf{C}^\top\mathbf{X}^1 = \mathbf{C}^\top\mathbf{X}^1 = \ldots = \mathbf{C}^\top\mathbf{X}^p$$

$$\mathbf{U} = \lambda_1\mathbf{X}^1 + \ldots + \lambda_p\mathbf{X}^p$$

$$\text{avec } \sum_{i=1}^{p} \lambda_i = 1 \qquad 0 \leq i \leq 1$$

$$\mathbf{C}^\top\mathbf{U} = \lambda_1\mathbf{C}^\top\mathbf{X}^1 + \ldots + \lambda_p\mathbf{C}^\top\mathbf{X}^p$$

$$\mathbf{C}^\top\mathbf{U} = \sum_{i=1}^{p} \lambda_i z_{opt} = z_{opt}$$

$$\text{D'où } \mathbf{C}^\top\mathbf{U} = z_{opt}$$

□

B.3 Principe

L'algorithme procède de la façon suivante :

1. on recherche un sommet de départ ;
2. on teste si ce sommet est l'optimum ou si la fonction objectif n'est pas bornée inférieurement ; dans ce cas le problème n'a pas de solution finie ;
3. si le sommet que l'on vient d'examiner n'est pas optimal, on se déplace sur un sommet voisin pour lequel la fonction diminue et on repasse à l'étape précédente.

Le nombre de sommets étant fini et tout minimum local étant absolu, le sommet optimal est atteint lorsqu'aucun des sommet voisin ne permet plus de diminution du critère.

B.4 Recherche d'un sommet de départ

B.4.1 Forme simpliciale

Le problème étant sous la forme standard (Equation B.2), il peut se décomposer sous la forme

$$[\mathbf{B} \qquad \mathbf{H}]. \begin{bmatrix} x_B \\ x_H \end{bmatrix} = \mathbf{b}$$

Supposons que $\mathbf{B}$ est inversible, alors

$$[\mathbf{1} \qquad \mathbf{B}^{-1}\mathbf{H}]. \begin{bmatrix} x_B \\ x_H \end{bmatrix} = \mathbf{B}^{-1}\mathbf{b}$$

Si $\mathbf{B}^{-1}\mathbf{b} > 0$ alors la solution

$$x_B = \mathbf{B}^{-1}\mathbf{b} \qquad x_H = \mathbf{0}$$

est une solution de base admissible.

Cette approche nécessite :
- l'inversion d'une matrice ;
- la recherche d'une matrice telle que $\mathbf{B}^{-1}\mathbf{b} > 0$.

La forme simpliciale permet d'éliminer ces inconvénients. En effet, la mise en évidence d'une solution est particulièrement simple si $\mathbf{B} = \mathbf{1}$ et $\mathbf{b} > 0$.

On appellera **forme simpliciale** une expression des contraintes égalités sous la forme

$$[\mathbf{1} \qquad \mathbf{H}]. \begin{bmatrix} x_B \\ x_H \end{bmatrix} = \mathbf{b} \text{ avec } \mathbf{b} \geq 0 \tag{B.4}$$

On a alors une solution directe :

$$\begin{cases} x_B = \mathbf{b} \\ x_H = \mathbf{0} \end{cases}$$

B.4.2 Cas où la forme simpliciale n'est pas évidente

Lorsqu'il n'existe pas de forme simpliciale de départ évidente pour le problème sous forme standard (Equation B.2), on rajoute des variables, dites *artificielles*, pour faire apparaître une telle forme.

Le problème devient :

$$\min w = \sum_{i=1}^{m} y_i \text{ sous } \mathbf{A}.\mathbf{X} + \mathbf{Y} = \mathbf{b} \text{ et } \mathbf{X} \geq 0 \qquad \mathbf{Y} \geq 0 \quad \text{avec } \mathbf{C} \text{ et } \mathbf{A} \text{ connus} \tag{B.5}$$

Pour le problème B.5, on a une forme simpliciale évidente.

$$[\mathbf{1} \qquad \mathbf{A}]. \begin{bmatrix} \mathbf{Y} \\ \mathbf{X} \end{bmatrix} = \mathbf{b} \text{ avec } \mathbf{b} \geq 0$$

La méthode des deux phases permet alors de déterminer une forme simpliciale du problème de départ. Son principe est le suivant :

- On résout le problème B.5 par l'algorithme du simplexe.
- Si la solution de ce problème conduit à une solution avec des variables artificielles non nulles ($w \neq 0$), le problème initial n'a pas de solution.
- Si la solution de ce problème conduit à une solution avec des variables artificielles nulles ($w = 0$), on utilise la solution ainsi obtenue comme sommet de départ pour le problème initial (on supprime les variables artificielles et on a ainsi une forme simpliciale du problème initial).

B.5 Passage d'un sommet à un autre

Dès que l'on quitte un sommet, une des variables hors base correspondantes au moins devient non nulle. Comme on veut de diriger vers un sommet voisin, il ne faut en rendre non nulle qu'une, qui va ainsi devenir variable de base pour le prochain sommet, donc *entrer* dans la base.

B.5.1 Choix de la variable entrante

En exprimant le critère z en fonction des variables hors base. Le critère de sélection de la variable entrante est de prendre la variable hors base qui fournit la plus forte baisse de la fonction objectif. Autrement dit, on choisit celle dont le coût marginal est le plus négatif.

B.5.2 Choix de la variable sortante

En considérant le problème sous forme simpliciale, les contraintes lorsqu'une variable hors base x_r rentre dans la base sont :

$$\begin{aligned} x_1 + a_{1r}x_r &= b_1 \\ &\vdots \\ x_s + a_{sr}x_r &= b_s \\ &\vdots \\ x_m + a_{mr}x_m &= b_m \end{aligned}$$

La solution doit être admissible, donc les x_i sont positifs ou nuls. On va donc "pousser" la variable rentrante jusqu'à ce qu'une des variables de base s'annule. C'est cette variable qui est choisie comme variable sortante. Pour les i tels que $a_{ir} > 0$, on choisit le plus petit des $\frac{b_i}{a_{ir}}$, noté $\frac{b_s}{a_{sr}}$ et on donne à x_r la valeur $\frac{b_s}{a_{sr}}$, alors $x_s = 0$. Si tous les a_{ir} sont inférieurs à zéro, alors la solution est non bornée.

B.5.3 Calcul du nouveau sommet

Le calcul du nouveau sommet consiste à remplacer
- les a_{ij} par $a_{ij} - a_{jr}\frac{a_{si}}{a_{sr}}$
- les b_j par $b_j - a_{jr}\frac{b_s}{a_{sr}}$

Ce calcul est appelé "opération de pivot".

B.5.4 Test d'optimalité

Le critère d'arrêt est le suivant :

La solution de base courante est optimale si, lorsque la fonction est exprimée avec les variables hors base, aucune des variable ne conduit à l'amélioration du critère. Autrement dit, si les tous les coûts marginaux sont positifs ou nuls.

B.6 Résumé

Terminons en donnant une description schématique de l'algorithme du simplexe.

Pas 0. Initialisation
- Trouver une forme simpliciale du problème

Pas 1. Test d'optimalité
- Exprimer la fonction à partir des variables hors base
- **Si** tous les coûts marginaux sont positifs ou nuls **Alors** STOP, la solution obtenue est la solution optimale
- **Sinon**

Pas 2. Choix de la variable entrante
- Choisir la variable hors base dont le coût marginal est négatif et le plus petit possible. Soit x_r la variable entrante.

Pas 3. Choix de la variable sortante
- La variable sortante est la première à s'annuler : c'est celle pour laquelle le $\frac{b_i}{a_{ir}}$ est le plus petit avec $a_{ir} > 0$. Soit x_s la variable sortante.

Pas 4. Déterminer la nouvelle solution de base : opération du pivot
- Remplacer les a_{ij} par $a_{ij} - a_{jr}\frac{a_{si}}{a_{sr}}$
- Remplacer les b_j par $b_j - a_{jr}\frac{b_s}{a_{sr}}$

Retour au Pas 1.

C Principaux projets de recherche utilisant les drones

La liste (non exhaustive) suivante indique les principaux projets de recherche utilisant les drones :

- Université de Linköping, Suède - le projet WITAS [Doherty *et al.* 2000], développe des méthodes pour un drone complètement autonome effectuant ses opérations sur un terrain comprenant un trafic routier et aérien. Le but est de développer une plate-forme pour déployer des applications de surveillance de trafic ou d'assistance des services urgentistes.
- Université de Californie, Berkeley, USA - le projet [BEAR 2005] concerne le développement d'architectures pour les drones, particulièrement dans un contexte multi-agent.
- ONERA, France - le projet [RESSAC 2002] a pour objectif de produire une démonstration renouvelable et généralisable de capacités d'autonomie de contrôle du vol, de conduite de mission, d'acquisition et de traitement d'information pour la décision. La validation s'effectuera par une démonstration en vol et une en simulation, sur un scénario de recherche et sauvetage de personnes en milieu hostile.
- Georgia Institute of Technology, Atlanta, USA - le projet [UAVRF 2005] développe un banc test générique pouvant être utilisé pour tester le vol pour d'autres projets de recherche.
- Université de Stanford, USA - Le projet [Hummingbird 1999] du laboratoire Aerospace Robotics a pour but de démontrer l'utilité d'utiliser des robots hélicoptères à bas coûts pour réaliser des tâches sans l'aide d'opérateurs qualifiés.
- Université de Carnegie Mellon, USA - Le projet Hélicoptère autonome [CMU 1998] concerne le développement d'un robot hélicoptère guidé par vision capable de réaliser de manière autonome un ensemble bien défini d'objectifs en s'adaptant à la météo.
- Le projet européen [COMETS 2005] consiste à concevoir et à mettre en œuvre un système de coordination et de contrôle de véhicules aériens non habités. Le système "COMETS" inclue actuellement des hélicoptères et des dirigeables.
- ONERA- DCE/GESMA, France - le projet [NIVAS 2003] vise à réaliser le système de planification en boucle fermée qui permettra à un engin autonome sous-marin d'effectuer des missions de plusieurs heures, consistant à dérouler des procédures spécifiques en plusieurs points d'une zone d'intérêt.

D Scénarios tests et Résultats

☞ Dans les table de résultats, la première solution admissible est soit la solution donnée par la méthode de recherche d'une première solution admissible (Algorithme 4), soit la première solution admissible trouvée par l'algorithme. Dans tous les cas, il s'agit de la première solution qui fait évoluer la borne.

D.1 Récapitulatif des scénarios

La table D.1 récapitule toutes les missions testées.

Mission	Nbre zones objectif	Nbre E/S	Navigation
1	6	2	ligne droite/ contournement
2	4	2	ligne droite/ contournement
3	6	4	ligne droite/ contournement
4	6	2	ligne droite

Tab. D.1 – Récapitulatif des missions tests

Pour chaque mission, une planification globale est effectuée en début de mission. Ensuite, trois instants de planification sont déterminés, en début (D), milieu (M) et fin (F) de mission. Pour chaque instant de replanification, trois événements sont étudiés : manque de carburant (Evt_1), changement de la carte des dangers (Evt_2), changement de la carte des objectifs (Evt_3). On notera chaque scénario de replanification par le nom de la mission principale, suivi de la lettre correspondant à l'instant de replanification et du numéro de l'événement.

D.2 Mission 1

Carte principale et planification initiale

Carburant embarqué : 400 kg

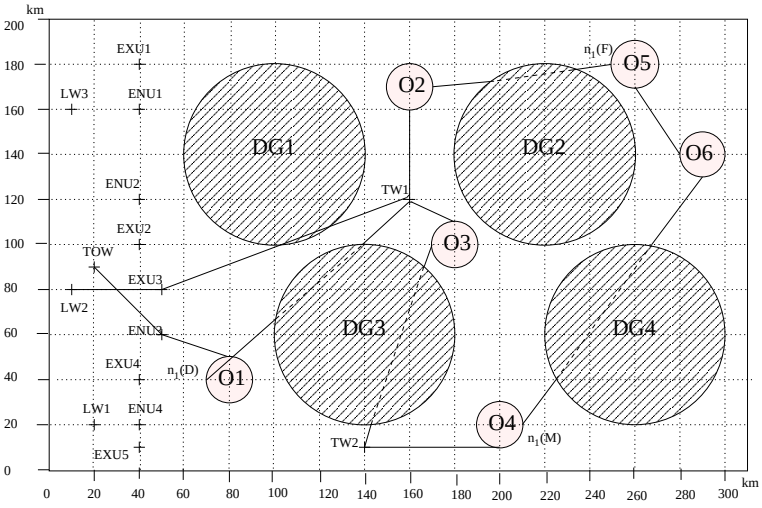


FIG. D.1 – Mission 1 : planification globale, choix des points de replanification

Mission 1-D1

Carburant restant : 250 kg

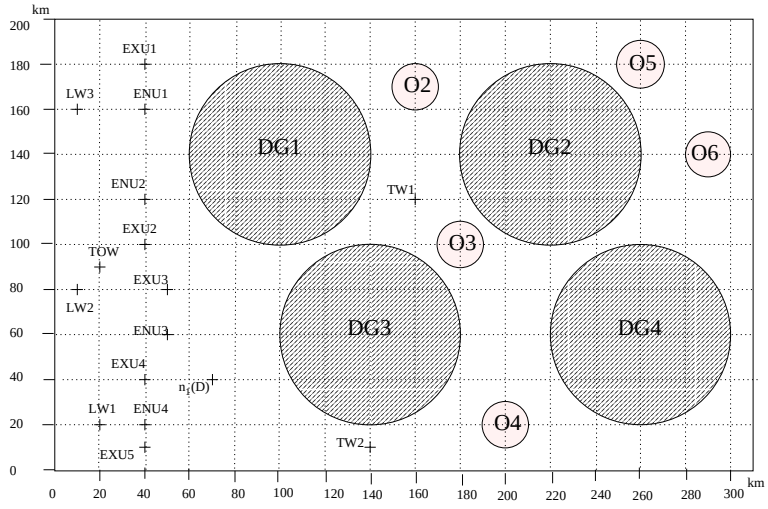


FIG. D.2 – Mission 1 : replanification en début de mission par manque de carburant

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-1500	15,2	-1500	15,2
H_1R_2	-1500	15,9	-1500	15,9
H_1R_3	-264	1,6	-264	1,6
H_1R_4	-264	2,2	-264	2,2
H_iR_1	-1500	15,8	-1500	15,8
H_iR_2	-1500	15,8	-1500	15,8
H_iR_3	-264	1,6	-264	1,6
H_iR_4	-264	2,2	-264	2,2
H_rR_1	-1500	15,9	-1500	15,9
H_rR_2	-1500	15,8	-1500	15,8
H_rR_3	-264	1,6	-264	1,6
H_rR_4	-264	2,2	-264	2,2
H_WR_1	-1500	16,2	-1500	16,2
H_WR_2	-1500	16,6	-1500	16,6
H_WR_3	-264	1,8	-264	1,8
H_WR_4	-264	3,7	-264	3,7

TAB. D.2 – Résultats Mission 1-D1 - replanification critique

Mission 1-D2

Carburant restant : 350 kg

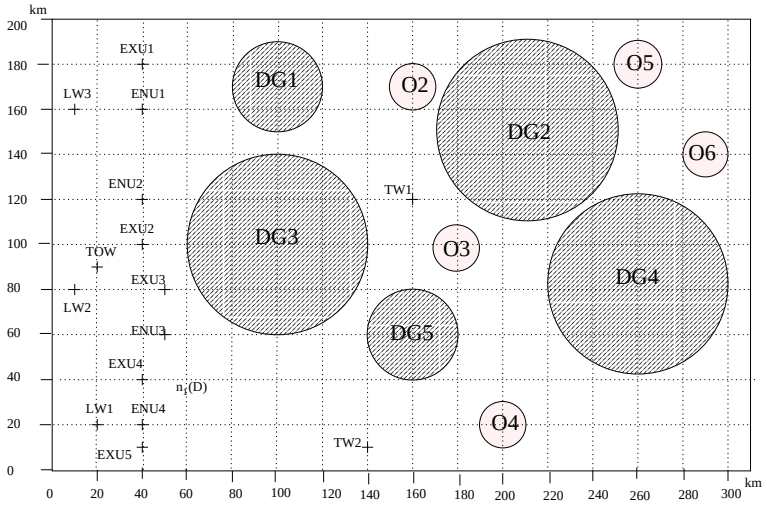


FIG. D.3 – Mission 1 : replanification en début de mission par changement de la carte des dangers

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-2027	10,2	-2228	285,4
H_1R_2	-2027	10,9	-2227	264,2
H_1R_3	-240	1,8	-1139	204,8
H_1R_4	-240	2,4	-240	2,4
H_iR_1	-2027	10,9	-2228	286,9
H_iR_2	-2027	11	-2228	289,8
H_iR_3	-240	1,9	-1139	204,1
H_iR_4	-240	2,2	-240	2,2
H_rR_1	-2027	11,1	-2027	11,1
H_rR_2	-2027	11,1	-2027	11,1
H_rR_3	-240	1,8	-240	1,8
H_rR_4	-240	2,5	-240	2,5
H_WR_1	-2027	11,4	-2227	211,2
H_WR_2	-2027	10,5	-2252	61
H_WR_3	-240	1,6	-1149	222,7
H_WR_4	-240	3,9	-240	3,9

TAB. D.3 – Résultats Mission 1-D2 - replanification nécessaire

Mission 1-D3

Carburant restant : 350 kg

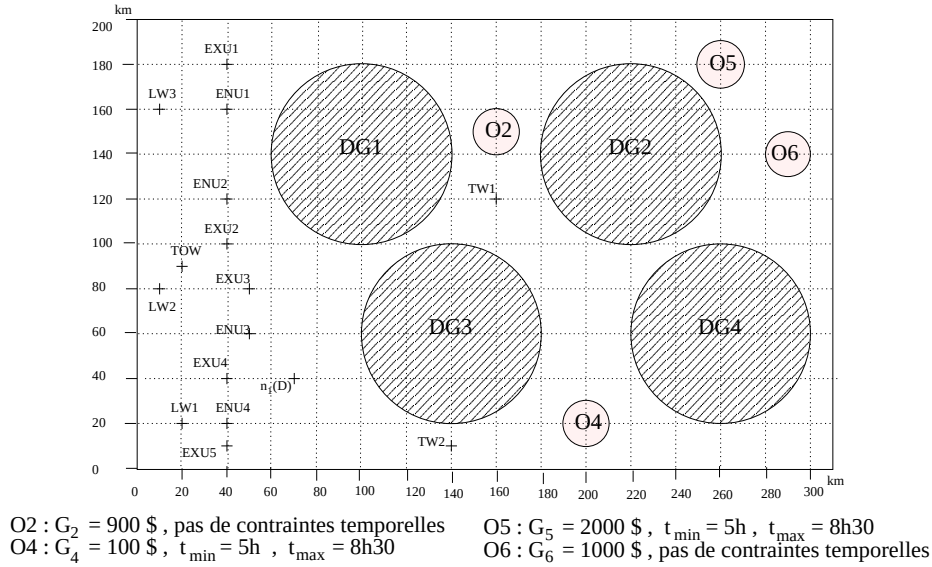


FIG. D.4 – Mission 1 : replanification en début de mission par changement de la carte des objectifs

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	55	11,6	-56	19,6
H_1R_2	55	11,6	-56	19,3
H_1R_3	-1521	1,7	-1521	1,7
H_1R_4	-1521	2,2	-1521	2,2
H_iR_1	55	11,6	-56	19,7
H_iR_2	55	11,6	-56	19,4
H_iR_3	-1521	1,7	-1521	1,7
H_iR_4	-1521	2,2	-1521	2,2
H_rR_1	55	11,7	55	11,7
H_rR_2	55	12	55	12
H_rR_3	-1521	1,9	-1521	1,9
H_rR_4	-1521	2,2	-1521	2,2
H_WR_1	55	11,4	55	11,4
H_WR_2	55	11,6	55	11,6
H_WR_3	-1521	1,7	-1521	1,7
H_WR_4	-1521	3,3	-1521	3,3

TAB. D.4 – Résultats Mission 1-D3 - replanification critique

Mission 1-M1

Carburant restant : 160 kg

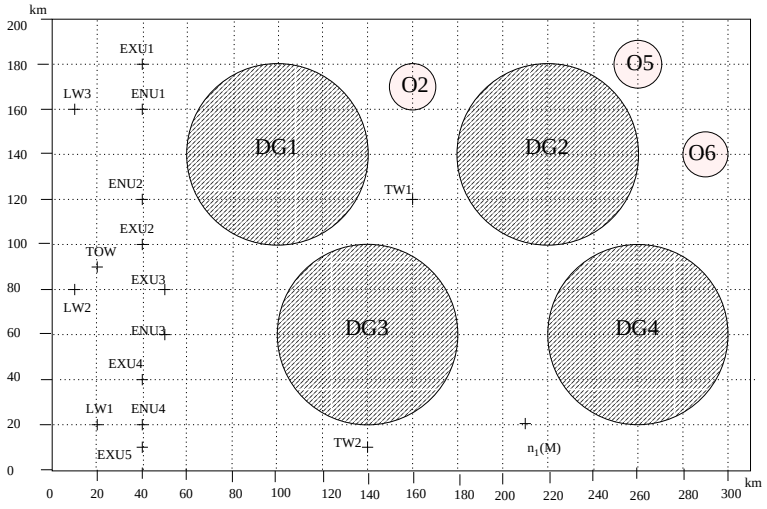


FIG. D.5 – Mission 1 : replanification en milieu de mission par manque de carburant

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-1402	4,9	-1402	4,9
H_1R_2	-1402	5	-1402	5
H_1R_3	-1649	2,4	-1649	2,4
H_1R_4	-1649	2,8	-1649	2,8
H_iR_1	-1402	4,9	-1402	4,9
H_iR_2	-1402	4,9	-1402	4,9
H_iR_3	-1649	2,4	-1649	2,4
H_iR_4	-1649	2,8	-1649	2,8
H_rR_1	-1402	4,9	-1402	4,9
H_rR_2	-1402	5	-1402	5
H_rR_3	-1649	2,4	-1649	2,4
H_rR_4	-1649	2,8	-1649	2,8
H_WR_1	-1402	4,9	-1402	4,9
H_WR_2	-1402	5	-1402	5
H_WR_3	-1649	2,4	-1649	2,4
H_WR_4	-1649	3,9	-1649	3,9

TAB. D.5 – Résultats Mission 1-M1 - replanification critique

Mission 1-M2

Carburant restant : 250 kg

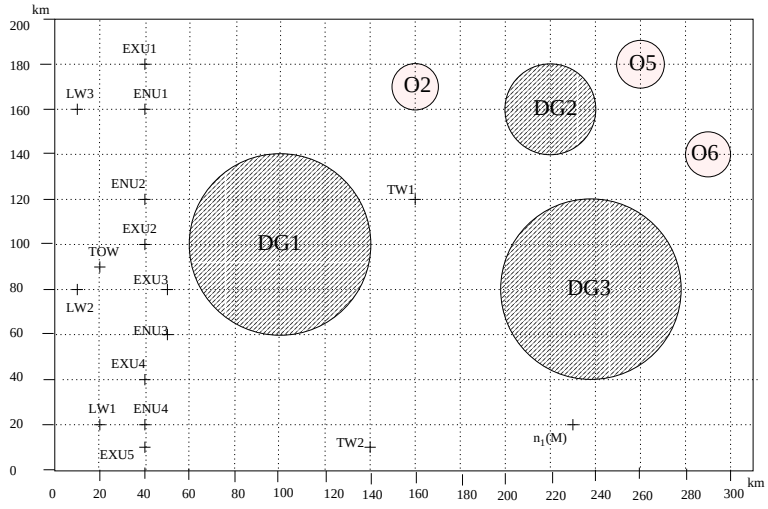


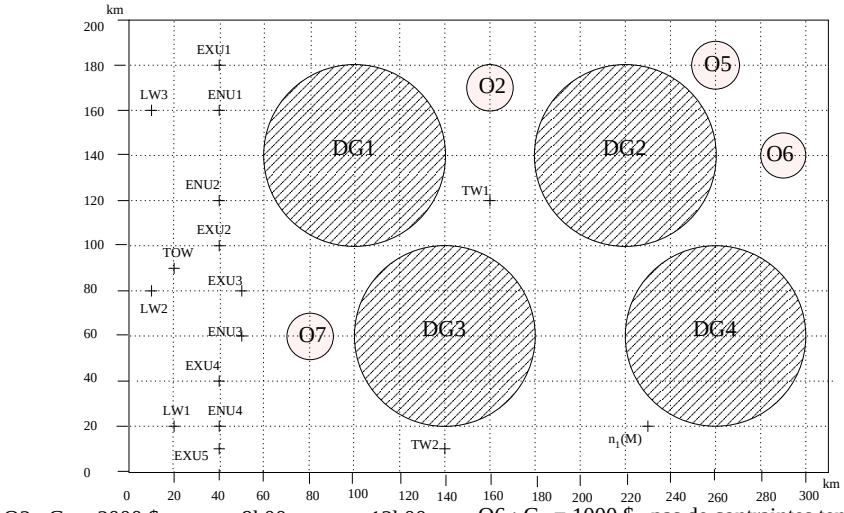
FIG. D.6 – Mission 1 : replanification en milieu de mission par changement de la carte des dangers

Algorithme	Meil. Sol.	
	$\hat{J}$	temps (s)
H_1R_1	-2229	3,4
H_1R_2	-2229	3,4
H_1R_3	-2229	2
H_1R_4	-2229	2,4
H_iR_1	-2229	3,4
H_iR_2	-2229	3,4
H_iR_3	-2229	2
H_iR_4	-2229	2,4
H_rR_1	-2229	3,4
H_rR_2	-2229	3,4
H_rR_3	-2229	2
H_rR_4	-2229	2,4
H_WR_1	-2229	3,4
H_WR_2	-2229	3,4
H_WR_3	-2229	2
H_WR_4	-2229	3,4

TAB. D.6 – Résultats Mission 1-M2 - replanification éventuelle

Mission 1-M3

Carburant restant : 250 kg



O2 : G_2 = 2000 \$, t_{min} = 9h00 , t_{max} = 12h00
O5 : G_5 = 900 \$, pas de contraintes temporelles
O6 : G_6 = 1000 \$, pas de contraintes temporelles
O7 : G_7 = 1500 \$, t_{min} = 11,8h , t_{max} = 15,3h

FIG. D.7 – Mission 1 : replanification en milieu de mission par changement de la carte des objectifs

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-817	11	-3225	19,1
H_1R_2	-766	11	-3225	19
H_1R_3	-723	2,7	-723	2,7
H_1R_4	-723	3,8	-723	3,8
H_iR_1	-817	11	-3225	19,2
H_iR_2	-766	11	-3225	19
H_iR_3	-723	2,8	-723	2,8
H_iR_4	-723	3,8	-723	3,8
H_rR_1	∞	-	∞	-
H_rR_2	∞	-	∞	-
H_rR_3	-723	2,8	-723	2,8
H_rR_4	-723	3,8	-723	3,8
H_WR_1	-817	11	-3225	19,1
H_WR_2	-1281	15,9	-1281	15,9
H_WR_3	-723	2,7	-723	2,7
H_WR_4	-723	6,2	-723	6,2

TAB. D.7 – Résultats Mission 1-M3 - replanification critique

Mission 1-F1

Carburant restant : 70 kg

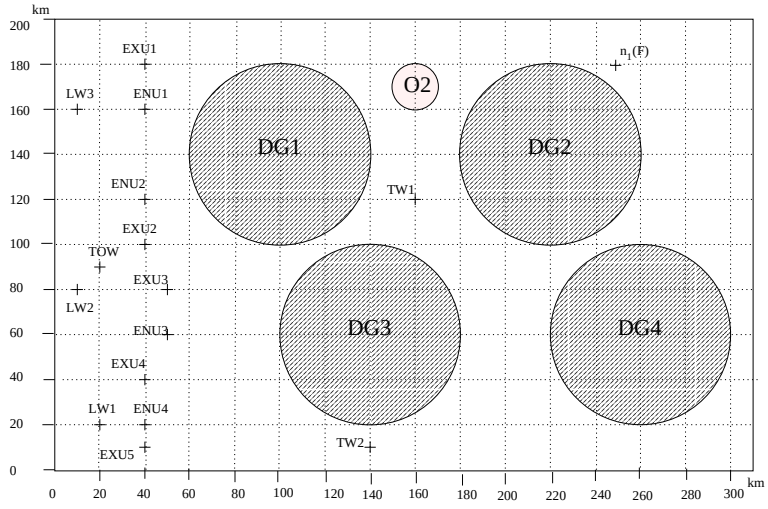


FIG. D.8 – Mission 1 : replanification en fin de mission par manque de carburant

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-2686	1,7	-2686	1,7
H_1R_2	-2686	1,7	-2686	1,7
H_1R_3	-2686	2,3	-2686	2,3
H_1R_4	-2686	2,7	-2686	2,7
H_iR_1	-2686	1,9	-2686	1,9
H_iR_2	-2686	1,8	-2686	1,8
H_iR_3	-2686	2,1	-2686	2,1
H_iR_4	-2686	2,6	-2686	2,6
H_rR_1	-2686	3	-2767	4,8
H_rR_2	176627	2,5	-2767	5,1
H_rR_3	-2686	4,3	-2767	5,7
H_rR_4	-2767	5,8	-2767	5,8
H_WR_1	-2686	1,7	-2686	1,7
H_WR_2	-2686	1,7	-2686	1,7
H_WR_3	-2686	2,1	-2686	2,1
H_WR_4	-2686	3	-2686	3

TAB. D.8 – Résultats Mission 1-F1 - replanification critique

Mission 1-F2

Carburant restant : 160 kg

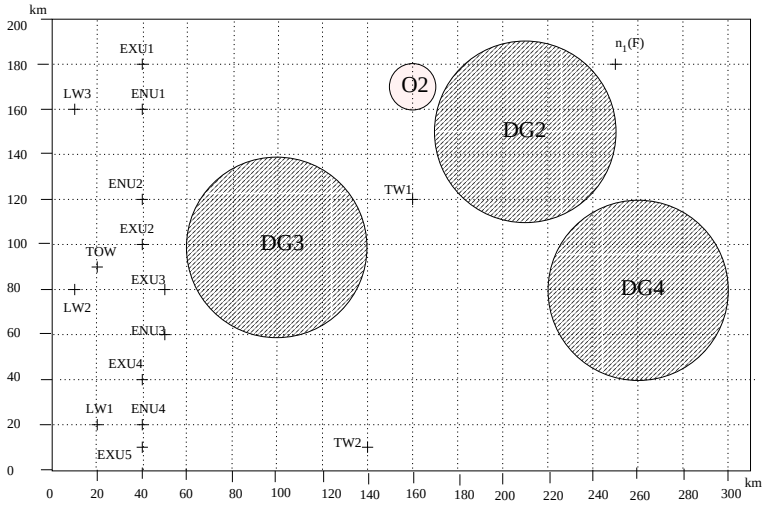


FIG. D.9 – Mission 1 : replanification en fin de mission par changement de la carte des dangers

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-3258	0,5	-3258	0,5
H_1R_2	-3258	0,5	-3258	0,5
H_1R_3	-3258	0,4	-3258	0,4
H_1R_4	-3258	0,5	-3258	0,5
H_iR_1	-3258	0,4	-3258	0,4
H_iR_2	-3258	0,5	-3258	0,5
H_iR_3	-3258	0,5	-3258	0,5
H_iR_4	-3258	0,5	-3258	0,5
H_rR_1	-3258	0,5	-3351	4,8
H_rR_2	-3258	0,5	-3351	2,5
H_rR_3	-3258	0,5	-3351	6,4
H_rR_4	-3258	0,5	-3351	2,6
H_WR_1	-3258	0,4	-3258	0,4
H_WR_2	-3258	0,4	-3258	0,4
H_WR_3	-3258	0,4	-3258	0,4
H_WR_4	-3258	0,4	-3258	0,4

TAB. D.9 – Résultats Mission 1-F2 - replanification nécessaire

Mission 1-F3

Carburant restant : 160 kg

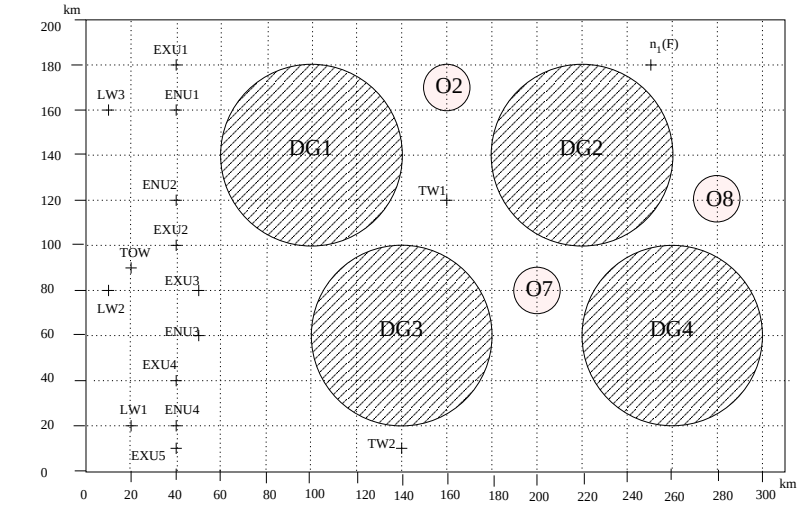


FIG. D.10 – Mission 1 : replanification en fin de mission par changement de la carte des objectifs

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-2686	8,8	-3556	18,2
H_1R_2	-2686	8,9	-3556	18,2
H_1R_3	-2686	2,5	-3556	15,8
H_1R_4	∞	-	∞	-
H_iR_1	-2686	8,5	-3556	17,9
H_iR_2	-2686	9	-3556	18,4
H_iR_3	-2686	2,5	-3556	16,2
H_rR_1	∞	-	∞	-
H_rR_2	∞	-	∞	-
H_rR_3	∞	-	∞	-
H_rR_4	∞	-	∞	-
H_WR_1	-2686	9	-2686	9
H_WR_2	-2996	11,1	-2996	11,1
H_WR_3	-2686	2,5	-3556	15,9
H_WR_4	∞	-	∞	-

TAB. D.10 – Résultats Mission 1-F3 - replanification critique

D.3 Mission 2

Carte principale et planification initiale

Carburant embarqué : 300 kg

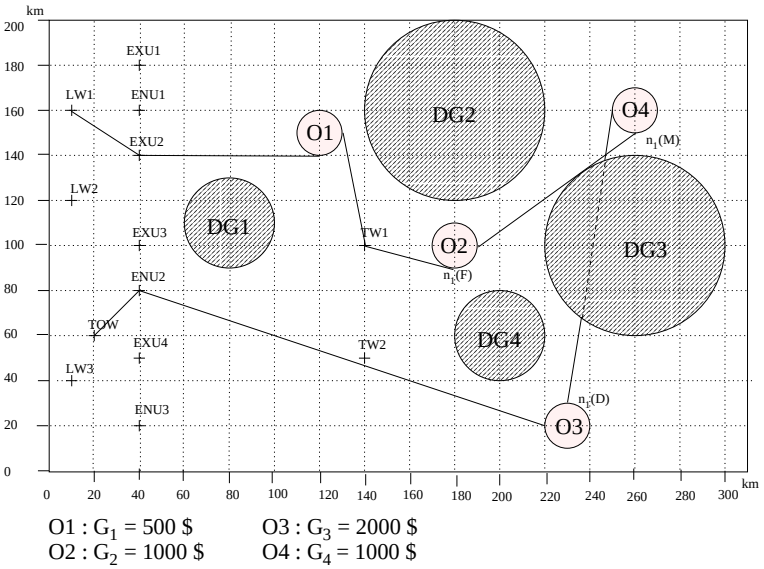


FIG. D.11 – Mission 2 : planification globale, choix des points de replanification

Mission 2-D1

Carburant restant : 130 kg

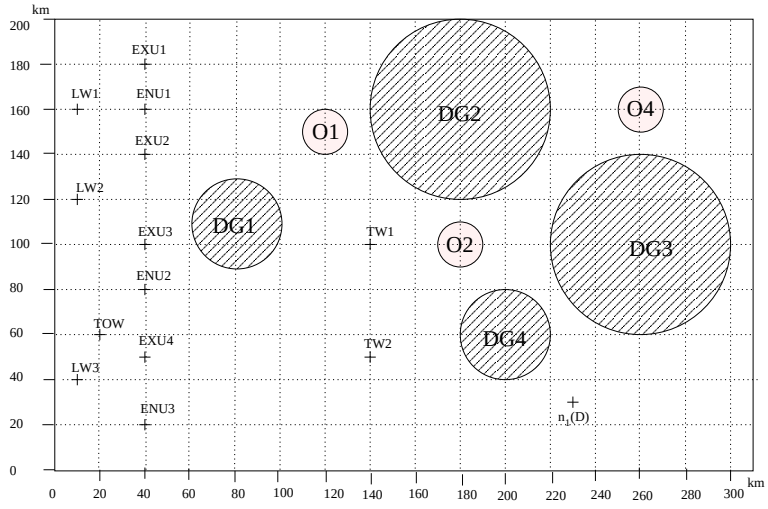


FIG. D.12 – Mission 2 : replanification en début de mission par manque de carburant

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-2286	5	-2286	5
H_1R_2	-2286	5	-2286	5
H_1R_3	-1805	1,9	-1805	1,9
H_1R_4	-1805	3,1	-1805	3,1
H_iR_1	-2286	5	-2286	5
H_iR_2	-2286	5	-2286	5
H_iR_3	-1805	1,9	-1805	1,9
H_iR_4	-1805	3,1	-1805	3,1
H_rR_1	-2286	5,1	-2286	5,1
H_rR_2	-2286	5	-2286	5
H_rR_3	-1805	1,9	-1805	1,9
H_rR_4	-1805	3,1	-1805	3,1
H_WR_1	-2286	5	-2286	5
H_WR_2	-2286	5	-2286	5
H_WR_3	-1805	1,9	-1805	1,9
H_WR_4	-1805	5,6	-1805	5,6

TAB. D.11 – Résultats Mission 2-D1 - replanification critique

Mission 2-D2

Carburant restant : 240 kg

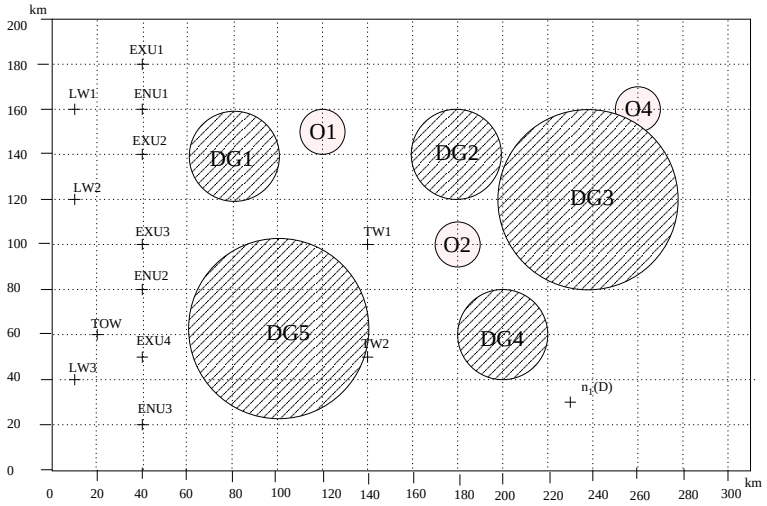


FIG. D.13 – Mission 2 : replanification en début de mission par changement de la carte des dangers

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-2649	5	-2649	5
H_1R_2	-2649	5	-2649	5
H_1R_3	-1795	1,9	-1795	1,9
H_1R_4	-1795	2,9	-1795	2,9
H_iR_1	-2649	5	-2649	5
H_iR_2	-2649	4,6	-2649	4,6
H_iR_3	-1795	1,8	-1795	1,8
H_iR_4	-1795	2,9	-1795	2,9
H_rR_1	-2649	5	-2649	5
H_rR_2	-2649	5	-2649	5
H_rR_3	-1795	1,9	-1795	1,9
H_rR_4	-1795	3	-1795	3
H_WR_1	-2649	5	-2649	5
H_WR_2	-2649	5,1	-2649	5,1
H_WR_3	-1795	1,9	-1795	1,9
H_WR_4	-1795	5,3	-1795	5,3

TAB. D.12 – Résultats Mission 2-D2 - replanification critique

Mission 2-D3

Carburant restant : 240 kg

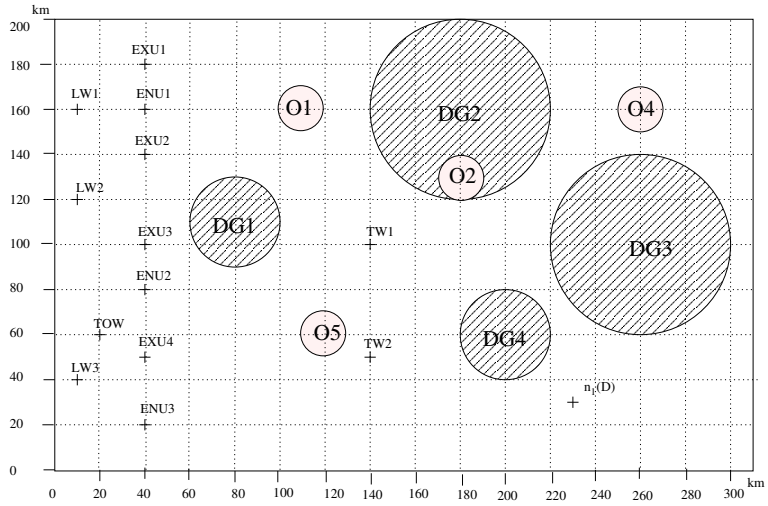


FIG. D.14 – Mission 2 : replanification en début de mission par changement de la carte des objectifs

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-2547	17,6	-2547	17,6
H_1R_2	-2547	17,6	-2547	17,6
H_1R_3	-2407	2,1	-2407	2,1
H_1R_4	-2407	2,5	-2407	2,5
H_iR_1	-2547	17,4	-2547	17,4
H_iR_2	-2547	17,4	-2547	17,4
H_iR_3	-2407	2	-2407	2
H_iR_4	-2407	2,5	-2407	2,5
H_rR_1	-2547	17,4	-2547	17,4
H_rR_2	-2547	17,4	-2547	17,4
H_rR_3	-2407	2	-2407	2
H_rR_4	-2407	2,5	-2407	2,5
H_WR_1	-2547	17,7	-2547	17,7
H_WR_2	-2547	17,9	-2547	17,9
H_WR_3	-2407	2,1	-2407	2,1
H_WR_4	-2407	3,7	-2407	3,7

TAB. D.13 – Résultats Mission 2-D3 - replanification critique

Mission 2-M1

Carburant restant : 160 kg

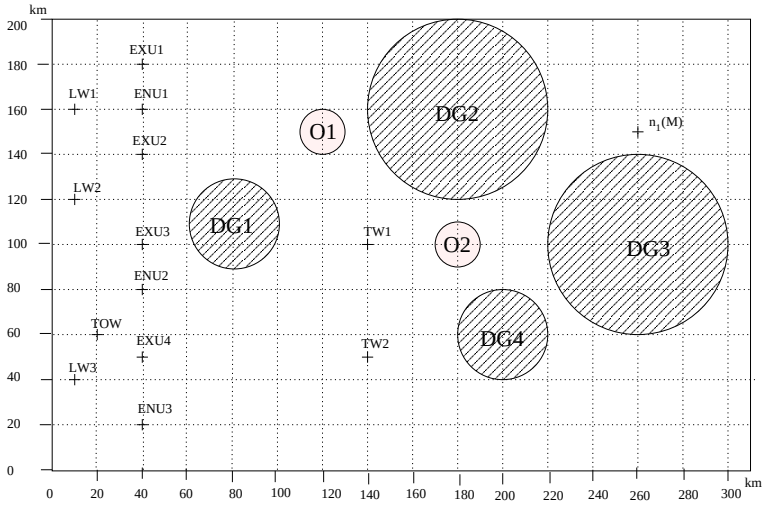


FIG. D.15 – Mission 2 : replanification en milieu de mission par manque de carburant

Algorithme	Meil. Sol.	
	$\hat{J}$	temps (s)
H_1R_1	-3415	1,8
H_1R_2	-3415	1,8
H_1R_3	-3415	0,9
H_1R_4	-3415	1,2
H_iR_1	-3415	1,8
H_iR_2	-3415	1,8
H_iR_3	-3415	0,9
H_iR_4	-3415	1,2
H_rR_1	-3415	1,8
H_rR_2	-3415	1,9
H_rR_3	-3415	0,9
H_rR_4	-3415	1,2
H_WR_1	-3415	1,8
H_WR_2	-3415	1,8
H_WR_3	-3415	0,9
H_WR_4	-3415	1,7

TAB. D.14 – Résultats Mission 2-M1 - replanification éventuelle

Mission 2-M2

Carburant restant : 190 kg

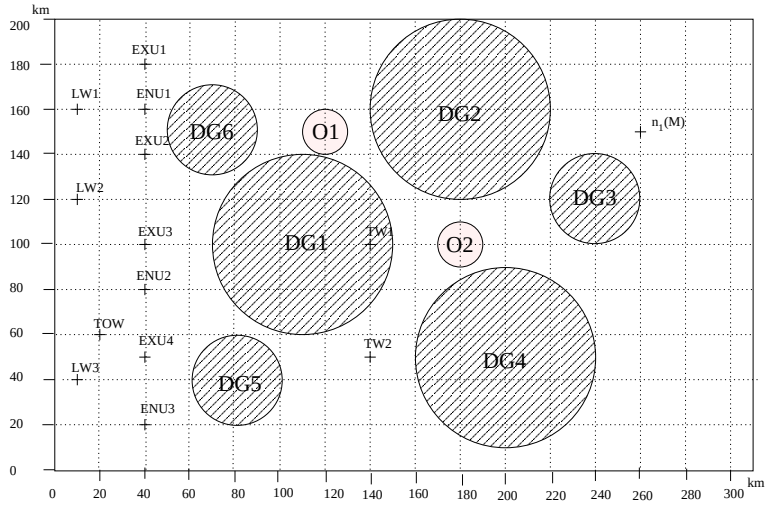


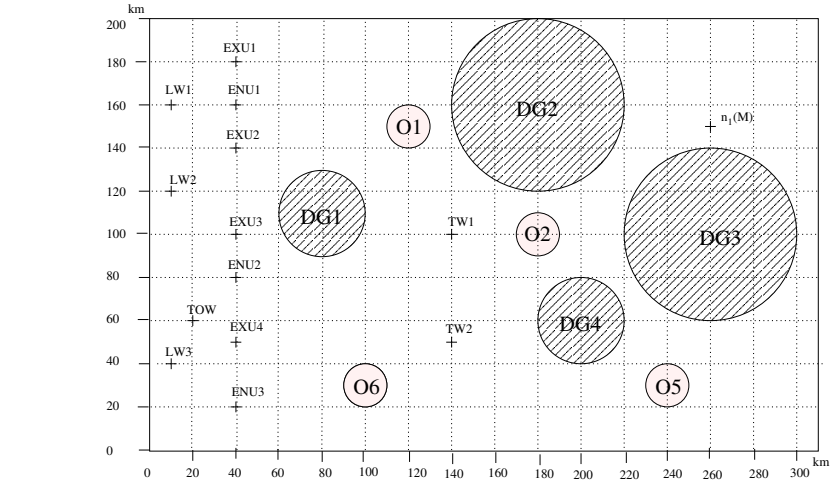
FIG. D.16 – Mission 2 : replanification en milieu de mission par changement de la carte des dangers

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-3178	1,6	-3178	1,6
H_1R_2	-3178	1,7	-3178	1,7
H_1R_3	-2746	1,3	-2746	1,3
H_1R_4	-2746	1,9	-2746	1,9
H_iR_1	-3178	1,6	-3178	1,6
H_iR_2	-3178	1,7	-3178	1,7
H_iR_3	-2746	1,3	-2746	1,3
H_iR_4	-2746	1,9	-2746	1,9
H_rR_1	-3178	1,7	-3178	1,7
H_rR_2	-3178	1,7	-3336	16,9
H_rR_3	-2746	1,3	-2746	1,3
H_rR_4	-2746	1,9	-3336	18,2
H_WR_1	-3178	1,7	-3178	1,7
H_WR_2	-3178	1,7	-3178	1,7
H_WR_3	-2746	1,3	-2746	1,3
H_WR_4	-2746	3,3	-2746	3,3

TAB. D.15 – Résultats Mission 2-M2 - replanification critique

Mission 2-M3

Carburant restant : 190 kg



O1 : G_1 = 500 \$ – t_{min} = 9h ; t_{max} = 15h
O2 : G_2 = 1000 \$ – pas de contraintes temporelles
O5 : G_5 = 2000 \$ – pas de contraintes temporelles
O6 : G_6 = 1000 \$ – t_{min} = 6,5h ; t_{max} = 11h

FIG. D.17 – Mission 2 : replanification en milieu de mission par changement de la carte des objectifs

Algorithme	Meil. Sol.	
	$\hat{J}$	temps (s)
H_1R_1	-4670	32,2
H_1R_2	-4670	33,6
H_1R_3	-4625	274,8
H_1R_4	-3390	3,2
H_iR_1	-4670	30,9
H_iR_2	-4670	35
H_iR_3	-4625	282,1
H_iR_4	-3390	3,2
H_rR_1	-4670	138,7
H_rR_2	-4796	152,4
H_rR_3	-3390	2,5
H_rR_4	-3390	3,1
H_WR_1	-3390	12,4
H_WR_2	-3390	12,4
H_WR_3	-4655	686,2
H_WR_4	-3390	4,5

TAB. D.16 – Résultats Mission 2-M3 - replanification éventuelle

Mission 2-F1

Carburant restant : 50 kg

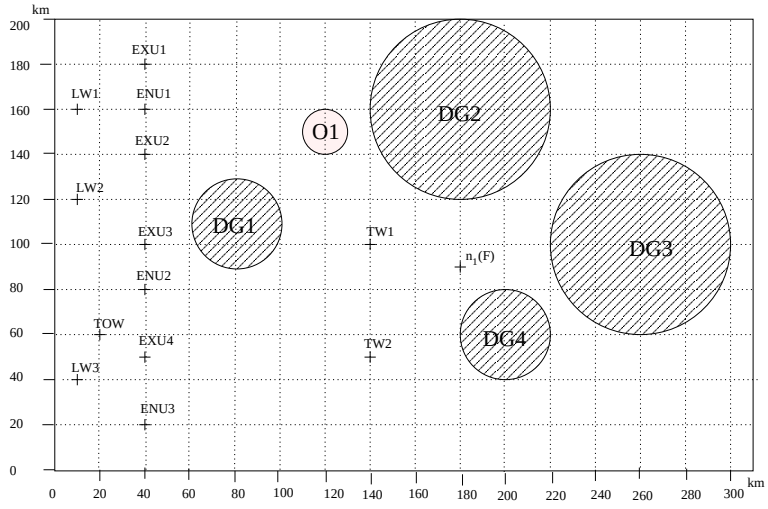


FIG. D.18 – Mission 2 : replanification en fin de mission par manque de carburant

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-3825	1,5	-3825	1,5
H_1R_2	-3807	1,5	-3825	1,7
H_1R_3	-3825	2,8	-3825	2,8
H_1R_4	-3825	3,5	-3825	3,5
H_iR_1	-3825	1,4	-3825	1,4
H_iR_2	-3807	1,5	-3825	1,7
H_iR_3	-3825	2,8	-3825	2,8
H_iR_4	-3825	3,5	-3825	3,5
H_rR_1	-3825	2,6	-3832	3,7
H_rR_2	-3825	2,7	-3832	3,6
H_rR_3	-3825	4,3	-3832	4,8
H_rR_4	-3832	4,8	-3832	4,8
H_WR_1	-3825	1,5	-3825	1,5
H_WR_2	-3807	1,5	-3807	1,5
H_WR_3	-3825	2,8	-3825	2,8
H_WR_4	-3807	4,8	-3807	4,8

TAB. D.17 – Résultats Mission 2-F1 - replanification critique

Mission 2-F2

Carburant restant : 150 kg

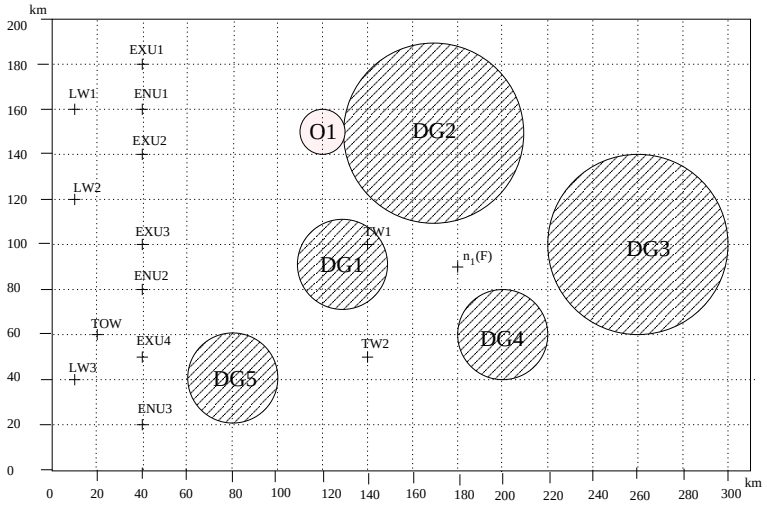


FIG. D.19 – Mission 2 : replanification en fin de mission par changement de la carte des dangers

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-3873	0,5	-3873	0,5
H_1R_2	-3873	0,4	-3873	0,4
H_1R_3	-3873	0,4	-3873	0,4
H_1R_4	-3873	0,5	-3873	0,5
H_iR_1	-3873	0,4	-3873	0,4
H_iR_2	-3873	0,4	-3873	0,4
H_iR_3	-3873	0,4	-3873	0,4
H_iR_4	-3873	0,5	-3873	0,5
H_rR_1	-3873	0,4	-3873	0,4
H_rR_2	-3873	0,4	-3873	0,4
H_rR_3	-3873	0,4	-3873	0,4
H_rR_4	-3873	0,5	-3873	0,5
H_WR_1	-3873	0,4	-3873	0,4
H_WR_2	-3873	0,4	-3873	0,4
H_WR_3	-3873	0,4	-3873	0,4
H_WR_4	-3873	0,7	-3873	0,7

TAB. D.18 – Résultats Mission 2-F2 - replanification critique

Mission 2-F3

Carburant restant : 150 kg

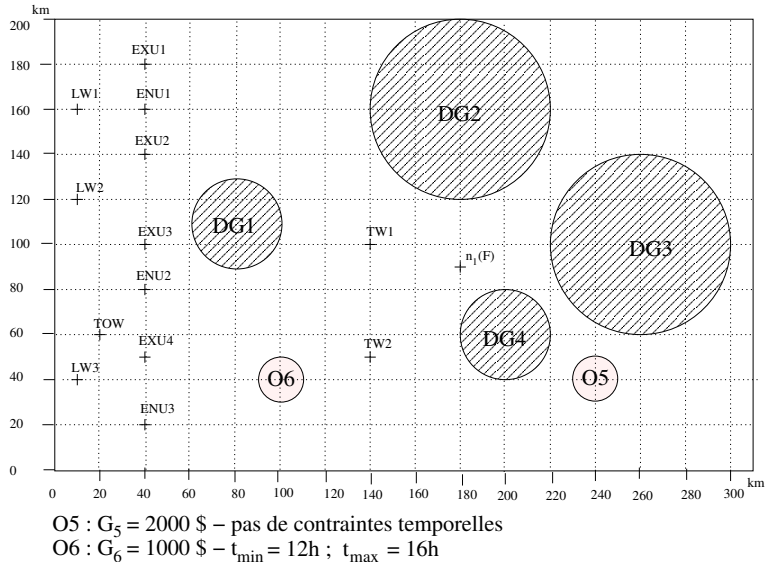


FIG. D.20 – Mission 2 : replanification en fin de mission par changement de la carte des objectifs

Algorithme	Meil. Sol.	
	$\hat{J}$	temps (s)
H_1R_1	-5797	21,9
H_1R_2	-5797	21,4
H_1R_3	-5797	19,6
H_1R_4	-5797	785,7
H_iR_1	-5797	21,3
H_iR_2	-5797	21,5
H_iR_3	-5797	19,6
H_iR_4	-5797	785,6
H_rR_1	-5797	28,9
H_rR_2	-5797	16,9
H_rR_3	-5797	41,7
H_rR_4	-5797	53,7
H_WR_1	-5794	12,3
H_WR_2	-5794	12,2
H_WR_3	-5797	19
H_WR_4	-5785	255,8

TAB. D.19 – Résultats Mission 2-F3 - replanification éventuelle

D.4 Mission 3

Carte principale et planification initiale

Carburant embarqué : 510 kg

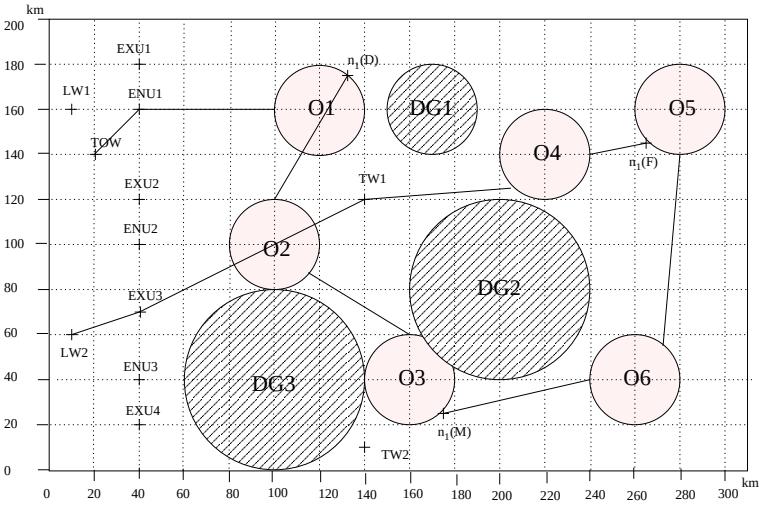


FIG. D.21 – Mission 3 : planification globale, choix des points de replanification

Mission 3-D1

Carburant restant : 350 kg

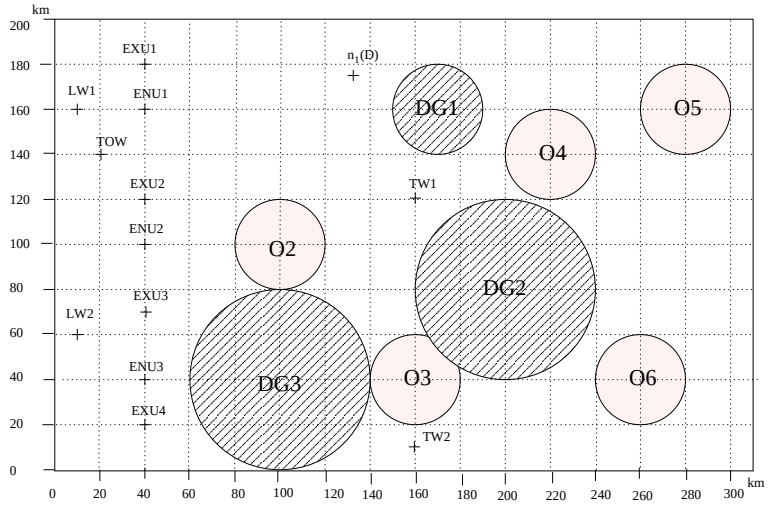


FIG. D.22 – Mission 3 : replanification en début de mission par manque de carburant

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-48	7,7	-48	7,7
H_1R_2	-48	7,9	-48	7,9
H_1R_3	-66	2,9	-66	2,9
H_1R_4	-66	5,2	-66	5,2
H_iR_1	-48	7,4	-48	7,4
H_iR_2	-48	7,4	-48	7,4
H_iR_3	-66	2,9	-66	2,9
H_iR_4	-66	5,4	-66	5,4
H_rR_1	-48	6,8	-48	6,8
H_rR_2	-48	7	-48	7
H_rR_3	-66	2,8	-66	2,8
H_rR_4	-66	4,8	-66	4,8
H_WR_1	-48	6,8	-48	6,8
H_WR_2	-48	7,3	-48	7,3
H_WR_3	-66	2,9	-66	2,9
H_WR_4	-66	10,2	-66	10,2

TAB. D.20 – Résultats Mission 3-D1 - replanification critique

Mission 3-D2

Carburant restant : 430 kg

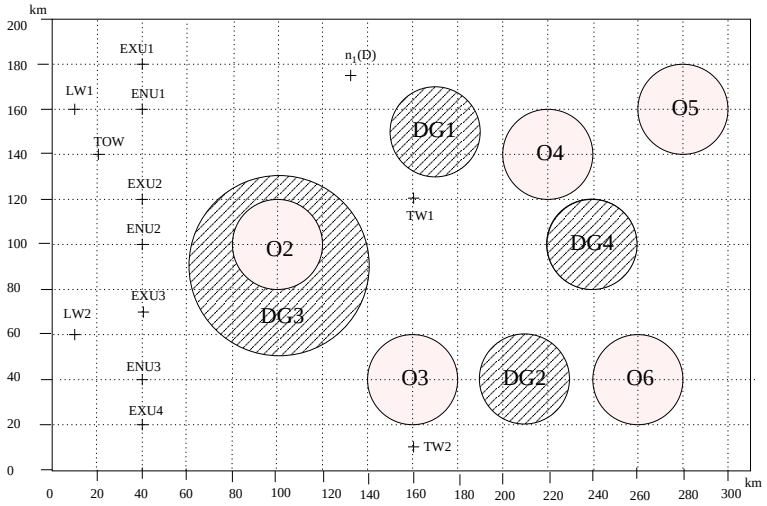


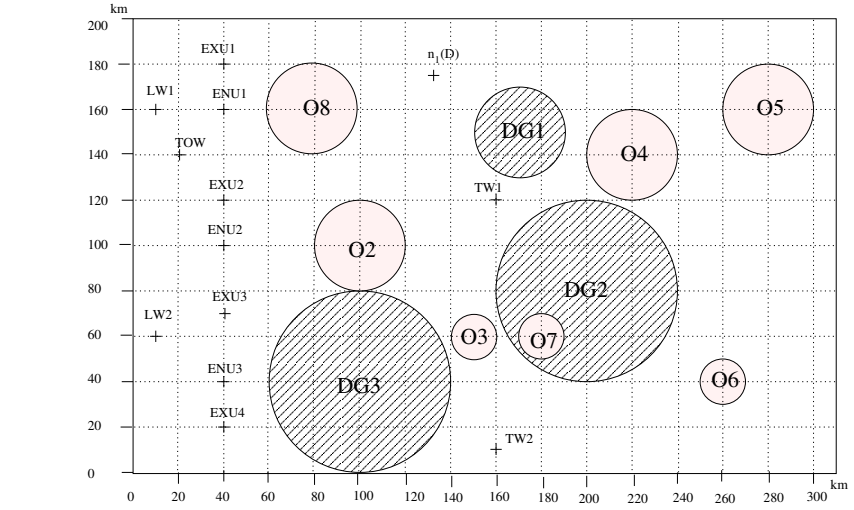
FIG. D.23 – Mission 3 : replanification en début de mission par changement de la carte des dangers

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	85	4	85	4
H_1R_2	85	4	85	4
H_1R_3	124	1,7	124	1,7
H_1R_4	124	3,1	124	3,1
H_iR_1	85	4,3	85	4,3
H_iR_2	85	4	85	4
H_iR_3	124	1,8	124	1,8
H_iR_4	124	3,2	124	3,2
H_rR_1	85	4,1	85	4,1
H_rR_2	85	4,3	85	4,3
H_rR_3	124	2	124	2
H_rR_4	124	3,1	124	3,1
H_WR_1	85	4	85	4
H_WR_2	85	3,7	85	3,7
H_WR_3	124	1,8	124	1,8
H_WR_4	124	5,6	124	5,6

TAB. D.21 – Résultats Mission 3-D2 - replanification critique

Mission 3-D3

Carburant restant : 430 kg



O2 : G_2 = 500 $ – pas de contraintes temporelles
O3 : G_3 = 2000 $ – pas de contraintes temporelles
O4 : G_4 = 4000 $ – pas de contraintes temporelles
O5 : G_5 = 1000 $ – pas de contraintes temporelles
O6 : G_6 = 1000 $ – t_{min} = 9h – t_{max}= 15h
O7 : G_7 = 2000$ – pas de contraintes temporelles
O8 : G_8 = 2000 $ – pas de contraintes temporelles

FIG. D.24 – Mission 3 : replanification en début de mission par changement de la carte des objectifs

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-1086	16,6	-5308	63,6
H_1R_2	-1086	17,3	-5234	239
H_1R_3	-1108	3	-2651	120,3
H_1R_4	-1108	4,7	-1108	4,7
H_iR_1	-1086	16,9	-5308	64,3
H_iR_2	-1086	17,1	-5234	236,9
H_iR_3	-1108	3,1	-2651	122,5
H_iR_4	-1108	4,4	-1108	4,4
H_rR_1	-1086	17,7	-1086	17,7
H_rR_2	-1086	17,1	-1086	17,1
H_rR_3	-1108	3	-1108	3
H_rR_4	-1108	5,1	-1108	5,1
H_WR_1	-1086	16,9	-5493	199,8
H_WR_2	-1086	16,5	-6131	263
H_WR_3	-1108	3	-2625	121,2
H_WR_4	-1108	8,3	-2827	205,3

TAB. D.22 – Résultats Mission 3-D3 - replanification nécessaire

Mission 3-M1

Carburant restant : 180 kg

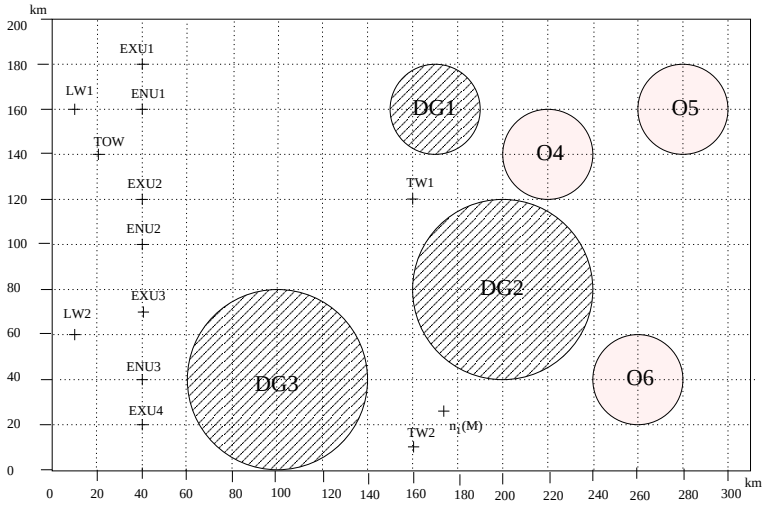


FIG. D.25 – Mission 3 : replanification en milieu de mission par manque de carburant

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-764	3,2	-835	7,8
H_1R_2	-764	3,2	-835	7,8
H_1R_3	-794	1,8	-794	1,8
H_1R_4	-794	2,8	-794	2,8
H_iR_1	-764	3,2	-835	7,8
H_iR_2	-764	3,2	-835	7,8
H_iR_3	-794	1,8	-794	1,8
H_iR_4	-794	2,8	-794	2,8
H_rR_1	-764	3,2	-764	3,2
H_rR_2	-764	3,2	-764	3,2
H_rR_3	-794	1,8	-794	1,8
H_rR_4	-794	2,8	-794	2,8
H_WR_1	-764	3,2	-764	3,2
H_WR_2	-764	3,2	-764	3,2
H_WR_3	-794	1,8	-794	1,8
H_WR_4	-794	5,1	-794	5,1

TAB. D.23 – Résultats Mission 3-M1 - replanification critique

Mission 3-M2

Carburant restant : 280 kg

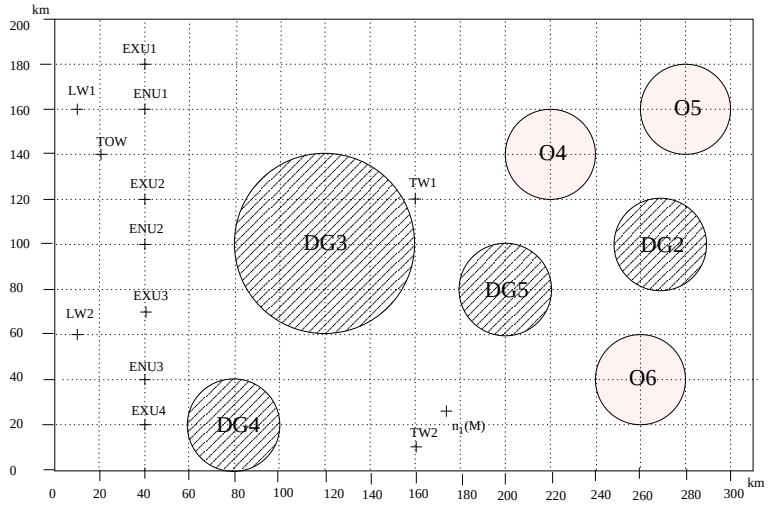


FIG. D.26 – Mission 3 : replanification en milieu de mission par changement de la carte des dangers

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-794	3,5	-946	14,3
H_1R_2	-794	3,5	-946	14,3
H_1R_3	-824	1,8	-824	1,8
H_1R_4	-824	2,8	-824	2,8
H_iR_1	-794	3,5	-946	14,4
H_iR_2	-794	3,5	-946	14,4
H_iR_3	-824	1,8	-824	1,8
H_iR_4	-824	2,8	-824	2,8
H_rR_1	-794	3,5	-794	3,5
H_rR_2	-794	3,5	-794	3,5
H_rR_3	-824	1,8	-824	1,8
H_rR_4	-824	2,8	-824	2,8
H_WR_1	-794	3,5	-794	3,5
H_WR_2	-794	3,6	-794	3,6
H_WR_3	-824	1,8	-824	1,8
H_WR_4	-824	5,1	-824	5,1

TAB. D.24 – Résultats Mission 3-M2 - replanification critique

Mission 3-M3

Carburant restant : 280 kg

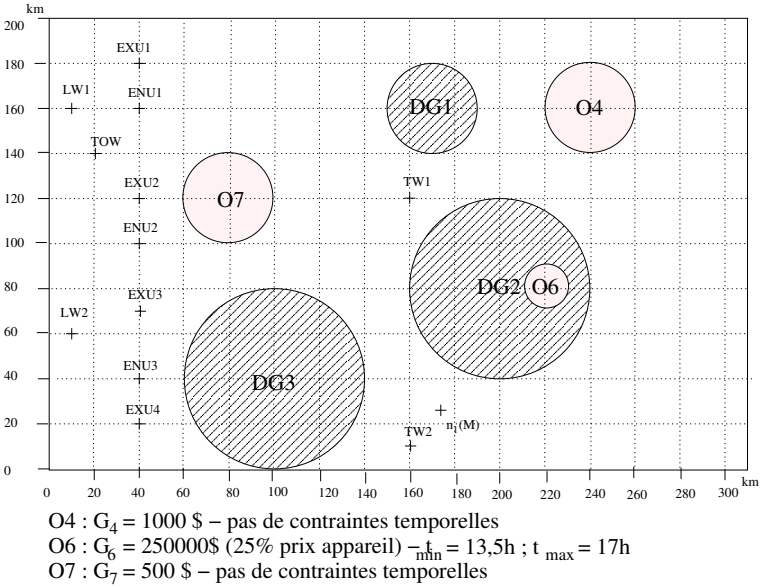


FIG. D.27 – Mission 3 : replanification en milieu de mission par changement de la carte des objectifs

Algorithme	Meil. Sol.	
	$\hat{J}$	temps (s)
H_1R_1	-24590	4,8
H_1R_2	-24590	4,8
H_1R_3	-24590	7
H_1R_4	-24590	10
H_iR_1	-24590	4,8
H_iR_2	-24590	4,8
H_iR_3	-24590	7
H_iR_4	-24590	10,1
H_rR_1	-24590	4,8
H_rR_2	-25456	126,9
H_rR_3	-24590	7
H_rR_4	-25456	132,9
H_WR_1	-24590	4,8
H_WR_2	-25068	897,3
H_WR_3	-24590	7
H_WR_4	-24590	16,7

TAB. D.25 – Résultats Mission 3-M3 - replanification éventuelle

Mission 3-F1

Carburant restant : 60 kg

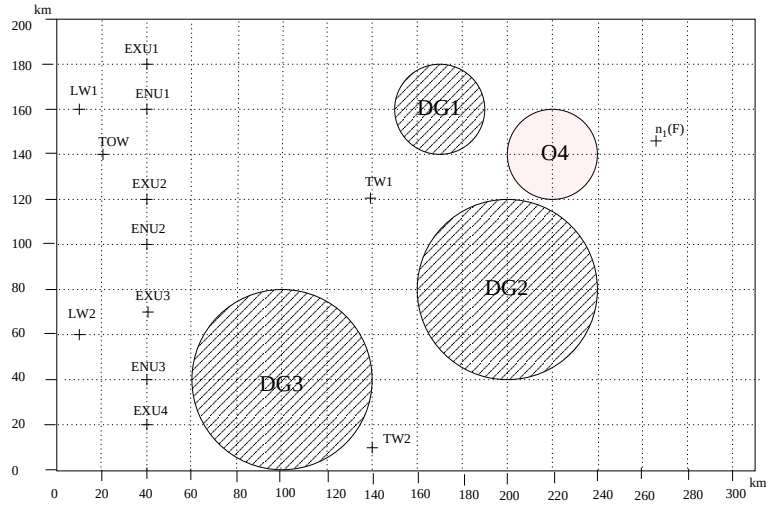


FIG. D.28 – Mission 3 : replanification en fin de mission par manque de carburant

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-2737	1	-2743	1,1
H_1R_2	-2737	1	-2743	1,1
H_1R_3	-2743	0,8	-2743	0,8
H_1R_4	-2743	0,9	-2743	0,9
H_iR_1	-2737	1,1	-2743	1,1
H_iR_2	-2737	1	-2743	1,1
H_iR_3	-2743	0,8	-2743	0,8
H_iR_4	-2743	0,9	-2743	0,9
H_rR_1	-2737	1,7	-2745	1,8
H_rR_2	-2743	1,8	-2745	1,8
H_rR_3	-2743	0,8	-2745	1,9
H_rR_4	-2743	0,9	-2745	1,9
H_WR_1	-2737	1	-2737	1
H_WR_2	-2623	0,7	-2623	0,7
H_WR_3	-2743	0,7	-2743	0,7
H_WR_4	-2743	1	-2743	1

TAB. D.26 – Résultats Mission 3-F1 - replanification critique

Mission 3-F2

Carburant restant : 120 kg

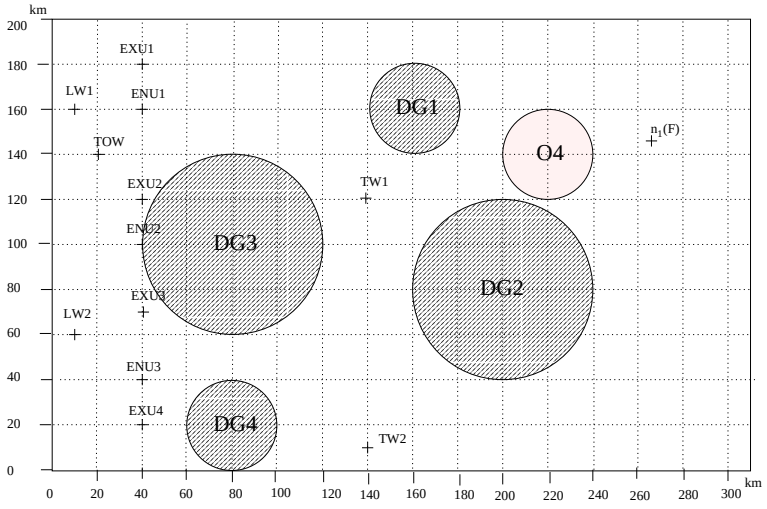


FIG. D.29 – Mission 3 : replanification en fin de mission par changement de la carte des dangers

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-2794	1,1	-2794	1,1
H_1R_2	-2794	1,2	-2794	1,2
H_1R_3	-2794	1,4	-2794	1,4
H_1R_4	-2794	1,4	-2794	1,4
H_iR_1	-2794	1,2	-2794	1,2
H_iR_2	-2794	1,2	-2794	1,2
H_iR_3	-2794	1,4	-2794	1,4
H_iR_4	-2794	1,5	-2794	1,5
H_rR_1	-2794	1,2	-2971	3,1
H_rR_2	-2794	1,2	-2971	3,1
H_rR_3	-2794	1,4	-2971	4,7
H_rR_4	-2794	1,5	-2971	3,3
H_WR_1	-2794	1,2	-2794	1,2
H_WR_2	-2794	1,2	-2794	1,2
H_WR_3	-2794	1,4	-2794	1,4
H_WR_4	-2794	1,6	-2794	1,6

TAB. D.27 – Résultats Mission 3-F2 - replanification nécessaire

Mission 3-F3

Carburant restant : 120 kg

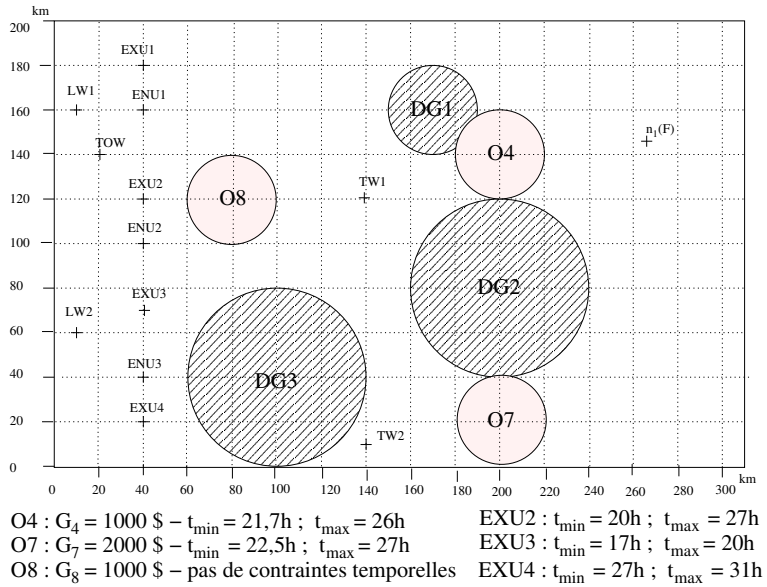


FIG. D.30 – Mission 3 : replanification en fin de mission par changement de la carte des objectifs

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-2968	4,5	-2968	4,5
H_1R_2	-2968	4,5	-2968	4,5
H_1R_3	-2968	2,2	-2968	2,2
H_1R_4	-2968	3,6	-2968	3,6
H_iR_1	-2968	4,5	-2968	4,5
H_iR_2	-2968	4,5	-2968	4,5
H_iR_3	-2968	2,2	-2968	2,2
H_iR_4	-2968	3,6	-2968	3,6
H_rR_1	-2968	4,5	-2968	4,5
H_rR_2	-2968	4,4	-2968	4,4
H_rR_3	-2968	2,2	-2968	2,2
H_rR_4	-2968	3,6	-2968	3,6
H_WR_1	-2968	4,5	-2968	4,5
H_WR_2	-2968	4,5	-2968	4,5
H_WR_3	-2968	2,2	-2968	2,2
H_WR_4	-2968	6,6	-2968	6,6

TAB. D.28 – Résultats Mission 3-F3 - replanification critique

D.5 Mission 4

Carte principale et planification initiale

Carburant embarqué : 400 kg

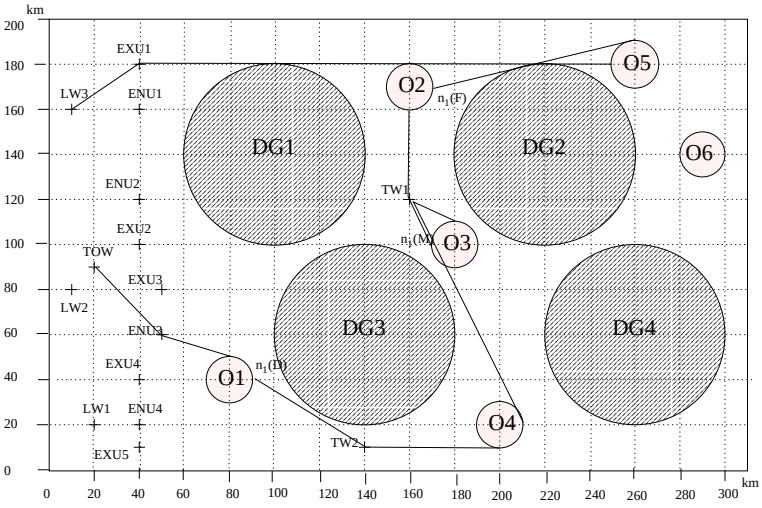


FIG. D.31 – Mission 4 : planification globale, choix des points de replanification

Mission 4-D1

Carburant restant : 250 kg

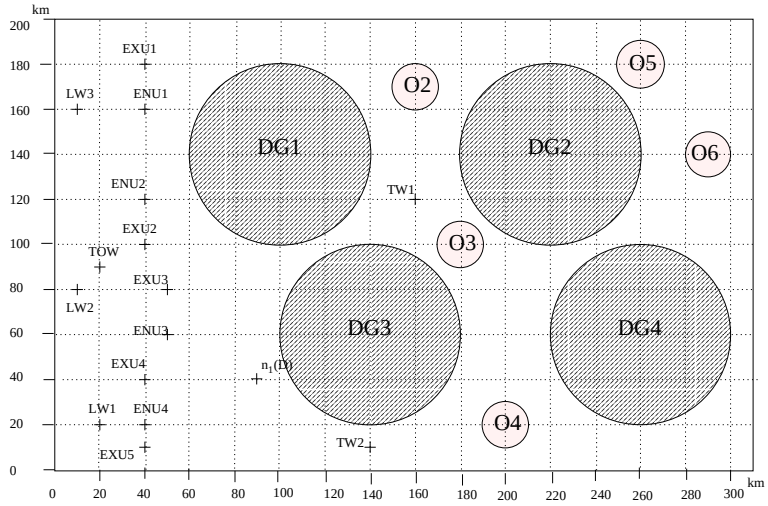


FIG. D.32 – Mission 4 : replanification en début de mission par manque de carburant

Algorithme	Meil. Sol.	
	$\hat{J}$	temps (s)
H_1R_1	-770	4,8
H_1R_2	-770	4,8
H_1R_3	-770	3,4
H_1R_4	-770	3,7
H_iR_1	-770	4,8
H_iR_2	-770	4,7
H_iR_3	-770	3,4
H_iR_4	-770	3,7
H_rR_1	-825	370,1
H_rR_2	-825	211,8
H_rR_3	-825	447,9
H_rR_4	-825	210,4
H_WR_1	-770	4,7
H_WR_2	-770	4,8
H_WR_3	-770	3,4
H_WR_4	-770	4,3

TAB. D.29 – Résultats Mission 4-D1 - replanification éventuelle

Mission 4-D2

Carburant restant : 350 kg

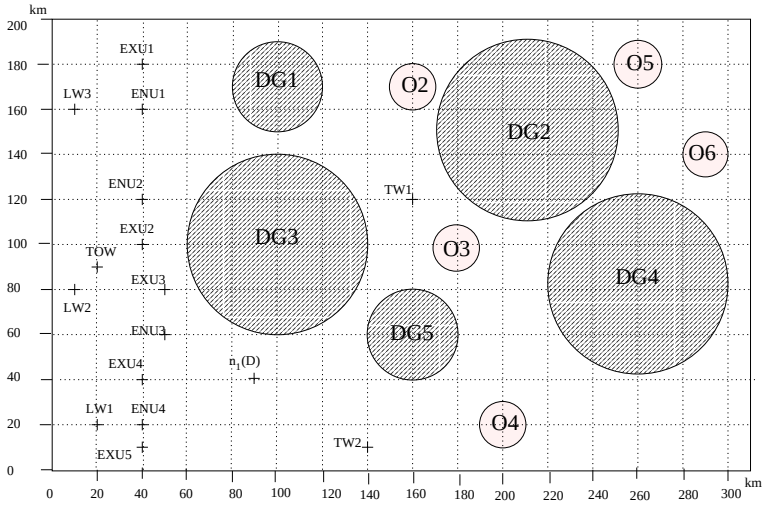


FIG. D.33 – Mission 4 : replanification en début de mission par changement de la carte des dangers

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-302	5,2	-302	5,2
H_1R_2	-302	5,2	-302	5,2
H_1R_3	74	4,7	74	4,7
H_1R_4	74	4,9	74	4,9
H_iR_1	-302	5,3	-302	5,3
H_iR_2	-302	5,2	-302	5,2
H_iR_3	74	4,7	74	4,7
H_iR_4	74	4,9	74	4,9
H_rR_1	-302	5,2	-302	5,2
H_rR_2	-302	5,2	-302	5,2
H_rR_3	74	4,7	74	4,7
H_rR_4	74	4,9	74	4,9
H_WR_1	-302	5,2	-302	5,2
H_WR_2	-302	5,3	-302	5,3
H_WR_3	74	4,8	74	4,8
H_WR_4	74	5,3	74	5,3

TAB. D.30 – Résultats Mission 4-D2 - replanification critique

Mission 4-D3

Carburant restant : 350 kg

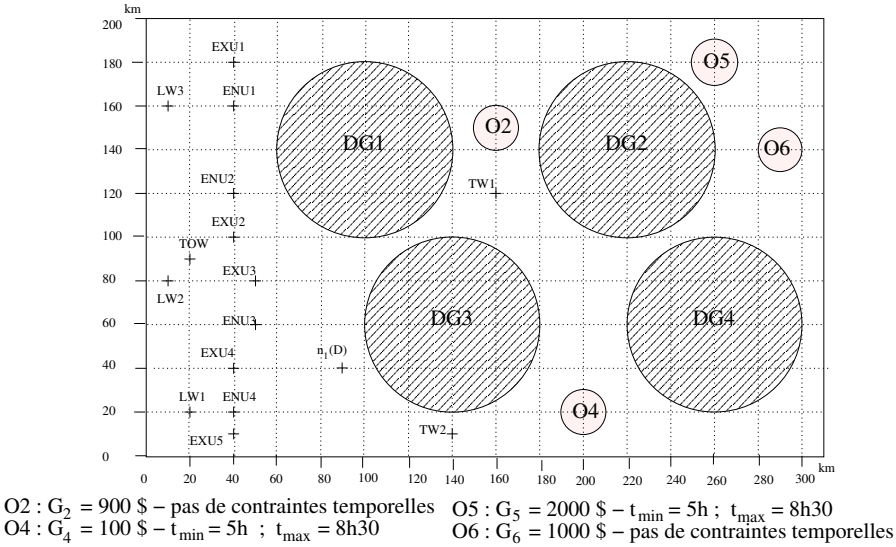


FIG. D.34 – Mission 4 : replanification en début de mission par changement de la carte des objectifs

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	74	4,7	74	4,7
H_1R_2	74	4,7	74	4,7
H_1R_3	74	3,6	74	3,6
H_1R_4	74	3,7	74	3,7
H_iR_1	74	4,7	74	4,7
H_iR_2	74	4,7	74	4,7
H_iR_3	74	3,6	74	3,6
H_iR_4	74	3,7	74	3,7
H_rR_1	74	4,7	74	4,7
H_rR_2	74	4,7	74	4,7
H_rR_3	74	3,6	74	3,6
H_rR_4	74	3,7	74	3,7
H_WR_1	74	4,7	74	4,7
H_WR_2	74	4,7	74	4,7
H_WR_3	74	3,6	74	3,6
H_WR_4	74	4,1	74	4,1

TAB. D.31 – Résultats Mission 4-D3 - replanification critique

Mission 4-M1

Carburant restant : 150 kg

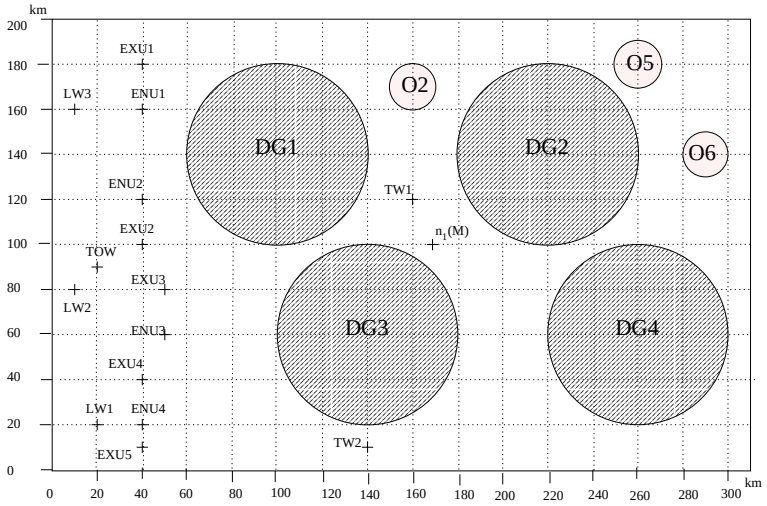


FIG. D.35 – Mission 4 : replanification en milieu de mission par manque de carburant

Algorithme	Meil. Sol.	
	$\hat{J}$	temps (s)
H_1R_1	-815	1,8
H_1R_2	-815	1,8
H_1R_3	-815	1,7
H_1R_4	-815	1,8
H_iR_1	-815	1,8
H_iR_2	-815	1,8
H_iR_3	-815	1,7
H_iR_4	-815	1,8
H_rR_1	-843	35,9
H_rR_2	-843	14,8
H_rR_3	-843	35,9
H_rR_4	-843	14,8
H_WR_1	-815	1,8
H_WR_2	-815	1,8
H_WR_3	-815	1,7
H_WR_4	-815	1,9

TAB. D.32 – Résultats Mission 4-M1 - replanification éventuelle

Mission 4-M2

Carburant restant : 250 kg

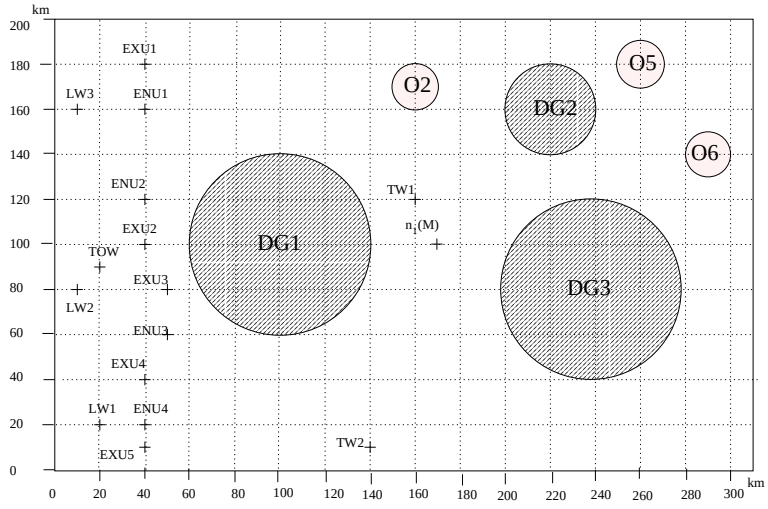


FIG. D.36 – Mission 4 : replanification en milieu de mission par changement de la carte des dangers

Algorithme	Meil. Sol.	
	$\hat{J}$	temps (s)
H_1R_1	-2321	3,6
H_1R_2	-2321	3,6
H_1R_3	-2334	3,1
H_1R_4	-2334	3,3
H_iR_1	-2321	3,6
H_iR_2	-2321	3,6
H_iR_3	-2334	3,1
H_iR_4	-2334	3,3
H_rR_1	-2339	23,7
H_rR_2	-2339	23,8
H_rR_3	-2339	23,2
H_rR_4	-2339	23,4
H_WR_1	-2321	3,6
H_WR_2	-2321	3,6
H_WR_3	-2334	3,1
H_WR_4	-2334	3,6

TAB. D.33 – Résultats Mission 4-M2 - replanification éventuelle

Mission 4-M3

Carburant restant : 250 kg

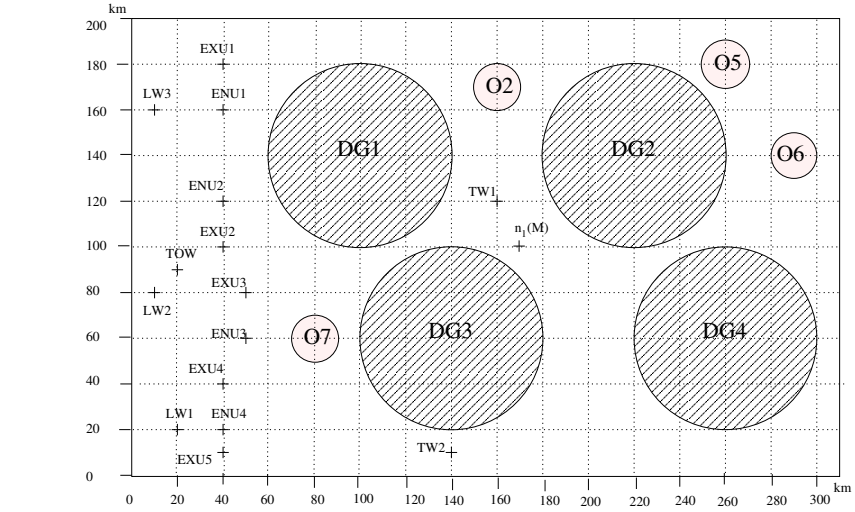


FIG. D.37 – Mission 4 : replanification en milieu de mission par changement de la carte des objectifs

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	447598	16,5	447598	16,5
H_1R_2	447598	16,5	447598	16,5
H_1R_3	-843	4	-843	4
H_1R_4	-843	4,3	-843	4,3
H_iR_1	447598	16,5	447598	16,5
H_iR_2	447598	16,5	447598	16,5
H_iR_3	-843	4	-843	4
H_iR_4	-843	4,3	-843	4,3
H_rR_1	∞	-	∞	-
H_rR_2	∞	-	∞	-
H_rR_3	∞	-	∞	-
H_rR_4	∞	-	∞	-
H_WR_1	447598	16,9	326599	19,8
H_WR_2	180190	13,9	180190	13,9
H_WR_3	-843	4	-843	4
H_WR_4	∞	-	∞	-

TAB. D.34 – Résultats Mission 4-M3 - replanification critique

Mission 4-F1

Carburant restant : 50 kg

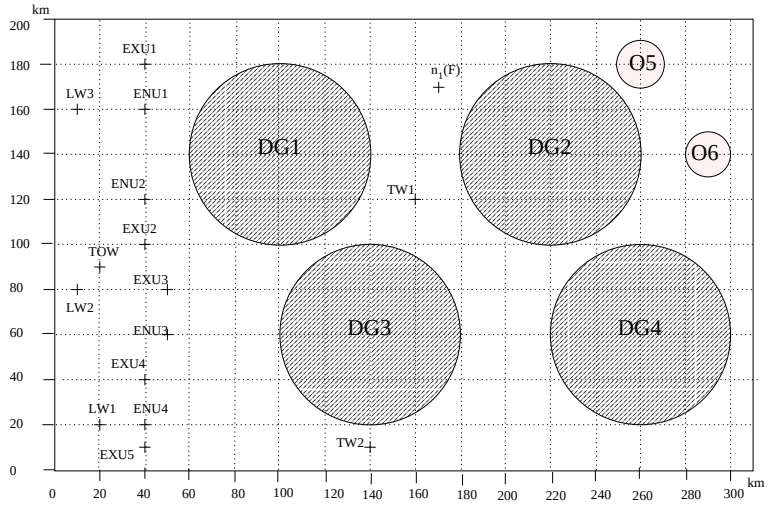


FIG. D.38 – Mission 4 : replanification en fin de mission par manque de carburant

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	180197	1,6	180158	2,3
H_1R_2	180197	1,6	180158	2,5
H_1R_3	180158	1,2	180158	1,2
H_1R_4	180158	1,5	180158	1,5
H_iR_1	180197	1,6	180158	2,3
H_iR_2	180197	1,6	180158	2,5
H_iR_3	180158	1,2	180158	1,2
H_iR_4	180158	1,5	180158	1,5
H_rR_1	180197	4,4	180158	5,5
H_rR_2	180158	3,6	180158	3,6
H_rR_3	180158	1,2	180158	1,2
H_rR_4	180158	1,5	180158	1,5
H_WR_1	180197	1,6	180197	1,6
H_WR_2	180209	1,8	180209	1,8
H_WR_3	180158	1,2	180158	1,2
H_WR_4	180158	2,1	180158	2,1

TAB. D.35 – Résultats Mission 4-F1 - replanification critique

Mission 4-F2

Carburant restant : 210 kg

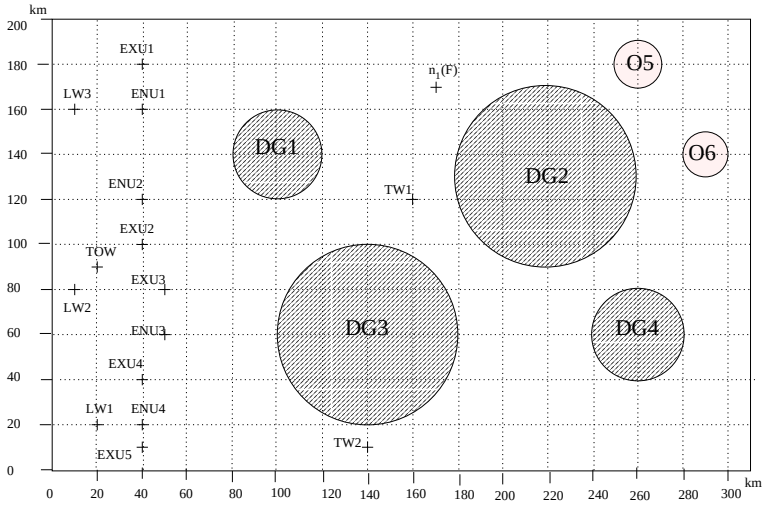


FIG. D.39 – Mission 4 : replanification en fin de mission par changement de la carte des dangers

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-764	1,1	-764	1,1
H_1R_2	-764	1,2	-764	1,2
H_1R_3	-764	0,9	-764	0,9
H_1R_4	-764	0,9	-764	0,9
H_iR_1	-764	1,1	-764	1,1
H_iR_2	-764	1,1	-764	1,1
H_iR_3	-764	1	-764	1
H_iR_4	-764	0,9	-764	0,9
H_rR_1	-764	1,2	-764	1,2
H_rR_2	-764	1,2	-764	1,2
H_rR_3	-764	0,9	-764	0,9
H_rR_4	-764	0,9	-764	0,9
H_WR_1	-764	1,1	-764	1,1
H_WR_2	-764	1,1	-764	1,1
H_WR_3	-764	0,8	-764	0,8
H_WR_4	-764	0,9	-764	0,9

TAB. D.36 – Résultats Mission 4-F2 - replanification nécessaire

Mission 4-F3

Carburant restant : 210 kg

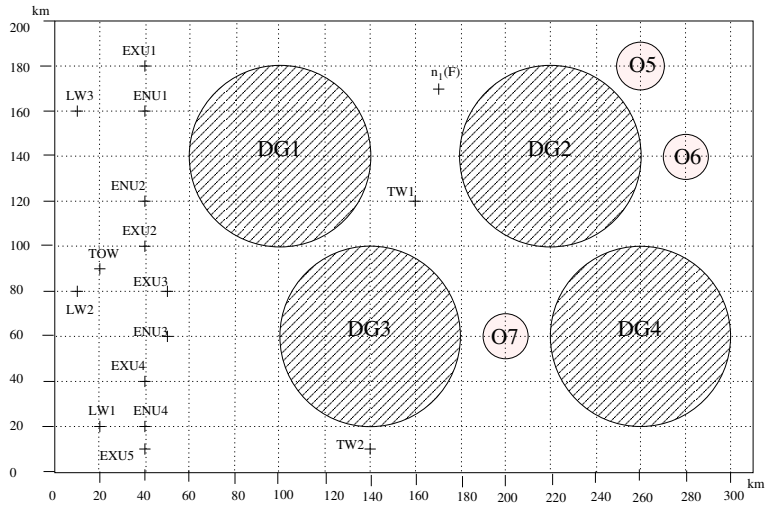


FIG. D.40 – Mission 4 : replanification en fin de mission par changement de la carte des objectifs

Algorithme	1ère Sol. adm.		Meil. Sol.	
	$\hat{J}$	temps (s)	$\hat{J}$	temps (s)
H_1R_1	-2306	4,9	-2312	44,1
H_1R_2	-2306	5,1	-2312	44,5
H_1R_3	-2312	3,3	-2312	3,3
H_1R_4	-2312	3,3	-2312	3,3
H_iR_1	-2306	4,7	-2312	44,5
H_iR_2	-2306	5	-2312	45,2
H_iR_3	-2312	3,3	-2312	3,3
H_iR_4	-2312	3,3	-2312	3,3
H_rR_1	-2306	5,1	-2312	39,1
H_rR_2	-2306	5,1	-2312	25,3
H_rR_3	-2312	3,3	-2312	3,3
H_rR_4	-2312	3,4	-2312	3,4
H_WR_1	-2306	5,1	-2306	5,1
H_WR_2	-2306	5	-2306	5
H_WR_3	-2312	3,3	-2312	3,3
H_WR_4	-2312	3,4	-2312	3,4

TAB. D.37 – Résultats Mission 4-F3 - replanification nécessaire

Bibliographie

[Alami *et al.* 1998] Alami, R. ; Chatila, R. ; Fleury, S. ; Ghallab, M. ; and Ingrand, F. 1998. An architecture for autonomy. *International Journal of Robotics Research* 17(4) :315–337.

[Allo, Guettier, & Lecubin 2001] Allo, B. ; Guettier, C. ; and Lecubin, N. 2001. A demonstration of dedicated constraint-based planning within agent-based architectures for autonomous aircraft. In *ISIC 2001.*

[Barbier & Chanthery 2004] Barbier, M., and Chanthery, E. 2004. Autonomous mission management for unmanned aerial vehicles. *Aerospace Science and Technology* 8 :359–368.

[Barbier, Lemaire, & Toumelin 2001] Barbier, M. ; Lemaire, J. ; and Toumelin, N. 2001. Procedures planner for an A.U.V. In *12th International Symposium on Unmanned Untethered Submersible Technology.*

[Barrouil & Lemaire 1998] Barrouil, C., and Lemaire, J. 1998. An integrated navigation system for a long range A.U.V. In *OCEAN'S98 :"Engineering for Sustainable Use of the Oceans.*

[BEAR 2005] BEAR. 2005. http ://robotics.eecs.berkeley.edu/bear/. (dernière mise à jour).

[Beard *et al.* 2002] Beard, R. W. ; McLain, T. W. ; Goodrich, M. A. ; and Anderson, E. P. 2002. Coordinated target assignment and intercept for unmanned air vehicles. *IEEE Trans. Robotics and Automation* 18(6) :911–922.

[Bellman 1952] Bellman, R. 1952. On the theory of dynamic programming. *National Academy of Sciences* 38 :716–719.

[Bernstein *et al.* 2001] Bernstein, D. ; Zilberstein, S. ; Washington, R. ; and Bresina, J. 2001. Planetary rover control as a markov decision process. In *6th International Symposium on Artificial Intelligence, Robotics and Automation in Space.*

[Blaer & Allen 2003] Blaer, P., and Allen, P. K. 2003. TopBot : Automated network topology detection with a mobile robot. In *International Conference on Robotics and Automation.*

[Blum & Furst 1995] Blum, A. L., and Furst, M. L. 1995. Fast planning through planning graph analysis. In *14th International Joint Conference on Artificial Intelligence (IJCAI95)*, 1636–1642.

[Boiffier 2001] Boiffier, J.-L. 2001. *Dynamique du Vol de l'avion, Notes de cours.* Toulouse : SupAéro-Département Aéronefs.

[Bonasso *et al.* 1997] Bonasso, R. P. ; Firby, R. J. ; Gat, E. ; Kortenkamp, D. ; Miller, D. ; and Slack, M. 1997. Proven three-tiered architecture for programming autonomous robots. *Journal of Experimental and Theoretical Artificial Intelligence* 9(2).

[Bonet & Geffner 2004] Bonet, B., and Geffner, H. 2004. Planning as heuristic search. In *14th International Conference on Automated Planning and Scheduling.*

[Bouali 1996] Bouali, D. 1996. *Heuristiques et Approche Polyédrale du Problème du Voyageur de Commerce International.* Ph.D. Dissertation, Institut National Polytechnique de Grenoble.

[Brooks 1986] Brooks, R. A. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* 2 :14–23.

[Chandler, Patcher, & Rasmussen 2001] Chandler, P. ; Patcher, M. ; and Rasmussen, S. 2001. UAV cooperative control. In *American Control Conference.*

[Chanthery, Barbier, & Farges 2004] Chanthery, E. ; Barbier, M. ; and Farges, J.-L. 2004. Mission planning for autonomous aerial vehicles. In *IAV2004 - 5th IFAC Symposium on Intelligent Autonomous Vehicles.*

[CMU 1998] CMU. 1998. www.cs.cmu.edu/afs/cs/project/chopper/www/. (dernière mise à jour).

[COMETS 2005] COMETS. 2005. http ://www.comets-uavs.org/. (dernière mise à jour).

[Dalgalarrondo 2003] Dalgalarrondo, A. 2003. A propos de l'autonomie des robots. Technical report, Délégation Générale pour l'Armement, Centre Technique d'Arcueil.

[Damiani, Verfaillie, & Charmeau 2004] Damiani, S. ; Verfaillie, G. ; and Charmeau, M.-C. 2004. Autonomous management of an earth watching satellite. In *IAV2004 - 5th IFAC Symposium on Intelligent Autonomous Vehicles.*

[Dantzig 1963] Dantzig, G. B. 1963. *Linear Programming and Extensions.* Princeton University Press. Princeton, NJ.

[Do & Kambhampati 2001] Do, M. B., and Kambhampati, S. 2001. Sapa : A domain-independent heuristic metric temporal planner. In *European Conference on Planning.*

[Do & Kambhampati 2004] Do, M. B., and Kambhampati, S. 2004. Partial satisfaction (over-subsription) planning as heuristic search. In *5th International Conference on Knowledge Based Computer Sciences (KBCS).*

[Doherty *et al.* 2000] Doherty, P. ; Granlund, G. ; Kuchcinski, K. ; Sandewall, E. ; Nordberg, K. ; Skarman, E. ; and Wiklund, J. 2000. The WITAS unmanned aerial vehicle project. In Horn, W., ed., *European Conference on Artificial Intelligence*, 747–755.

[Fargeon 1993] Fargeon, C. 1993. *Robotique mobile.* Teknea. chapter Introduction : émergence d'une science, 1–9.

[Feillet, Dejax, & Gendreau 2001] Feillet, D. ; Dejax, P. ; and Gendreau, M. 2001. Traveling salesman problems with profits : an overview. In *ORP*^{3} *2001.*

[Ferguson & Stentz 2005] Ferguson, D., and Stentz, A. 2005. The delayed D^* algorithm for efficient path replanning. In *IEEE International Conference on Robotics and Automation.*

[Figueira, Mousseau, & Roy 2005] Figueira, J. ; Mousseau, V. ; and Roy, B. 2005. ELECTRE methods. In Figueira, J. ; Greco, S. ; and Ehrgott, M., eds., *Multiple Criteria Decision Analysis : State of the Art Surveys*. Boston, Dordrecht, London : Springer Verlag. 133–162.

[Fikes & Nilsson 1971] Fikes, R., and Nilsson, N. 1971. STRIPS : A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2 :189–208.

[Firby 1987] Firby, R. J. 1987. An investigation into reactive planning in complex domains. In *National Conference on Artificial Intelligence*, 202–206.

[Focacci & Godard 2002] Focacci, F., and Godard, D. 2002. A practical approach to multi criteria optimization problems in constraint programming. In *CP-AI-OR*.

[Fox & Long 2003] Fox, M., and Long, D. 2003. PDDL2.1 : An extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res. (JAIR)* 20 :61–124.

[Frank & Kurklu 2003] Frank, J., and Kurklu, E. 2003. SOFIA's choice : Scheduling observations for an airborne observatory. In *13th International Conference on Automated Planning & Scheduling*.

[Frank & Wolfe 1956] Frank, M., and Wolfe, P. 1956. *An Algorithm for quadratic programming*, volume 3. Naval Research Logistic Quaterly.

[Galindo, Fernández-Madrigal, & González 2004] Galindo, C. ; Fernández-Madrigal, J. ; and González, J. 2004. Improving efficiency in mobile robot task planning through world abstraction. *IEEE Transaction on Robotics and Automation* 20(4) :677–690.

[Gat 1992] Gat, E. 1992. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *National Conference on Artificial Intelligence (AAAI)*.

[Gat 1997] Gat, E. 1997. ESL : A language for supporting robust plan execution in embedded autonomous agents. In *IEEE Aerospace Conference*.

[Ghallab, Nau, & Traverso 2004] Ghallab, M. ; Nau, D. ; and Traverso, P. 2004. *Automated planning : theory and practice*. Morgan Kaufmann Publishers.

[Gondran & Minoux 1995] Gondran, M., and Minoux, M. 1995. *Graphes et Algorithmes*. Eyrolles.

[Hassoun 1997] Hassoun, M. 1997. *Contrôle d'exécution des mouvements d'un robot mobile : application à l'assistance à la conduite automobile*. Ph.D. Dissertation, Institut National Polytechnique de Grenoble, Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle.

[Hayes-Roth 1995] Hayes-Roth, B. 1995. An architecture for adaptive intelligent systems. *Artificial Intelligence : Special Issue on Agents and Interactivity* 72 :329–365.

[Helsgaun 2000] Helsgaun, K. 2000. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research* 126(1) :106–130.

[Herschel 1862] Herschel, A. S. 1862. Sir Wm. Hamilton's Icosian Game. *Quart. J. Pure Applied Math* 5(305).

[Hoffmann & Nebel 2001] Hoffmann, J., and Nebel, B. 2001. The FF planning system : Fast plan generation through heuristic search. *JAIR* 14 :253–302.

[Hoffmann 2002] Hoffmann, J. 2002. Extending FF to numerical state variables. In *15th European Conference on Artificial Intelligence.*

[Hummingbird 1999] Hummingbird. 1999. http ://sun-valley.stanford.edu/users/heli/. (dernière mise à jour).

[Kantor & Rosenwein 1992] Kantor, M., and Rosenwein, M. 1992. The orienteering problem with time windows. *Journal of Operational Research Society* 43(6) :629–635.

[Keller 1989] Keller, C. P. 1989. Algorithms to solve the orienteering problem : a comparison. *European Journal of Operational Research* 41 :224–231.

[Koenig, Likhachev, & Furcy 2004] Koenig, S. ; Likhachev, M. ; and Furcy, D. 2004. Lifelong planning A*. *Artif. Intell.* 155(1-2) :93–146.

[Kramer & Giulianno 1997] Kramer, L., and Giulianno, M. 1997. Reasoning about and scheduling linked HST observations with spike. In *International Workshop on Planning and Scheduling for Space.*

[Laporte & Nobert 1983] Laporte, G., and Nobert, Y. 1983. Generalized traveling salesman problem through n sets of nodes : an integer programming approach. *INFOR* 21 :61–75.

[Latombe 1991] Latombe, J. C. 1991. *Robot Motion Planning.* Boston, MA : Kluwer Academic Publishers.

[Lemai, Ingrand, & Gallien 2005] Lemai, S. ; Ingrand, F. ; and Gallien, M. 2005. Embedded decision in the LAAS architecture. In *ICRA Workshop on "Principles and Practice of Software Development in Robotics" (SDIR2005).*

[Lemaitre & Verfaillie 1997] Lemaitre, M., and Verfaillie, G. 1997. Daily management of an earth observation satellite : comparison of ILOG solver with dedicated algorithms for valued constraint satisfaction problems. In *Proccedings of the Third ILOG International Users Meeting.*

[Likhachev *et al.* 2005] Likhachev, M. ; Ferguson, D. ; Gordon, G. ; Stentz, A. ; and Thrun, S. 2005. Anytime dynamic A* : An anytime, replanning algorithm. In *International Conference on Automated Planning and Scheduling (ICAPS).*

[L.Kavraki *et al.* 1994] L.Kavraki ; Svestka, P. ; Latombe, J.-C. ; and Overmars, M. 1994. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. Technical Report CS-TR-94-1519.

[Luzeaux & Dalgalarrondo 2001] Luzeaux, D., and Dalgalarrondo, A. 2001. HARPIC : an hybrid architecture based on representations, perception and intelligent control : a way to provide autonomy to robots. In *Intelligent Systems Design and Applications conference.*

[Murata 1989] Murata, T. 1989. Petri nets : properties, analysis and applications. In *Proceedings of the IEEE*, number 77(4), 541–580.

[Nesnas *et al.* 2003] Nesnas, I. ; Wright, A. ; Bajracharya, M. ; Simmons, R. ; Estlin, T. ; and Kim, W. S. 2003. CLARAty : An architecture for reusable robotic software. In *SPIE Aerosense Conference.*

[Nigenda & Kambhampati 2000] Nigenda, X. N. R., and Kambhampati, S. 2000. AltAlt : Combining the advantages of graphplan and heuristic state search. Technical report, Arizona State University, US.

[Nilson 1971] Nilson, N. J. 1971. *Problem Solving Methods in Artificial Intelligence*. New-York : McGraw-Hill.

[NIVAS 2003] NIVAS. 2003. http ://www.cert.fr/dcsd/cd/membres/barrouil/nivas/. (dernière mise à jour).

[ONR 2000] ONR. 2000. *Review of ONR's Uninhabited Combat Air Vehicles Program*. National Academy Press, Washington D.C. Committee for the Review of ONR's Uninhabited Combat Air Vehicles Program, Naval Studies Board, National Research Council.

[OSD 2002] OSD. 2002. Unmanned aerial vehicles roadmap 2002-2027. Technical report, Office of the Secretary of Defence, USA.

[Pai & Reissel 1998] Pai, D. K., and Reissel, L.-M. 1998. Multiresolution rough terrain motion planning. In *IEEE Transactions on Robotics and automation*, 14(1) : 19–33.

[Pearl 1988] Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann.

[Pettersson & Doherty 2004] Pettersson, P., and Doherty, P. 2004. Probabilistic roadmap based path planning for autonomous unmanned aerial vehicle. In *Workshop on Connecting Planning and Theory with Practice. ICAPS'2004*.

[Puterman 1994] Puterman, M. 1994. *Markov Decision Processes*. John Wiley and Sons, INC.

[Rao & Iyengar 1990] Rao, N., and Iyengar, S. 1990. Autonomous robot navigation in unknown terrains : visibility graph based methods. *IEEE Trans. Systems Man Cybernet.* 20(6) :1443–1449.

[RESSAC 2002] RESSAC. 2002. http ://www.cert.fr/dcsd/ressac/. (dernière mise à jour).

[Richards & How 2002] Richards, A., and How, J. P. 2002. Aircarft trajectory planning with collision avoidance using mixed integer linear programming. In *IEEE 2002 ACC*, 1936–1941.

[Rosenblatt 1995] Rosenblatt, J. 1995. DAMN : A distributed architecture for mobile navigation thesis summary. In *AAAI Spring Symp. on Lessons Learned from Implememted Software Architectures for Physical Agents*.

[Roy 1968] Roy, B. 1968. Classement et choix en présence de points de vue multiples (la méthode ELECTRE). *RIRO* (8) :57–75.

[Roy 1991] Roy, B. 1991. The outranking approach and the foundations of ELECTRE methods. *Theory and Decision* 31(1) :49–73.

[Schesvold *et al.* 2003] Schesvold, D. ; Tang, J. ; Ahmed, B. M. ; Altenburg, K. ; and Nygard, K. E. 2003. POMDP planning for high level UAV decisions : search vs. strike. In *16th International Conference on Computer Applications in Industry and Engineering*.

[Simmons 1994] Simmons, E. 1994. Structured control for autonomous robots. *IEEE Transactions on Robotics and Automation* 10 :34–43.

[Smith 2004] Smith, D. E. 2004. Choosing objectives in over-subscription planning. In *14th International Conference on Automated Planning and Scheduling*.

[Stentz 1994] Stentz, A. 1994. Optimal and efficient path planning for partially-known environments. In *IEEE International Conference on Robotics and Automation (ICRA'94)*, volume 4, 3310–3317.

[Stentz 1995] Stentz, A. 1995. The focussed D^* algorithm for real-time replanning. In *International Joint Conference on Artificial Intelligence*.

[Stentz 2002] Stentz, A. 2002. CD^* : A real-time resolution optimal re-planner for globally constrained problems. In *AAAI-02*.

[Teichteil & Fabiani 2005] Teichteil, F., and Fabiani, P. 2005. Influence of modeling structure in probabilistic sequential decision problems. *RAIRO Operations Research* to appear.

[Thayse 1990] Thayse, A. 1990. *Approche logique de l'intelligence artificielle*, volume 1 of Informatique. Dunod.

[UAVRF 2005] UAVRF. 2005. http ://controls.ae.gatech.edu/uavrf/. (dernière mise à jour).

[Vallée & Zielniewicz 1994] Vallée, D., and Zielniewicz, P. 1994. *ELECTRE III-IV, version 3.x - Aspects méthodologiques*. Université de Paris-Dauphine. Document du LAMSADE no 85.

[van den Briel *et al.* 2004] van den Briel, M. ; Nigenda, R. ; Do, M. ; and Kambhampati, S. 2004. Effective approaches for partial satisfaction (over-subscription) planning. In *AAAI*, 562–569.