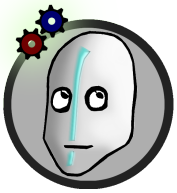

Programme CONTINT
Edition 2011

Projet ROBOERGOSUM
Deliverable D3.1

Acronyme	RoboErgoSum
Nom du projet	Robots conscients
Référence	ANR-12-CORD-0030-02
Numéro de la tâche	T3
Nom de la tâche	Self-awareness and deliberation
Numéro du rapport	D3.1
Titre du rapport	Technical report on the proposed artificial motivation algorithm
Partenaires	*ISIR*, LAAS
Date	T0+24

Table des matières

1 Résume du deliverable / summary

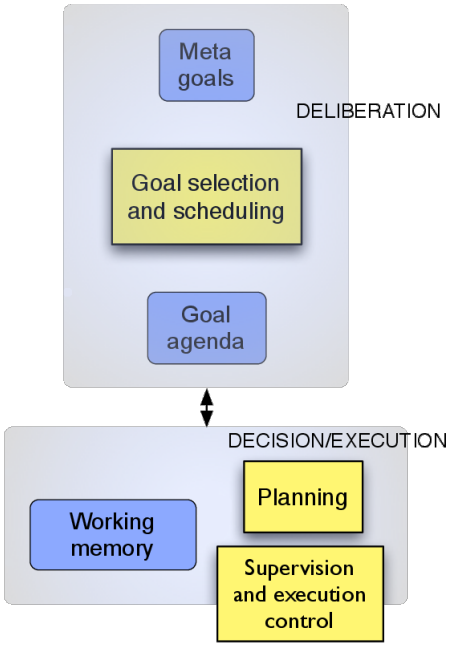

Figure 1 – *Extrait de l'architecture*

Dans le cadre de la tâche 3, nous souhaitons doter le robot d'une capacité à prendre des décisions sans requête explicite d'un humain ou en pur réflexe à un stimulus extérieur. Pour y arriver, nous explorerons la notion de motivation, comme représentation originale d'un méta-objectif. Les motivations auront pour rôle de représenter les méta-objectifs en définissant un ensemble d'objectifs simples, pouvant être planifiés grâce à un planificateur MDP. Le planificateur fournira pour chaque objectif un plan d'action (une tâche), ainsi qu'un ensemble de données statistiques sur le comportement du plan.

Ces éléments prendront leur place dans une architecture cognitive, accompagnés d'un mécanisme délibératif, placé au sommet de l'architecture. Ce dernier aura pour rôle de rechercher à l'aide des motivations, des tâches et des informations sur les tâches, un agenda de buts. Cet agenda prendra la forme d'un arbre de tâches, donnant pour chaque possible situation future, la prochaine tâche à accomplir.

2 Introduction

Nous présenterons dans ce rapport l'avancement des travaux de la tâche 3, intitulée "Self-awareness and deliberation". Cette tâche concerne l'étude de la capacité pour un robot à prendre des décisions de façon autonome, sans réagir directement à des stimulus et sans répondre simplement à une requête. Nous proposerons donc ici une solution à cette problématique. Ce travail se place majoritairement dans la partie "Délibération" de l'architecture robotique que nous nous proposons d'étudier dans ce projet. Nous allons cependant étudier et définir la planification de la partie "Décision/Exécution" afin de nous donner les outils dont nous aurons besoin. Nous nous plaçons dans une représentation probabiliste du monde, avec l'ajout de ressources, comme le temps et le niveau de batterie.

Dans le domaine qui nous intéresse, celui du paradigme de la décision stochastique, la représentation des objectifs le plus couramment rencontré est réduit à son minimum : une fonction de récompense. Cette fonction est plus pratique qu'expressive et ne peut être raffinée qu'en ajoutant au modèle des variables concernant directement les buts. Ces rajouts ont pour conséquence d'augmenter l'espace d'état au modèle du monde, et de rendre confuse la notion d'état du monde, puisqu'une variable concernant les objectifs sera exprimée avec la même sémantique que celle concernant la position du robot par exemple. En opposition à ces techniques classiques, nous souhaitons bâtir deux modèles distincts : un **modèle opérationnel**, basé uniquement sur ce sur quoi le robot peut agir, et un **modèle intentionnel**, permettant de modéliser finement les objectifs et les méta-objectifs.

Nous souhaitons donc développer un modèle intentionnel capable représenter des méta-objectifs, c'est-à-dire des objectifs permanents ou fortement structurés, ne pouvant pas être atteints par l'application d'un seul plan d'action. Il pourra par exemple s'agir d'objectifs bouclant sur eux-même (tâche répétitive), d'objectifs n'ayant pas de fin (le maintien du niveau de batterie). Une **motivation** aura pour rôle de décrire un méta-objectif de manière structurée et durable. Elle livrera un ensemble de **déclencheurs** guidant la progression de la motivation. Ces déclencheurs représenteront différentes conditions pouvant faire évoluer l'état de la motivation et seront accompagnés d'une valeur d'utilité, qualifiant leur impact positif ou négatif sur le méta-objectif. Parmi les différentes conditions, des **objectifs simples** pourront être définis. Les déclencheurs pourront être provoqués par l'utilisation de **tâches**, des plans d'actions accomplissant des objectifs simples

demandés.

Afin de pouvoir créer les tâches à partir d'objectifs simples, nous aurons besoin du modèle opérationnel et d'un planificateur MDP chargé de calculer des politiques. Le modèle du monde utilisé ici sera purement opérationnel, ne représentant que ce sur quoi le robot peut agir et les actions physiques du robot. Les objectifs simples des motivations devront s'exprimer à l'aide d'action à exécuter et les politiques crées devront finir sur l'exécution d'une action demandée. De plus, si d'habitude une politique est un produit fini destiné à être uniquement exécuté, elle représente ici une tâche et donc le moyen d'atteindre un déclencheur. Nous utiliserons plus tard un modèle où les tâches des motivations tiendrons lieu d'action, nous souhaiterons donc, à la manière d'une fonction de transition, connaître le comportement des tâches calculées. Pour chaque politique sera donc calculée la probabilité de finir sur chaque transition finale possible ainsi que le coût en ressource moyen pour y parvenir.

Nous aurons alors une représentation des différentes intentions du robot, les motivations, et un moyen de faire évoluer ces intentions, les politiques calculées par le planificateur. Avec la capacité de prévoir l'évolution du modèle après l'exécution d'une tâche, nous aurons les informations nécessaires pour choisir le méta-objectif que nous voulons faire progresser à un instant donné. Nous créerons un modèle dont les états seront seront des couples état du monde - état des motivations,les actions et transitions seront déterminées à partir des tâches. En partant d'une situation donnée, nous calculerons un **arbre des tâches**, donnant pour chaque déroulement possible, la prochaine tâche à jouer, jusqu'à un horizon de temps donné.

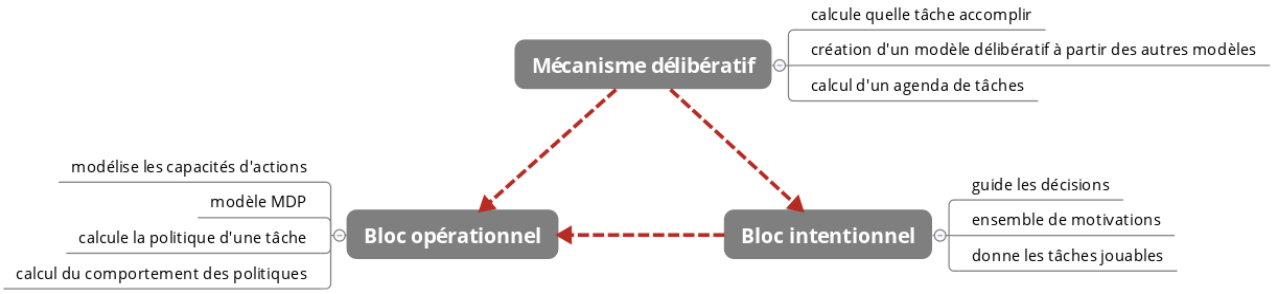

Figure 2 – *Organisation de l'architecture*

3 Modèle du monde

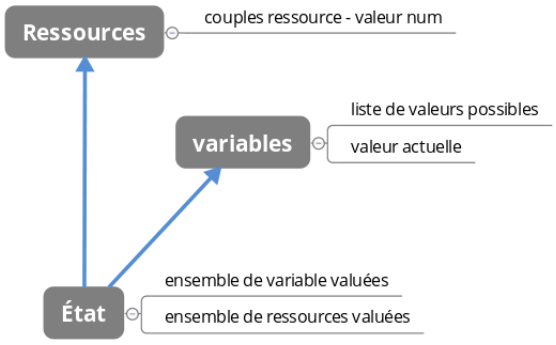

Figure 3 – *Composition d'un état du monde*

Le modèle du monde utilisé par la suite est le suivant :
- S : ensemble d'états (un état est noté ws)
- A : ensemble d'actions (une action est notée a)
- T : fonction de transition (une transition est notée tr ou (ws, a, ws'))
- C_r : fonction de coût de la ressource r

Un état est composé de différentes variables, dont les valeurs sont concaténées pour donner un identifiant unique, ainsi que de la valeur numérique associée à chaque ressource r.

Les actions possèdent une précondition sur l'état du monde informant si elle sont jouables à partir d'un état donné. La valeur des ressources ne peut conditionner une action.

La fonction de transition donne pour chaque (ws, a, ws') la probabilité de passer de ws à ws' en utilisant l'action a, sans prise en compte des ressources.

La fonction de coût C_r donne pour chaque transition tr et pour la ressource r l'évolution moyenne de la ressource pour cette transition.

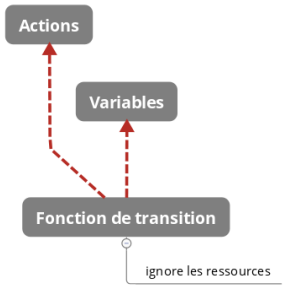

Figure 4 – *Fonction de transition*

Nous ne voulons dans notre modèle que l'aspect purement opérationnel du monde. Imaginons par exemple un monde très simple où un agent peut se déplacer sur une grille et deux points, A et B, sur cette grille. Si notre objectif est de faire des aller-retours entre A et B, nous auront tendance à créer deux variables : la position de notre agent et le prochain point à atteindre, s'agissant du seul moyen de créer une politique unique permettant de créer une politique donnant le comportement souhaité.
Ici, nous allons chercher à retirer tout élément intentionnel du modèle du monde. Nous allons ne garder que la variable de la position. L'objectif de faire des aller-retours étant, en fait, toujours réalisable grâce à deux politique, jouées alternativement : aller à A, aller à B.

Présentons pour finir le modèle du monde de l'exemple que nous utiliserons. Il s'agit d'un ensemble de pièces et de couloirs dans lesquels notre agent, un robot, se déplace. Dans cette représentation, le robot se positionne dans une des cellules grises, délimitée par un mur (trait noir) ou une porte (verte ou rouge), et les grandes pièces ont été découpées en parcelles (les pointillés). Le robot peut se déplacer d'une case à une autre case adjacente reliée directement par des pointillés ou une porte ouverte (en vert). Certaines portes peuvent être fermées, ici la porte sud entre le corridor A et le corridor B. Le robot n'a pas la possibilité d'ouvrir ou de fermer une porte. Les carrés notés 1 ou 2 sont des emplacements disponibles pour objets.

À noter que le robot possède deux cases également, ses deux mains, représentées en bas et chacune occupée par un objet (un circuit imprimé et une puce électronique). Les objets dont la position est inconnue sont placés sous "Unknown positions". Le robot peut prendre un objet dans la main de son choix, si elle est libre, ou déposer un objet qu'il tient dans un emplacement vide de la case où il se trouve. L'action que le robot est en train d'entreprendre est représenté en rouge. Ici, le robot cherche à se déplacer sur la gauche. Pour finir, la ressource du temps n'est pas représentée, mais le niveau de batterie est indiqué via la jauge à gauche.

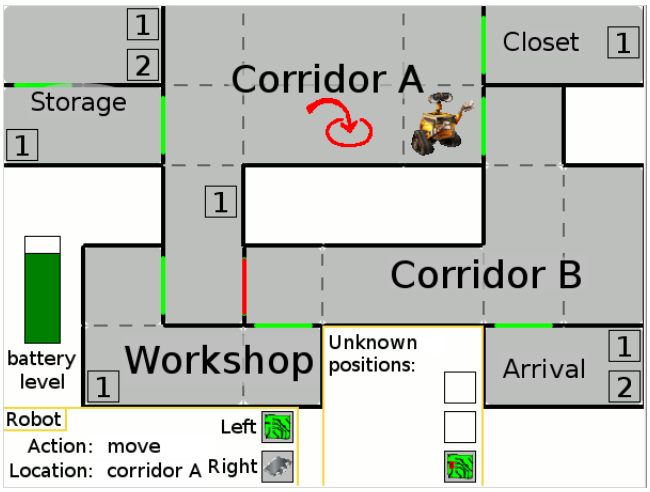

Figure 5 – *Représentation du modèle de notre exemple*

4 Planificateur

Le planificateur utilisé se base sur un planificateur MDP classique. Il démarre avec un modèle du monde et avec une fonction de récompense R vide, renvoyant $R(ws, a, ws') = 0$ pour toute transition (ws, a, ws'). Il ajoute un état supplémentaire à l'ensemble d'état, un puits à partir duquel seule une action n'ayant aucun effet est jouable. Son fonctionnement est celui d'un service auquel nous pouvons demander de calculer des politiques en lui donnant en entrée la spécification de la politique souhaitée. Le service renvoie alors une estimation sur le comportement de la politique, et stocke ensuite la dite politique ainsi que l'estimation.

Le rôle du planificateur est de prendre en entrée un ensemble de transitions, dites finales, abrégées tf, accompagnées de récompenses, puis de créer une politique π. À chaque demande, la fonction de récompense sera renseignée en allouant à chaque transition finale donnée en entrée la récompense correspondante fournie. De plus, chaque transition a son état de fin redirigé vers le puits créé afin de garantir que la politique stoppe dès qu'une transition finale aura été jouée.

Ne disposant pas de planificateur capable de prendre facilement en considération les ressources telles que nous les avons définies, nous disposons d'une fonction de transition les ignorant (leur valeur ne peut donc pas avoir d'incidence sur la jouabilité ou la finalité d'une action dans le modèle). Le calcul de la politique, effectué grâce à l'algorithme Value Iteration, ne pourra donc pas spécifier de ressources à optimiser. Cependant, que nous disposions ou non d'un moyen de prendre en compte les ressources dans le calcul d'une politique, nous verrons que nous serons en mesure de connaître l'évolution des ressources lors de l'utilisation des politiques.

Une fois calculée, et afin de donner des estimations quant au résultat de l'utilisation de la politique, nous allons modéliser la politique obtenue comme une chaîne de Markov. Grâce à la fonction de transition T et de coût C_r, nous serons en mesure de donner $Pr(\pi, ws, tr)$, la probabilité depuis ws de finir sur la transition tr, ainsi que $C_r(\pi, ws, tr)$, le coût moyen pour la ressource r en finissant sur la transition tr en utilisant la politique π.

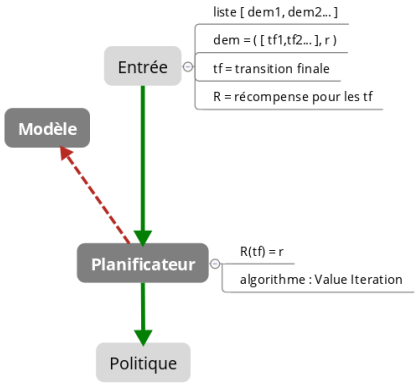

Figure 6 – *Schéma du planificateur*

5 Motivation

Nous souhaitons donner la définition et une modélisation d'un méta-objectif, respectant un certain nombre d'impératifs : pouvoir décrire un objectif comme un fonctionnement, formaliser des méta-objectifs sans fin et les décomposer en sous-objectifs. Avec un modèle opérationnel donné, le but est d'avoir un ensemble fixe d'éléments homogènes donnant des objectifs simples à atteindre dans le modèle opérationnel, et permettant de décrire toutes les intentions actuelles et à venir. Les méta-objectifs ainsi modélisés seront appelées motivations.

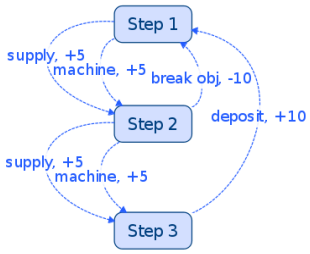

Figure 7 – *motivation Usinage*

La modélisation retenue est celle des **automates finis**. Nous avons donc une notion d'état (nous les nommerons état de la motivation ou état motivationnel, abrégé ms) nous permettant de préciser à partir de quel moment un méta-objectif devient actif ou de le diviser en sous-buts. Les transitions entre états représenteront les conditions pour passer d'un état à un autre, nous les appellerons **déclencheurs**. Ces déclencheurs s'activeront en fonction de l'évolution de l'état du monde, celui décrit dans le modèle opérationnel. Ils détecteront : soit une ou plusieurs transitions finales tf données sans prendre en compte les ressources, soit le niveau des ressources et s'activeront quand un seuil donné sera dépassé. Une récompense accompagnera chaque déclencheur, afin de quantifier l'utilité d'atteindre le déclencheur pour la motivation. Lorsqu'un déclencheur placé entre l'état courant de la motivation et un second état se déclenche, l'état de la motivation change pour se dernier, et donnant accès à de nouveaux déclencheurs.

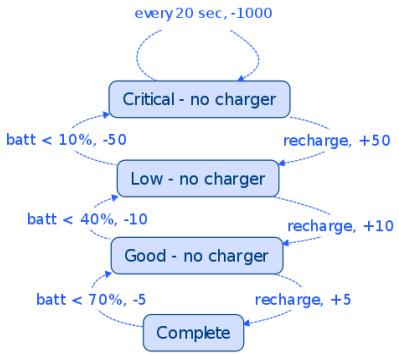

Figure 8 – *motivation Survie*

Nous allons présenter ci-dessous les motivations utilisées dans notre exemple. Notons que pour des raisons de lisibilité, les déclencheurs sont résumés par une courte expression. Le déclencheur *Deposit* par exemple consiste à atteindre la transition $([robotenN, objet5enmain], deposit, [robotenN])$.

La motivation usinage représente une tâche sans fin consistant à assembler des pièces pour construire un objet final. Les pièces peuvent être assemblées par le robot avec un risque d'échec, ou par une autre machine sans risque d'échec. Il y a donc trois états de la motivations, correspondant à trois sous-objectifs : construire un objet intermédiaire, finir l'objet final et rendre l'objet final à un endroit prédéfini. Le déclencheur *Breakobj* est atteint si le robot casse une pièce critique lors de l'action *Machine*, à l'étape 2. Les déclencheurs *Supply*, *Machine* et *Deposit* apportent une récompense positive, *Breakobj* une récompense négative.

L'objectif de la motivation Survie est de maintenir le niveau de batterie au dessus d'un seuil critique et de favoriser un haut niveau d'énergie. Sur la représentation de la motivation ci-contre, nous pouvons voir quatre niveaux de batterie, correspondant aux intervalles $0-10\%$, $10-40\%$, $40-70\%$ et $70-100\%$. Les transitions montrent les conditions sur les transitions (ws, a, ws') pour passer d'un état de motivation ms à un autre, ainsi que la récompense qui l'accompagne. Il faut donc utiliser l'action *recharger* pour passer de *Critical* à *Low*, et l'état passe de *Complete* à *Good* si en fin de transition, le niveau d'énergie de la batterie passe sous les 70%. Recharger la batterie apporte une récompense positive, tandis que passer à un niveau de batterie inférieur ou rester dans le niveau *Critical* apporte une récompense négative.

La motivation du cycle ouverture/fermeture représente la situation où le robot peut travailler 24 heures par jour, mais où certaines parties des locaux sont interdites au robot lorsque les locaux sont fermés. Dans notre exemple, il est interdit, la fermeture passée, d'aller dans le *corridorA* et les pièces *Storage* et *Closet*. Cela veut dire concrètement que les deux déplacement menant de *corridorB* à *corridorA* ainsi qu'entre *Workshop* et *CorridorA* deviennent pénalisants. Le robot ne connaissant pas l'heure de fermeture des locaux, il en est prévenu 15 minutes avant. Le passage de l'état *Open* à *ImpendingClosure* et de *Closed* à *Open* n'est pas spécifié sur le schéma car le robot n'a aucun moyen d'intervenir sur ce changement. Le déclencheur passant de *ImpendingClosure* à *Closed* n'apporte aucune récompense, mais passer dans la zone nord pendant la nuit apporte une récompense négative.

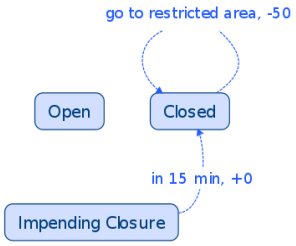

Figure 9 – *motivation Cycle ouverture / fermeture*

6 Délibération

Une fois les modèles opérationnel et intentionnel définis, ainsi qu'en possession de la capacité de planifier des politiques et d'estimer leur comportement, nous allons réunir toutes ces informations pour créer un système délibératif.

En premier lieu, nous allons créer la notion d'état global, gs. La différenciation entre état du monde et état de motivation nous oblige à créer un état plus large rassemblant les deux notions. Un état global est donc défini comme $(ws, ms1, ms2, ms3\ldots)$ ou abrégé (ws, ms), l'état du monde suivi de l'état de chaque motivation. Dans notre exemple, nous pourrons définir par exemple un état global comme $(ws, step1, good, closed)$.

Afin de passer d'un état global gs à gs', nous allons utiliser le modèle des motivations pour créer des action ainsi qu'une fonction de transitions entre nos états globaux. La condition permettant de passer d'un état ms à un état ms' est d'activer un déclencheur partant de ms et allant à ms'.

Figure 10 – *Composition d'un état global*

Nous allons commencer à créer nos transitions en nous intéressant aux déclencheurs détectant des transitions particulières et ignorant les ressources. L'objectif simple consistant à provoquer l'apparition d'une de ces transitions peut être envoyé au planificateur du bloc opérationnel afin d'obtenir un plan résolvant cet objectif. Jouer ce plan permettrait donc de déclencher un changement d'état dans la motivation visée. Notons que tous les déclencheurs ne sont pas accessible depuis un état motivationnel. Par exemple, le déclencheur *Deposit* n'est accessible qu'à partir de l'état *step*3 de la motivation Usinage.

Une fois tous les déclencheurs accessibles détectant des transitions finales pour un état gs déterminés, nous allons ne garder que ceux dont la récompense est positive pour créer nos actions. Prenons par exemple l'état $(ws, step1, good, closed)$, avec ws un état du monde quelconque. Les déclencheurs sur des transitions finales disponibles sont : Supply, Machine, Recharge et GoToRestrictedArea. Nous n'allons garder que les trois premiers, puisqu'eux seuls renvoient une récompense positive.

Afin de créer une action, nous allons envoyer au planificateur les transitions finales avec la récompense d'un déclencheur. Le planificateur utilisera cette entrée et constituera avec elle la fonction de récompense à optimiser. Une fois le plan calculé, il constituera une **tâche**, jouable comme une action pour notre modèle. Cependant, pour créer la politique correspondant à la tâche voulue, nous n'allons pas utiliser que le déclencheur choisi. Nous allons utiliser également tous les autres déclencheurs accessibles. La raison est qu'il est possible d'atteindre, pendant l'exécution du plan sensé atteindre un déclencheur, la transition finale d'un autre déclencheur. Souhaitant connaître le comportement des tâches, nous voulons surveiller ces autres transitions également. Pour cela, nous allons également envoyer en argument au planificateur les spécifications de tous les autres déclencheurs accessibles avec les modifications suivantes : les récompenses positives deviendrons nulle, afin de ne pas créer une politique pour une tâche qui tendrait à en exécuter une autre à la place. Les déclencheurs ayant une récompense négative resterons tels quels, poussant la politique à éviter les transitions non-voulues.

Pour continuer avec notre exemple, étudions le cas de la tâche Supply. Les autres déclencheurs accessibles sont Recharge, Machine et GoToRestrictedArea. Pour créer la tâche souhaitée, nous allons envoyer au planificateur la spécification des transitions finales et de la fonction de récompense à utiliser. Nous commençons par envoyer les transitions finales de la tâche Supply accompagnées de leurs récompenses. Ensuite, nous fournirons les transitions finales des tâches Recharge et Machine, mais avec une récompense nulle. Nous enverrons enfin les transitions finales de GoToRestrictedArea avec la récompense, négative, qui l'accompagne. En résumé :

- ([robot en H, obj 1 et 2 en main], supply, [robot en H]), récompense 5
- ([robot à un chargeur], recharge, [robot à un chargeur]), récompense 0
- ([robot en S, obj 1 et 2 en main], machine, [robot en S]), récompense 0
- ([robot au sud], move, [robot au nord]), récompense -50

L'étape suivante consiste à prévoir les effets d'une tâche sur un état global gs. Nous connaissons l'effet sur l'état du monde ws, grâce à l'estimation sur les effets de la tâche donné par le planificateur. Celui-ci calcule en plus de la politique et à partir d'un état ws, la probabilité de finir sur chaque transitions finales et le coût moyen en ressource lorsque le robot y arrive. Avec l'information de la probabilité de finir sur une transition finale donnée, nous pouvons savoir quel déclencheur aura été atteint, et avec quelle probabilité. Nous pouvons donc associer à chaque transition finale un état global atteint après l'exécution de la tâche et la probabilité d'y accéder à l'issue de l'exécution.

Nous avons pour l'instant pris en compte les déclencheurs sur des transitions finales, il reste à prendre en compte les déclencheurs surveillant les ressources et de déterminer si les coûts de la tâche en activera.

Nous allons donc vérifier pour toutes les transitions finales et pour leur coûts associés si un déclencheur sur les ressources atteignable risque de se déclencher. Une fois ce calcul fini, nous serons en mesure de connaître pour gs et une tâche ta, un ensemble d'états globaux ainsi que la probabilité pour chacun de s'y trouver après l'exécution de ta, c'est-à-dire une fonction de transition pour notre modèle délibératif.

Pour finir avec notre exemple, commençons par écrire (avec des probabilités données arbitrairement) la table de transition donnée sans prise en compte des ressources.

(ws,step1,good,closed), Supply	
0.94	(ws^1,step2,good,closed)
0.06	(ws^2,step1,good,closed)

Et enfin, avec les déclencheurs sur les ressources :

(ws,step1,good,closed), Supply	
0.79	(ws^1,step2,good,closed)
0.15	(ws^2,step2,low,closed)
0.06	(ws^3,step1,good,closed)

Construisons à présent la fonction de récompense de notre modèle. Celle-ci se calcule en même temps que la fonction de transition, en prenant la récompense des déclencheurs activés lors de la transition. Noter que contrairement à la requête faite au planificateur, si une transition finale appartient à un autre déclencheur que celui visé par la tâche, nous prendrons toujours la récompense du déclencheur, même positive.
Pour notre exemple, notons que le fait de passer du niveau de batterie Good à Low donne une récompense de -10. Voici donc un extrait de la table de récompense :

(ws,step1,good,night), Supply	
(ws^1,step2,good,night)	5
(ws^2,step2,low,night)	-5
(ws^3,step1,good,night)	-50

Nous possédons à présent ce qu'il nous faut pour construire notre modèle délibératif. Un ensemble d'états, les états globaux gs, des actions, les tâches, une fonction de transition, grâce aux estimations sur les politiques apportées par le planificateur et pour finir, une fonction de récompense avec la définition des motivations.

Nous pourrions théoriquement calculer la meilleure tâche à jouer depuis chaque état global, à la manière de Value Itération, mais ce calcul prendrait trop de temps, dû au grand espace d'état ainsi qu'au calcul complet de la nouvelle fonction de transition. Pour contourner ce problème, nous allons calculer, depuis l'état présent du système, les meilleurs choix de tâches jusqu'à un horizon de temps (la ressource de l'état du monde) donné en s'inspirant de Value Itération. Cela aura pour avantage d'être un calcul facile, faisable en ligne, et proche du calcul optimal si l'horizon de temps est suffisamment éloigné.

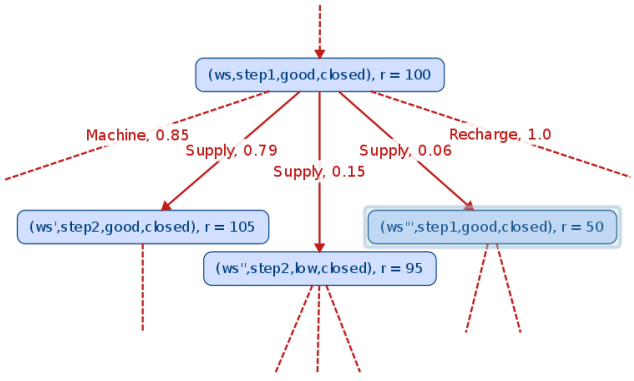

Figure 11 – *Fragment de l'arbre de l'exemple avant sélection des meilleures tâches*

Pour ce calcul, nous allons développer un arbre partant de l'état actuel avec une récompense de base de 0 et où chaque nœud a pour fils, pour chaque tâche jouable et chaque état d'arrivée possible, un fils décrivant l'état futur ainsi que la récompense totale accumulée jusqu'à cet état. La branche étant annotée de la tâche jouée et de la probabilité d'arriver dans le nœud suivant. Une fois développé jusqu'à un horizon de temps

donné, nous sélectionnerons pour chaque nœud en partant des feuilles la tâche promettant la récompense la plus forte en moyenne.

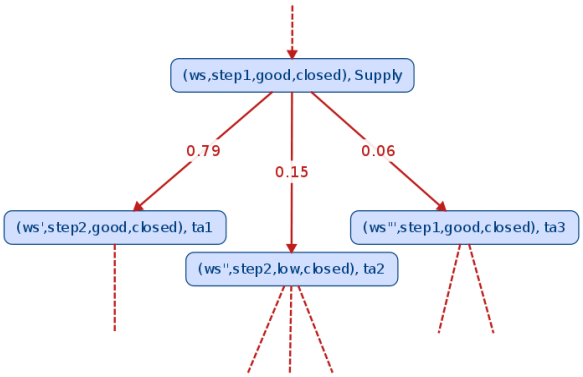

Figure 12 – *Fragment de l'arbre de l'exemple final*

Le but sera donc de calculer un arbre des décisions maximisant la récompense accumulée en moyenne pendant le parcours. Le résultat final sera un arbre dont la racine donne l'état global au moment où le calcul aura été lancé. Chaque nœud donnera un couple (état global, tâche) et chaque branche est étiqueté de la probabilité qu'à partir du père et en exécutant la tâche, l'état passe au nœud fils. La transition entre chaque nœud père et un nœud fils décrira donc une transition possible : $(gs, ta) \rightarrow$ probabilité $\rightarrow (gs', ta')$.

7 Conclusion

Notre but dans ce travail était de proposer et développer une architecture donnant à un robot la capacité de prendre des décisions par lui-même, et non pas par réaction à un stimulus ou une requête.

Pour y arriver, nous avons voulu séparer le modèle décisionnel en deux parties : une partie opérationnelle, exprimant ce que le robot peut faire, et une partie intentionnelle, ce que le robot veut faire. De cette manière, les objectifs du robot ainsi que les requêtes n'interviendront que dans la partie du bloc intentionnel, et les moyens pour satisfaire ces intentions viendront du bloc opérationnel. L'objectif est donc de savoir quelles actions du modèle opérationnel jouer. Pour finir, la phase de délibération prendra en compte tous les objectifs du robot et toutes les possibilités permettant de faire évoluer les objectifs afin de prendre une décision.

La partie opérationnelle, directement héritière des techniques habituelles de décision, consiste en un modèle type MDP, donnant l'évolution du monde en fonction des actions du robot. En plus de ce modèle, un planificateur prenant en entrée la spécification d'une tâche à réaliser et en sortie une politique ainsi qu'un ensemble d'estimations sur le comportement de cette politique.

La partie intentionnelle, elle, se compose de motivations, sous la forme d'automates finis particulier. Cette forme permet de modéliser les objectifs de façon complète, temporalité, sous-tâches, embranchements multiples. Les transitions des automates spécifient des conditions à atteindre dans le modèle opérationnel pour que la transition s'effectue ainsi qu'une notion de récompense donnant l'utilité de cette transition.

Le mécanisme délibératif sera pour finir chargé de coordonner les deux précédentes parties. Les états du monde et des motivations sont alors combinées pour donner un méta-état. Les tâches des motivations sont envoyées au planificateur afin de créer des méta-actions. La fonction de méta-transition est alors déterminée par les estimations sur la politique des tâches retournées et la fonction de récompense est directement héritée de celles des motivations. En générant les scénarios possibles à partir d'un état courant, nous sommes alors capables de donner un plan constitué de tâches à suivre, et donc d'actions à jouer, maximisant l'utilité des motivations.

Pour conclure, L'architecture proposé ici permet donc de formuler des objectifs variés et de les rassembler au sein d'un modèle intentionnel, coupé du modèle opérationnel. Les motivations possède une notion d'utilité, mais le choix de la tâche à jouer à un instant t n'en dépend pas directement. Le mécanisme délibératif étudie la faisabilité des tâches, les états dans lesquels elles peuvent mener, l'impact sur les autres motivations et la récompense promise pour les différents scénarios afin de délibérer. Le modèle fait donc preuve de méta-raisonnement sur ses propres objectifs et lui permet de prendre ses propres décisions.