Université Paris 8
Laboratoire d'Informatique Avancée de Saint-Denis

Habilitation à diriger des recherches

présentée par

Nicolas JOUANDEAU

Contributions de la robotique mobile, humanoïde, multi-robots aux jeux à information incomplète

Soutenue le 11 décembre 2014 devant le jury :

Arab ALI CHERIF	Directeur	UNIVERSITÉ PARIS8
Rachid ALAMI	Rapporteur	LAAS-CNRS
Younès BENNANI	Rapporteur	UNIVERSITÉ PARIS13
Jacqueline SIGNORINI	Rapporteur	UNIVERSITÉ PARIS8
Lakhdar SAIS	Examinateur	UNIVERSITÉ D'ARTOIS
Frédéric SAUBION	Examinateur	UNIVERSITÉ D'ANGERS
Herman AKDAG	Examinateur	UNIVERSITÉ PARIS8

Remerciements

Je tiens à remercier vivement tous les membres du jury :

Arab ALI CHERIF pour la direction de mon habilitation à diriger des recherches.

Rachid ALAMI, Younès BENNANI et Jacqueline SIGNORINI, de m'avoir fait l'honneur d'accepter d'être rapporteurs de mon HDR, de participer au jury et de s'être interessés à mes travaux.

Lakhdar SAIS, Frédéric SAUBION et Herman AKDAG, d'avoir accepté d'examiner mes travaux.

Les travaux présentés dans cette habilitation à diriger des recherches ne seraient pas formulés dans le présent rapport, sans la présence et le soutien, sans les discussions diverses et enrichissantes, sans l'ensemble des personnes ayant partagées à un moment cette période, et plus particulièrement : Zhi YAN, Tristan CAZENAVE, Vincent HUGEL, Patrick GREUSSAY, Pierre BLAZEVIC, Patrick BONNIN, Abdallah SAFFIDINE, Aldenis GARCIA-MARTINEZ, Loic THIMON, Thomas DA COSTA, Nawel ZEDADRA, Lawrence NDERU et Sylvie GOARRIN.

Résumé

Nous présentons dans cette habilitation à diriger des recherches, les résultats de 10 années de travaux et d'expérimentation sur les algorithmes probabilistes, sur l'exploration et le transport multi-robots, sur les mouvements en robotique humanoïde, sur l'informatique médicale et sur les jeux. Des problèmes mono-robot aux jeux à 1 joueur, des problèmes multi-robots aux jeux à N joueurs, les problématiques diffèrent mais les problèmes et les solutions sont très proches. En robotique mobile, nous avons utilisé la décomposition de l'espace des configurations en cellules régulières pour accélérer la planification géométrique par construction informée d'un arbre d'exploration. Nous avons présenté des solutions permettant de réduire le temps de construction de l'arbre proportionnellement au pas de discrétisation des commandes et à la position des nœuds dans l'espace. En robotique mobile collective, nous avons formalisé la notion de situation d'attente dans les problèmes de transport et d'exploration et nous avons proposé des solutions minimisant temps et énergie consommés en utilisant des approches heuristiques. Nous avons également proposé des solutions probabilistes par échantillonnage, par estimation des congestions hors-ligne et par adaptation en-ligne à l'encombrement de l'espace de travail. En robotique humanoïde, nous avons présenté des solutions de génération de mouvements, de vision temps-réel embarquée et de génération de stratégies collectives. Nous avons présenté des travaux précurseurs en optimisation morphologique humanoïde. Dans le domaine des jeux, nous avons proposé les premières solutions de parallélisation des algorithmes Monte Carlo pour le *Jeu de Go*. Nous avons combiné analyse retrograde, résolution parallèle et simplification par symétries dans le matériel pour des problèmes à 1 et 2 joueurs. Les nombreuses participations dans les compétitions internationales de robotique avec la plateforme humanoïde *NAO* et dans les compétitions internationales de jeux telles que les *Computer Olympiad*, nous ont permis de valider en pratique les idées présentées dans nos travaux de recherche. Pour chacune des contributions citées dans ce mémoire, nous tentons de dresser un bilan de l'effectivité de nos recherches et d'en présenter les perspectives possibles.

Table des matières

Chapitre 1

Introduction

Nous présentons dans cette habilitation à diriger des recherches, les résultats de 10 années de travaux et d'expérimentation sur les algorithmes probabilistes, sur l'exploration et le transport multi-robots, sur les mouvements en robotique humanoïde, sur l'informatique médicale et sur les jeux. Des problèmes mono-robot aux jeux à 1 joueur, des problèmes multi-robots aux jeux à N joueurs, les problématiques diffèrent mais les problèmes et les solutions sont très proches.

Nos travaux s'instaurent initialement dans l'étude des algorithmes Monte Carlo pour la planification de mouvement en robotique mobile. Face aux méthodes de résolution géométrique complète adéquates aux espaces sous-dimensionnés, les algorithmes probabilistes eurent le bénéfice de proposer des solutions sous optimales convergentes permettant de traiter des problèmes de planification de mouvement proches de la réalité de la robotique mobile. Ce changement modifia considérablement les problèmes, les modèles et les solutions en robotique mobile. Quelques années après, l'avènement des algorithmes Monte Carlo pour la résolution des jeux abstraits fût une expérience passionnante en nous permettant de revivre l'effervescence d'une telle révolution dans le domaine des algorithmes arborescents pour les jeux abstraits. Ce lien entre robotique et jeux marque notre intérêt pour tous les problèmes combinatoires et les possibilités de résolutions incomplètes, sous optimales, évolutionnaires et toujours convergentes vers une meilleure solution au fil des itérations.

Algorithmes probabilistes

En planification de mouvement, l'intérêt pour les algorithmes Monte Carlo apparaît à partir de 1995 avec les Méthodes de Réseaux Probabilistes (MRP) sous diverses formes dans les travaux de L.E. Kavraki, de J.C. Latombe et de P. Svestka [135, 137, 213]. La notion d'échantillonnage est utilisée pour pallier l'énumération complète des états de l'espace de recherche, permettant potentiel-

lement d'obtenir une meilleure solution à chaque itération. Les méthodes MRP combinent discrétisation de l'espace de recherche, possibilité d'obtenir une solution sous-optimale et nouvelle puissance de calcul des ordinateurs. Elles sont à cette période un précalcul hors-ligne qui permet de répondre à des requêtes de planification entre 2 positions dans un espace des configurations prédéfini. Avec les travaux de S.M. LaValle [153], elles deviennent partie intégrante d'un calcul en-ligne dans la construction d'un arbre des états atteignables sous l'intitulé *Rapidly-exploring Random Tree* (RRT). Quelques années après l'apparition des RRT, les travaux de S. Carpin et E. Pagello [34] présentent les premiers résultats sur la parallélisation des algorithmes probabilistes RRT à l'aide d'un ordinateur parallèle IBM RS/6000 SP composé de 16 processeurs. Nous proposerons des variantes spécialisées des RRT, permettant d'accélérer le temps d'obtention des résultats et de construire des graphes dans des espaces des configurations contenant des passages étroits [112, 114]. En 2005, E. Plaku et L.E. Kavraki [188] développeront de nouveaux algorithmes parallèles pour des *solver* de planification pour des problèmes à 126 dimensions sur des clusters regroupant 286 processeurs. Cette considération des algorithmes Monte Carlo dans les problèmes décisionnels est arrivée plus tard dans le domaine des jeux [54, 141, 43] mais a également modifié l'ordre établi dans une profusion de variantes [28]. Les algorithmes Monte Carlo ont remis en question la suprématie de la recherche minimax dotée de coupes alpha-beta. Les problèmes du *Jeu de Go* ont été les premières opportunités pour nous de participer à des compétitions internationales *Computer Olympiad*. En collaboration avec T. Cazenave, nous avons décrit les premières solutions Monte Carlo parallèles [40, 39] pour les jeux abstraits. Les premières participations en *Go 9x9* furent concluantes. Lors de la 12ème édition des *Computer Olympiad* en 2007, Golois finit 5ème [1] en *Go 9x9* et obtient la médaille d'or [2] en *Go-phantom*. Golois est alors le seul programme de la compétition en *Go 9x9* à s'exécuter sur un cluster de 16 ordinateurs [3]. Dans la compétition, les programmes adverses utilisent des processeurs à 2 et 4 cœurs [4]. Lors de la compétition suivante en 2008, un grand nombre de programmes s'exécutèrent sur des clusters et des

1. Golois fût 5ème ex-aequo avec Mango. Dans les 6 premiers programmes, nous trouvons 4 programmes écrits par des Français : Mogo de S. Gelly et Y. Wang, Crazy Stone de R. Coulom, Indigo de B. Bouzy et Golois de T. Cazenave et N. Jouandeau. Pour sa première apparition en compétition, Mogo remporte la première place dans la catégorie *Go 19x19*.

2. Golois n'a perdu sa médaille d'or qu'une seule fois depuis.

3. 16 pentium4 3Ghz sous Linux. La première version de Golois utilise PVM (Parallel Virtual Machine), une librairie de programmation parallèle pour les réseaux hétérogènes d'ordinateurs.

4. Les premiers processeurs multi-cœurs IBM apparaissent en 2001 mais pour les ordinateurs courants, Intel et AMD en produisent à partir de 2005. Ces ordinateurs ne dépassent pas 4 coeurs. Les ordinateurs SMP sont restés longtemps une référence. Depuis 2005, les fabriquants de processeurs développent les architectures multi-cœurs. Depuis 2008, Intel présente le i7, un processeur 8 cœurs.

ordinateurs multi-cœurs.

Exploration et transport multi-robots

Partant des problèmes mono-robot, nous avons étudié les problèmes multi-robots avec la thèse de Z. Yan [235]. Doctorant au LIASD [5], ses travaux ont commencé par une étude comparative des plateformes de simulation : CARMEN, *the Carnegie Mellon Robot Navigation Toolkit* aura été pour nous le début de manipulations fastidieuses avec de longs fichiers de configuration. Player/Stage aura été une alternative plus simple, plus modulaire, plus adéquate à nos besoins de validation par simulation de systèmes multi-robots. Ainsi les questions de coordination dans les systèmes multi-robots sont devenues importantes et nous avons étudié des algorithmes centralisés et des algorithmes distribués, avec des actions guidées par des fonctions heuristiques, des actions plus réactives guidées par une communication stigmergique et des actions réfléchies guidées par des échanges de messages dirigés. Nous avons étudié la robotique mobile collective, les problèmes de transport et d'exploration multi-robots [242, 128]. Du point de vue abstrait (par action) de la planification de tâches ou plus concret (proche des consignes de commande et des questions de géométrie dans les déplacements) de la planification de mouvement, l'utilisation de plusieurs robots nous permet de réduire le temps d'exécution des missions [241]. Nous avons également proposé des solutions permettant d'améliorer la coordination [127, 236], de réduire la consommation d'énergie et d'utiliser un protocole d'échanges de messages [239] décentralisé pour les problèmes d'exploration [238] et de transport [238]. Simultanément à ces recherches, nous avons étendu nos recherches au domaine de la robotique humanoïde, tout en maintenant un lien avec les questions des jeux abstraits.

Mouvements en robotique humanoïde

Intéressés par la robotique humanoïde, nous avons fait équipe avec 4 robots humanoïdes *NAO* pour participer aux compétitions internationales de *RoboCup Standard Plateform League*. Doté de 14 degrés de liberté (ddl), *NAO* possède une silhouette amicale et un modèle de construction similaire aux robots humanoïdes de plus grande taille. Il est largement utilisé pour expérimenter des solutions de déplacements, de stabilisation et d'interaction robot-humain. Ses nombreux capteurs regroupent une centrale inertielle (avec accéléromètre et gyromètre), 2 sonars ultrasons, des capteurs de pression sous les pieds, des bumpers pour détecter les collisions frontales au niveau des pieds, des micros, des haut-parleurs et 2 caméras indépendantes sans vue commune donc sans possibilité de stéréo-vision. Ainsi en

5. Laboratoire d'Informatique Avancée de Saint Denis, Université Paris8.

collaboration avec V. Hugel, nous avons étudié les déplacements, les perceptions et les stratégies collectives d'une équipe de *NAO* dans des compétitions de football avec des environnements simplifiés pour limiter les problèmes de perception. Avec les compétitions en public, des questions pratiques telles que les modifications de source d'éclairage et de terrain sont apparues et nous ont confronté à la qualification des conditions de bon fonctionnement de nos solutions dans des problèmes très concrets et plus ouverts [6] que le cadre classique des problèmes de recherche. Avec une architecture modulaire, nous avons développé des algorithmes temps-réel embarqués et présenté des résultats en vision [115] et en mouvement humanoïde [98]. Partant de travaux en vision temps-réel sur *Aibo*, nous avons proposé un algorithme robuste de segmentation pyramidale temps-réel en régions sans paramétrage. Nous avons établi des comportements individuels par automate hiérarchique à états finis et expérimenté des possibilités d'enrichissement des automates avec des états collectifs pour l'exécution de tâches coopératives. Avec pour contrainte le sous-actionnement des jambes du robot humanoïde *NAO* [7], nous avons proposé la première solution exacte de calcul de l'angle de cette articulation commune [94]. Avec les compétitions de *RoboCup-3D Soccer Simulation League*, nous avons développé les questions de paramétrage et de modélisation de la marche humanoïde [95]. L'utilisation de la simulation facilitant les expérimentations, nous avons étudié l'application de méthodes d'apprentissage et d'optimisation au robot humanoïde *NAO*. En conservant des profils de mouvements réalistes, proches des mouvements réalisables sur les robots humanoïdes de taille humaine, nous avons amélioré les performances de *NAO* en unifiant propriétés physiques et logiques dans un processus d'optimisation morphologique [122]. Nous avons validé les proportions physiques du robot réel et avons appliqué ce processus à des mouvements de coups de pied dans une balle [121]. Nous avons présenté un nouveau processus d'optimisation par croissance humanoïde pour l'établissement d'humanoïdes spécialisés dans certaines actions [119]. Les nombreuses participations [8] dans les compétitions de *RoboCup-SPL* et en *RoboCup-3DSSL* nous ont permis de proposer des solutions reproductibles, robustes et pratiques [9].

6. Dans la course au *DARPA Grand Challenge*, S. Thrun déclara après sa victoire que la détection des routes sans signalisation était un problème pratique difficile sur lequel il n'avait pas connaissance de travaux de recherche. L'étude de solutions dans des cas réels d'utilisation implique le développement de solutions plus robustes et suppose l'identification des conditions d'utilisation des traitements choisis. Il s'agit donc de concevoir des algorithmes capables d'évaluer les conditions de bonne exécution des traitements choisis.

7. Composé de 5 ddl à chaque jambe et d'une articulation commune au niveau des hanches, *NAO* possède la particularité d'avoir une locomotion à 11 ddl. Cette articulation commune implique une dépendance entre les mouvements des jambes.

8. Au fil des participations, de nombreux étudiants de Licence et de Master se sont impliqués. Les plus assidus furent A. Garcia-Martinez, L. Thimon et T. Da Costa.

9. Pour les auteurs de l'article *Recasting Robotics Challenges as Experiments* de *Robotics*

Informatique médicale

Inspirés par les travaux P. Greussay et J. Signorini sur la modélisation de pathologies et de dysfonctionnements organiques [205], nous avons décliné les algorithmes Monte Carlo dans le domaine de l'informatique médicale. En collaboration avec des chercheurs du laboratoire *Computer Vision and Virtual Reality Technology* de l'université *Central South University* de Changsha en Chine, nous avons étudié les solutions d'imagerie rétinienne et mis au point une nouvelle méthode probabiliste non supervisée pour la segmentation des images de fond de l'œil [129]. Cet algorithme fait partie intégrante d'un projet d'équipement de diagnostic médical automatique des patients. Ces premiers travaux en informatique médicale nous ont permis de découvrir un vaste domaine d'application allant de la compréhension du corps humain aux mécanismes d'interprétation de la médecine.

Parallélisation et problèmes de jeu

Parallèlement à nos travaux sur la robotique humanoïde et l'informatique médicale, nous avons étendu nos recherches sur la parallélisation Monte Carlo du *Jeu de Go* à d'autres problèmes de jeu. Avec T. Cazenave et A. Saffidine, nous avons étudié les bases d'ouverture, les algorithmes *Proof Number* (PN) et les bases de finales. A *Sokoban*, nous avons établi un algorithme de construction des positions bloquantes [42]. Nous avons calculé les positions initiales bloquantes selon un ensemble de règles simples et calculé des positions supplémentaires par Analyse Retrograde. Les positions bloquantes de 1 à 12 pierres ont été combinées à un algorithme glouton pour réduire le nombre d'itérations des résolutions. Nous avons obtenu des simplifications par palier de problèmes précédemment résolus et résolu des problèmes précédemment non résolus avec IDA*. Nous avons étudié les algorithmes PN qui réalisent un parcours en meilleur d'abord en exploitant les branches les plus prometteuses. Nous avons présenté l'algorithme ParallelPN^2 pour améliorer les résultats d'une recherche PN^2 en exploitant les possibilités des systèmes distribués [195]. Les expérimentations nous ont permis de résoudre *Breakthrough* 6x5, qui est toujours une partie gagnante pour le 2$^{\text{ème}}$ joueur. Pour réduire la taille des bases de finales, nous avons proposé une approche par identification des symétries de matériel [194]. Nous avons présenté la notion d'équivalence pour des ensembles de pièces et leur représentation en graphes d'équivalences. Les expérimentations sur les jeux *Chinese Dark Chess*, *Skat* et au *Jeu des*

& Automation Magazine de Juin 2011, les compétitions sont un excellent moyen d'avancement de l'état de l'art et d'évaluation des algorithmes dans leur contribution aux solutions de la vie courante.

Dominos montrent des facteurs de compression variant en fonction de la diversité matérielle des jeux. Plus récemment, nous avons présenté des variantes de *Chinese Dark Chess* permettant d'obtenir des problèmes plus stratégiques ou plus combinatoires [111]. En construisant les arbres MCTS en fonction d'ensemble de coups, nous avons comparé des variantes de regroupement de nœuds MCTS et établi les gains de *playout* moins informatifs et d'évaluations heuristiques [117]. Par application de ces résultats, nous avons participé à des compétitions internationales [10] de programmes de *Chinese Dark Chess*.

L'ensemble de ses travaux entremêle des solutions statistiques pour des problèmes combinatoires impliquant plusieurs entités. Ces entités sont des robots et des joueurs. Focalisés sur les capacités décisionnelles de ces entités, elles nous sont apparues comme très proches. Nous avons de fait considéré par périodes cycliques, la question de l'amélioration des performances décisionnelles de robots mobiles, de robots humanoïdes et de programmes joueurs. Nos travaux présentent un ensemble de résultats appliqués à la robotique et aux jeux abstraits, dans des problèmes impliquant 1 à plusieurs robots, dans des jeux à 1 et plusieurs joueurs. Avec une combinatoire grandissant avec le nombre d'entités impliquées, les solutions statistiques des algorithmes Monte Carlo, probabilistes et évolutionnaires nous ont permis de présenter de nouvelles solutions. Le 2ème chapitre est consacré à l'algorithmique probabiliste appliquée à la robotique mobile et aux jeux au travers de premières solutions séquentielles et parallèles. Le 3ème chapitre traite de la robotique humanoïde, de la génération de mouvement, de la vision, de l'optimisation morphologique et de la génération de stratégies collectives. Le 4ème chapitre traite de la robotique mobile multi-robots, des questions de coopération et de coordination dans des problèmes de transport et d'exploration. Le 5ème chapitre traite des extensions des algorithmes Monte Carlo à d'autres problèmes de jeu, de leur combinaison avec les algorithmes PN et avec l'Analyse Retrograde pour l'amélioration des solutions existantes ou la résolution complète de problèmes de jeu. Le 6ème chapitre décrit les perspectives de ces travaux, une extension en informatique médicale et les extensions actuellement en cours de développement dans 3 nouveaux projets en robotique aérienne, en robotique médicale et en parallélisation des algorithmes MCTS. Pour chacun de ces chapitres, nous tentons de dresser un bilan des contributions réalisées et de présenter les perspectives possibles.

10. Dans les compétitions *Computer Olympiad* et *Taiwan Computer Game Association*, Homer de N. Jouandeau et T. Cazenave est le seul participant européen de ces compétitions en 2013 et 2014. Lors de la compétition en 2013, Homer finit 7ème sur 10.

Chapitre 2

Algorithmique probabiliste

Sommaire

Nous décrivons dans ce chapitre les résultats de nos premiers travaux de recherche concernant la planification de mouvement probabiliste [1] et la parallélisa-

1. La planification de mouvement consiste en la prise de décision permettant l'évolution dans un espace et s'adresse sans réelle distinction aux robots mobiles et manipulateurs. Le terme robotique mobile englobe toutes les plateformes mobiles, a contrario des robots manipulateurs (dont la base est fixe), et caractérise aujourd'hui principalement les plateformes à roues. Les recherches en robotique mobile commencèrent au Stanford Research Institute (aujourd'hui SRI International) avec Shakey [177], le premier robot mobile, muni d'une caméra, de capteurs et de roues. Shakey est contrôlé à distance par un calculateur $SDS-940$ en 1966 et $PDP-10/PDP-15$ à partir de 1969. Le calculateur interprète les ordres écrits en langage naturel, les décompose en actions de plus bas niveau à l'aide du solveur STRIPS (STanford Research Institute Problem Solver) dans une carte de l'environnement et transmet à Shakey par liaison radio la séquence d'actions correspondante. Shakey contrôle l'absence d'obstacle sur son chemin, peut pousser les objets et donc si besoin réorganise l'occupation des salles au travers de mission du type "bloque la porte D1". Ces objets, avec lesquels Shakey interagit, sont similaires à du mobilier, de forme simple (carré ou rectangle) et de couleur clairement identifiable (rouge uni). Une caméra lui permet de se repositionner dans la carte. Des capteurs de pression lui permettent de détecter les obstacles sur son chemin, modifiant ainsi éventuellement la carte dans laquelle il évolue. Dans la catégorie des robots mobiles, les principales spécialisations dont l'intérêt est grandissant par le potentiel des interactions humaines possibles, sont les robots humanoïdes, les robots manipulateurs mobiles et les robots aériens. La complexité d'un problème de planification d'un robot est fonction de l'espace

tion [2] des algorithmes probabilistes.

théorique dans lequel il se déplace, lui-même fonction du nombre de degrés de liberté du robot et du nombre de contraintes sur ces degrés de liberté. Un degré de liberté est un paramètre géométrique, linéaire ou angulaire dans le système modélisant le robot. Pour un robot manipulateur, le nombre de degrés de liberté est défini par le nombre de ses articulations munies d'un actionneur. Pour un robot mobile de type voiture se déplaçant dans le plan à vitesse constante, le nombre de degrés de liberté est égal à 3 : un degré en x, un degré en y et un degré en θ (de position angulaire). De la dimension du modèle du robot naît la complexité du problème. La notion de complexité en planification a été formalisée dans le problème du déménageur de piano. J.T. Schwartz *et al.* [203] montrent la complexité déterministe-polynômiale de ce problème. H.J. Reif [190] a montré que les problèmes de planification les plus intéressants appartenaient à la classe PSPACE-hard dont la position dans le classement des complexités est : NL-hard < P-hard < NP-hard < PSPACE-hard < EXPTIME-hard < EXPSPACE-hard. J.F. Canny [32] a démontré l'appartenance de ce problème à la classe PSPACE-hard pour un nombre de degrés de liberté non borné. Il décrit également un algorithme de complexité $O(mn\mathtt{log}(m))$ en temps. La complexité en temps d'un problème de planification de mouvement en présence d'obstacles est donc exponentielle en la dimension de l'espace des configurations. Les premières méthodes de planification déterministes [180] permettent difficilement de faire face à la complexité de plateformes ayant plus de 3 degrés de liberté. Par opposition, les méthodes de planification non-déterministes le permettent et sont dites à complétude probabiliste. La résolution d'un problème de planification de mouvement consiste à explorer l'espace de recherche afin de trouver une solution permettant au robot de se déplacer d'une position initiale vers une position finale. La valeur ajoutée des méthodes probabilistes peut se résumer à l'utilisation de fonctions aléatoires permettant de réduire la complexité de la résolution des problèmes de planification.

2. La parallélisation concerne ici le traitement des informations simultanément au travers de plusieurs séquences d'instructions, que l'on retrouve aujourd'hui dans les ordinateurs courants multi-cœurs et les réseaux d'ordinateurs. Concernant le traitement associé aux algorithmes probabilistes, il s'agit d'un parallélisme de haut niveau (ou logiciel) réalisé via mémoire partagée principalement sur ordinateurs multi-cœurs ou mémoire distribuée principalement sur réseaux d'ordinateurs (clusters), avec pour objectif courant un gain en temps égal au nombre de séquences d'instructions exécutées simultanément. Dans un contexte d'exécution sans faute, le gain maximum est borné par une accélération linéaire, de par le temps requis pour l'exécution des portions de programme qui ne peuvent pas être parallélisées, conformément à la loi d'Amdahl. S. Carpin et E. Pagello [34] ont mené des travaux précurseurs sur la parallélisation des algorithmes probabilistes RRT sur un supercalculateur parallèle IBM RS/6000 SP et ont obtenu un facteur d'accélération de 3.5 avec 4 processeurs (1 nœud à 4 CPU) et un facteur d'accélération de 4 avec 16 processeurs (4 nœuds à 4 CPU). Notons qu'en 1998, la nouvelle version du supercalculateur IBM RS/6000 SP [17], utilisée par S. Carpin et E. Pagello, est considérée comme 5 fois plus puissante que Deep Blue (basé sur l'architecture du RS/6000 SP, un an plus tôt) qui en 1997 a battu le champion du monde G. Kasparov par 3.5 points contre 2.5 au *Jeu des Échecs*. E. Plaku et L.E. Kavraki [188] ont développé des algorithmes parallèles pour des problèmes de planification à 126 degrés de liberté et ont obtenu un facteur d'accélération de 72 avec 80 clients (sur le cluster nommé Rice's HP Intanium 2 Cluster, regroupant 286 processeurs). Plus récemment, D. Devaurs et al. [62] ont développé des algorithmes probabilistes distribués pour la résolution des problèmes de planification à l'aide des arbres d'explorations RRT et ont obtenu un facteur d'accélération de 25 avec un cluster regroupant 192 cœurs (24 serveurs bi-quadcore cadencé à 2.66 Ghz reliés par un réseau 10 Gbits/s). Pour résoudre des problèmes de planification probabiliste en incluant un détecteur de collisions parallèle, J. Bialkowski et al. [22] ont obtenu un facteur d'accé-

2.1 Planification probabiliste

Les méthodes de planification probabiliste permettent de résoudre de nombreux problèmes de planification. La méthode Randomized Path Planning (RPP) proposée par J. Barraquand et J.C Latombe [14, 16], a été développée à partir de la méthode de descente de gradient proposée par O. Khatib [139], qui est un parcours du champ de potentiel « en meilleur d'abord ». La méthode RPP présente une solution pour les mouvements d'un bras articulé réduit à un élément. Ces mouvements sont réalisés sans contrainte de dynamique et de temps dans l'espace des configurations C. Un déplacement Brownien [3] est ajouté pour sortir des minima locaux résultant de la descente de gradient. Les Méthodes de Réseaux Probabilistes (MRP) ont été proposées simultanément par L.E. Kavraki et J.C. Latombe [135, 137] sous l'intitulé Probabilistic RoadMap (PRM) et par P. Svestka [213] sous l'intitulé Probabilistic Path Planner (PPP). Le principe de résolution de ces méthodes divise la planification de mouvement en deux étapes successives de construction d'un graphe (ou phase d'apprentissage) et de construction d'un chemin basé sur ce graphe (ou phase de recherche d'une solution). La phase d'apprentissage construit un graphe de nœuds appartenant à C_{free} [4] dont les liens sont établis par un planificateur dit en ligne-droite (permettant de vérifier la non-collision avec les obstacles). Il est important de souligner que les méthodes PRM exploitent trois aspects du problème de planification, pour lesquels la construction du graphe peut se révéler identique :

- Répondre à tous les problèmes de planification,
- Répondre aux requêtes de connexion à partir d'une configuration initiale,
- Répondre aux requêtes de planification entre deux configurations.

Pour cette raison, les méthodes PRM sont dites à question-multiples, dans le sens où la construction de G peut permettre de résoudre différents problèmes de planification pour un même mobile dans un même espace. La méthode des arbres aléatoires d'exploration rapide (RRT) a été proposée par S.M. LaValle [153] sous l'intitulé Rapidly-exploring Random Tree. Son principe repose sur la construction

lération de 16 pour réaliser une détection de collisions parallèle sur carte GPU à 72 cœurs CUDA (Compute Unified Device Architecture). Si la parallélisation avec un gain linéaire est accessible au travers de moins d'une dizaine de séquences d'instructions parallèles, il reste difficile d'obtenir des gains similaires avec des dizaines et des centaines d'unités de calcul. L'évolution des ordinateurs parallèles s'oriente aujourd'hui vers les architectures massivement parallèles MPPA (Massively Parallel Processor Array) dont les projections prévoient d'accueillir de plusieurs milliers à des millions de processeurs et donc des facteurs d'accélération plus importants.

3. Le mouvement Brownien, également appelé *Processus de Wiener*, est un mouvement aléatoire dont l'amplitude suit une loi Gaussienne.

4. L'espace des configurations C est divisé en 2 parties qui sont C_{free} pour les configurations libres et C_{obs} pour les configurations en collision avec les obstacles.

d'un arbre G dans l'espace de recherche C [5]. A partir de la configuration initiale q_{init}, l'arbre est construit par application de commandes visant à rapprocher le mobile d'une configuration aléatoire. Le principe de cet algorithme réalise en séquence :

- La génération d'une configuration aléatoire q_{rand}.
- La sélection d'un élément de G proche de q_{rand}.
- L'expansion de l'arbre à partir de la configuration la plus proche de q_{rand} en direction de q_{obj}.

La variation de l'alternance entre ces 3 phases de génération, sélection, expansion produit des résultats différents. Il est naturel d'essayer d'adapter la construction de l'arbre à la progression de l'arbre dans l'espace. Une répartition uniforme des tirages aléatoires dans C tend à produire un graphe couvrant et garantie la convergence vers une solution [6]. La construction simultanée de deux arbres (partant de q_{init} et de q_{obj}) permet d'accélérer la résolution. Les deux arbres se développent tant qu'aucune connexion n'est établie entre-eux. A. Amato et B. Yamron ont proposé comme problème de référence, un problème connu difficile, nommé "Alpha 1.0 puzzle" composé de 2 tubes tordus emmêlés à séparer [7]. La résolution de ce problème utilise une variante bidirectionnelle [223]. Les solutions sont ici dans une très petite portion de C_{free}, ce qui donne lieu à de nombreuses spécialisations afin de rétablir les bonnes performances de cette méthode. S.M. LaValle et J.J. Kuffner [154] ont proposé une variante dédiée à la planification de mouvement de mobiles kinodynamiques diversifiés (la dimension de X variant de 4 à 12) dans un environnement statique encombré de nombreux obstacles créant ainsi des passages étroits. P. Cheng *et al.* [45, 46, 48] ont proposé la variante "Constraint Violation Probability" (CVP) favorisant l'indépendance à la distance métrique. Ils

5. La nature de l'espace de recherche est en réalité définie par le modèle du robot. T. Lozano-Pérez [157] utilise l'espace des configurations C, B.R. Donald *et al.* [63] utilisent l'espace des états X incluant des dérivées temporelles, T. Fraichard [72] utilise l'espace des états-temps ST.

6. Les méthodes probabilistes PRM et RRT répondent cependant à la question de l'existence d'une solution et de son unicité en fonction d'un pas de discrétisation choisi. Elles conservent l'atout indiscutable de résoudre des problèmes proches de la réalité. Un effet négatif, commun à toutes ces méthodes, lié à l'utilisation d'un échantillonnage aléatoire de l'espace, est de produire des solutions sous optimales, incluant des détours et des courbures inutiles pour atteindre la configuration objectif q_{obj}. Pour pallier cet effet, il est possible de procéder à un lissage de la première solution, permettant en quelques itérations de produire de meilleures solutions plus proches des possibilités d'asservissement des robots.

7. Ce problème a été proposé comme benchmark des problèmes de planification dans les espaces étroits. Parmi les 2 tubes emmêlés, l'un des tubes est fixe, considéré comme obstacle et l'autre est le mobile à déplacer. Chaque tube est composé de 1008 triangles. L'objectif est de séparer les tubes. Des variations de difficulté sont proposées en modifiant le diamètre du tube obstacle. La nature tubulaire symétrique permet d'obtenir deux directions de sortie possible. Dans la réalité, ce casse-tête est réalisé avec des clous, pour lesquels le problème sera plus difficile de par l'unicité de la solution.

ont également proposé une variante [47] ayant pour objectif de garantir la complétude [8] de la résolution. S. Carpin et E. Pagello [34] ont proposé une variation parallèle réduisant le temps de résolution et la longueur du chemin résultat. Ils proposent deux modes de parallélisation qui déterminent la contribution des différents processeurs. Le mode "OR parallel" pour lequel plusieurs processeurs sont associés à la résolution d'un même problème. Chaque processeur contribue à une solution différente. Le premier processeur ayant abouti à une solution, interrompt les autres processeurs. Le mode "embarrassingly parallel" pour lequel les processeurs contribuent à la même solution. Les dernières variantes permettent de résoudre des problèmes incluant des contraintes différentielles, à l'aide d'enchaînements de résolutions tels que :

- Le calcul d'un premier chemin $\mathscr{L}_0$ dans C_{free} en utilisant uniquement les degrés de positionnement du mobile.
- La production d'un chemin $\mathscr{L}_1$ par lissage de $\mathscr{L}_0$.
- Le calcul d'un chemin $\mathscr{L}_2$ guidé par $\mathscr{L}_1$ et satisfaisant les contraintes différentielles du mobile.
- La production d'un plan d'actions permettant de rester à proximité de $\mathscr{L}_2$ afin d'assurer un meilleur suivi de la solution à exécuter.

E. Frazzoli *et al.* [75] ont montré que la qualité des solutions produites reste cependant fortement dépendante des primitives de mouvement choisies. Th. Fraichard et H. Asama [73] ont proposé le calcul de la région des inévitables collisions X_{ric} pour réduire cette dépendance et conserver dans X les performances obtenues dans C. Le calcul de cette région est d'autant plus intéressant que la dérive de positionnement est importante (par conséquence d'une vitesse de déplacement importante ou par conséquence des imprécisions). Le choix de la distance métrique a également des conséquences importantes sur la phase de sélection des configurations q_{prox} dans C comme dans X. En robotique mobile, J.P. Laumond *et al.* [151] ont montré le lien entre dextérité et proximité de la configuration objectif dans l'exécution des mouvements d'un mobile non-holonome. En dehors de ce cadre, seule la fonction de distance qui spécifie une évaluation de la distance entre deux configurations. Cette valeur de distance permet de sélectionner q_{prox} dans l'arbre à partir de la configuration q_{rand}. Le rôle de la distance métrique est donc crucial pour l'expansion de l'arbre dans l'espace de recherche. Elle permet de mesurer la proximité entre une configuration et l'objectif, permettant ainsi une évaluation de la progression dans l'espace de recherche. Cette fonction est garantie sous l'hypo-

8. La complétude d'une méthode de planification probabiliste implique deux autres complétudes : celle de la discrétisation pour les questions de temps et de commandes et celle de la résolution pour les questions de pas de commande dans le temps. Ainsi, la première approximation caractérise l'espace du problème considéré et la seconde fixe les intervalles de temps considérés. J. Barraquand et J.C. Latombe [15] ont montré dans le cas d'un robot mobile, que le choix de la première approximation peut se résoudre par une diminution de son pas au fil du temps.

thèse d'une relation entre proximité et atteignabilité : les configurations les plus proches doivent être également les plus faciles à atteindre. Dans un espace comportant plus de deux dimensions, la définition d'une distance métrique n'est pas triviale. S'il existe une trajectoire entre deux configurations, une distance métrique est considérée comme bonne quand la distance mesure cette trajectoire ; s'il existe plusieurs trajectoires possibles, la distance associée est la plus petite de ces trajectoires. Le calcul de cette distance est fonction de la nature des paramètres du modèle du mobile : pour une configuration q identifiée par i paramètres linéaires et j paramètres angulaires, c_k représente le $k^{\text{ème}}$ paramètre linéaire et α_k représente le $k^{\text{ème}}$ paramètre angulaire (respect. q' identifiée par c'_k et α'_k). Les distances métriques les plus utilisées sont :

– La distance euclidienne [9, 90, 136] :

$$d(q,q') = \left(\sum_{k=0}^{i} (c_k - c'_k)^2 + n_f^2 \sum_{k=0}^{j} (\alpha_k - \alpha'_k)^2 \right)^{1/2}$$

avec n_f le facteur de normalisation égal au maximum des variations entre les valeurs de chacun des c_k

– La distance euclidienne pondérée [145, 155] :

$$d(q,q') = \left(s \sum_{k=0}^{i} (c_k - c'_k)^2 + n_f^2 (1-s) \sum_{k=0}^{j} (\alpha_k - \alpha'_k)^2 \right)^{1/2}$$

avec s une valeur fixée de l'intervalle $[0;1]$

– La distance de Manhattan [2] :

$$d(q,q') = \sum_{k=0}^{i} \left| c_k - c'_k \right| + n_f^2 \sum_{k=0}^{j} \left| \alpha_k - \alpha'_k \right|$$

– La longueur de la trace associée au volume balayé [233], qui est calculée par approximation d'un maillage de triangles obtenus par discrétisation des paramètres c_k et α_k.

Une simplification possible du calcul de la distance entre deux configurations de C est de réaliser ce calcul dans W. N.M. Amato *et al.* [8] ont montré qu'avec les méthodes PRM, la métrique euclidienne pondérée donne les meilleurs résultats dans la plupart des cas. Plus l'espace est encombré, plus les valeurs de pondération sur les positions doivent être importantes.

Mes premiers travaux de recherche réalisés pendant ma thèse [112] détaillent dans un premier temps une étude [113] des interactions entre les différents composants utiles à la planification de mouvement dans un but d'optimisation de

leurs interactions. Les méthodes de décomposition cellulaire de l'espace sont garantes d'une propriété de complétude sous condition d'un pas de discrétisation des contraintes géométriques. Ces méthodes s'adressent à des mobiles génériques mais dans des espaces sans grande dimension. Basée sur une analyse géométrique récursive, nous avons mis en relation décomposition de l'espace en succession de cellules régulières pour réaliser une construction informée de l'arbre d'exploration.

Pour réduire l'influence de l'espace à explorer, nous avons proposé [106, 107, 114] un algorithme permettant de diminuer le temps de construction de l'arbre proportionnellement :

- Au pas de discrétisation de l'ensemble des commandes applicables au mobile.
- A la position des nœuds de l'arbre dans l'espace à explorer. L'arbre regroupe des nœuds étendus (i.e. avec 1 ou plusieurs successeurs) et des nœuds à étendre (i.e. sans successeur). Parmi l'ensemble des nœuds de l'arbre, la probabilité d'être sélectionné (par la distance métrique) est plus forte pour les nœuds sans successeur. Si un nœud ne possède que des successeurs valides, la méthode d'extension la plus rapide choisit un de ces nœuds et le valide par un appel au détecteur de collisions. En utilisant notre algorithme, plus un nœud possède de successeurs et plus la probabilité d'accélération de la phase d'expansion de ce nœud augmente.

Notre algorithme diminue le temps nécessaire à l'exécution d'un nombre fixé d'expansions lors de la construction de l'arbre. Il permet de réduire la dépendance à la dimension des commandes applicables au mobile qui implique une augmentation des intégrations numériques nécessaire à l'expansion de chaque nœud. Le maintien en mémoire des résultats des intégrations diminue également le nombre des appels au détecteur de collisions. Nous avons également proposé une première décomposition de C en régions [108] augmentant le nombre d'expansions réalisées avec succès. Ces travaux augmentent l'expansion de l'arbre dans C_{free}. La déformation Gaussienne positionne le graphe G à proximité de l'axe médian de l'espace libre et augmente la progression de G dans les espaces étroits. Le calcul de ces régions permet de définir un échantillonnage progressif, contrôlé par la progression de G dans C. A chaque itération, une nouvelle configuration est sélectionnée dans le domaine de visibilité de G pour maximiser la probabilité d'expansion. La progression de l'arbre dans C est assurée par restriction des tirages aléatoires à C_{free}. L'absence d'une partition de C_{free} en tranches implique une complétude liée au pas de discrétisation des commandes (et au pas d'intégration numérique de ces commandes). L'existence d'une trajectoire sans collision dans C est caractérisée par l'existence d'une trajectoire associée à l'ensemble des commandes du mobile.

2.2 Premières contributions

Les méthodes probabilistes PRM et RRT appartiennent à deux classes plus larges d'algorithmes :

- Les algorithmes Las Vegas pour lesquels le temps de calcul n'est a priori pas fini (mais l'espérance du temps de calcul l'est).
- Les algorithmes Monte Carlo pour lesquels le temps de calcul est fixé et le résultat est défini par statistique de tirages aléatoires.

La complexité d'un problème de planification étant exponentielle en la taille du problème à résoudre, ces classes d'algorithmes sont particulièrement adéquates. En conséquence, mes travaux de recherche se sont orientés dans deux directions :

- L'adaptation de la phase d'expansion de la méthode RRT et l'évaluation des espaces libres.
- La parallélisation des algorithmes Monte Carlo.

Ces travaux sur les méthodes probabilistes et sur les algorithmes Monte Carlo sont décrits dans les paragraphes suivants.

2.2.1 Expansion et évaluation des espaces

Construction de l'arbre de la méthode RRT

La formulation originale de la méthode RRT est exempte de contrainte. Pendant la construction de G, l'ajout de contraintes implique principalement la prise en compte d'un planificateur local approprié et d'une fonction d'intégration permettant de calculer l'expansion d'une configuration. Présentée dans les algorithmes 1 et 2, cette formulation de la méthode RRT s'articule selon deux fonctions qui sont `consRrt` pour la construction de l'arbre et `nearestConfig` pour la sélection d'un plus proche voisin.

Function `consRrt` (q_{init} , k , Δt , C)

`init` (q_{init} , G);
for $i \leftarrow$ `1` **to** k **do**
  $q_{rand} \leftarrow$ `randConfig` (C);
  $q_{prox} \leftarrow$ `nearestConfig` (q_{rand} , G);
  $q_{new} \leftarrow$ `newConfig` (q_{prox} , q_{rand} , Δt);
  `addConfig` (q_{new} , G);
  `addEdge` (q_{prox} , q_{new} , G);
end
return G;

Algorithme 1 : Construction de G.

Function `nearestConfig` (q_{to_expand} , G)

$d \leftarrow \infty$;
foreach $q \in G$ **do**
  if `distance` (q , q_{to_expand}) $\leq d$ **then**
    $q_{prox} \leftarrow q$;
    $d \leftarrow$ `distance` (q , q_{to_expand});
  end
end
return q_{prox};

Algorithme 2 : Sélection du plus proche voisin dans G.

Le temps d'exécution de la fonction de construction de l'arbre est borné par k itérations. Pour atteindre une configuration finale q_{obj}, il faut ajouter une condition d'arrêt correspondant à une distance minimum entre q_{obj} et une nouvelle configuration q_{new} ajoutée dans G. Chaque nouvelle configuration q_{new} définie par extension de q_{prox} est fixée par la fonction d'expansion `newConfig`. Si l'algorithme n'arrive pas à une solution après k itérations, il est possible de poursuivre la construction de G avec la même fonction d'expansion ou de reprendre la construction de G depuis le début avec une nouvelle fonction d'expansion. Il est intéressant de souligner que l'algorithme utilise principalement les 3 fonctions suivantes :

- `randConfig` qui assure une répartition uniforme des tirages aléatoires dans C. S.R. Lindemann *et al.* [156] ont proposé une méthode itérative de construction d'un échantillonnage aléatoire sur $SO(3)$, appliquée au déplacement d'un mobile simple rectangulaire dans des formes $3D$ complexes. A. Yershova *et al.* [245] ont proposé une variante permettant d'optimiser chaque nouvel échantillon en fonction de critères tels que la dispersion et la divergence de l'échantillonnage.
- `nearestConfig` qui sélectionne la configuration la plus proche de q_{rand}, en accord avec une distance métrique. La recherche de plus proche voisin étant une composante clé des méthodes probabilistes, A. Yershova *et al.* [244] ont proposé un algorithme basé sur les KD-tree (K-Dimensional tree) permettant d'optimiser cette fonction dans les espaces de recherche les plus courants.
- `newConfig` qui intègre les contraintes du mobile et calcule un nouvel élément en accord avec un planificateur local. Planificateur local et recherche de plus proche voisin sont deux composants déterminants dans les résultats des algorithmes du type RRT. Les planificateurs locaux les plus utilisés sont la ligne droite, la rotation en s et les courbes de Reeds et Shepp [112].

Expansion de l'arbre de recherche

L'ajout d'un nouvel élément est lié à la poursuite de la construction ou à la construction d'un nouvel arbre utilisant un pas d'intégration plus petit. Pour définir dynamiquement la poursuite de l'algorithme, nous avons présenté une analyse de l'expansion de l'arbre dans l'espace de recherche. La figure 2.1 montre l'expansion d'un arbre de recherche dans un espace sans obstacle au fil des tirages aléatoires.

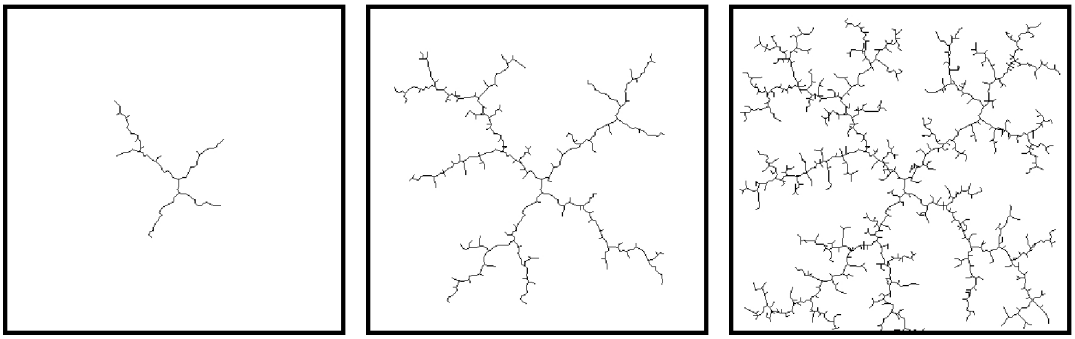


FIGURE 2.1 – Expansion de l'arbre de recherche après 100 tirages aléatoires, 500 tirages aléatoires et 1500 tirages aléatoires.

La figure 2.2 montre l'expansion simultanée d'un arbre de recherche et la triangulation de Delaunay associée. La figure 2.3 montre les variations de moyenne et d'écart-type des aires des triangles résultants. Dans [126], nous avons présenté une adaptation dynamique de la construction de l'arbre de recherche en fonction de ces variations.

Moyenne et écart-type des aires des triangles traduisent la progression de l'arbre dans l'espace. Les inflexions importantes constatées sur la figure 2.3 de la moyenne ou de l'écart-type, sont des indicateurs de décision pour la poursuite de la construction de l'arbre ou la mise en place d'un nouveau pas de construction. Ces travaux ont été réalisés avec la librairie MSL [9] et le détecteur de collisions PQP [10] pour planifier les mouvements de tube et pour les mouvements d'un bras articulé à 6 degrés de liberté dans des espaces en $3D$ comportant quelques obstacles.

Expansion dans les espaces encombrés

Dans les espaces encombrés, à l'instar du problème "Alpha 1.0 puzzle", l'affluence des obstacles produit un espace libre C_{free} très réduit dans lequel les appels

9. Motion Strategy Library, `http://msl.cs.uiuc.edu/msl`.
10. Proximity Query Package, `http://gamma.cs.unc.edu/SSV`.

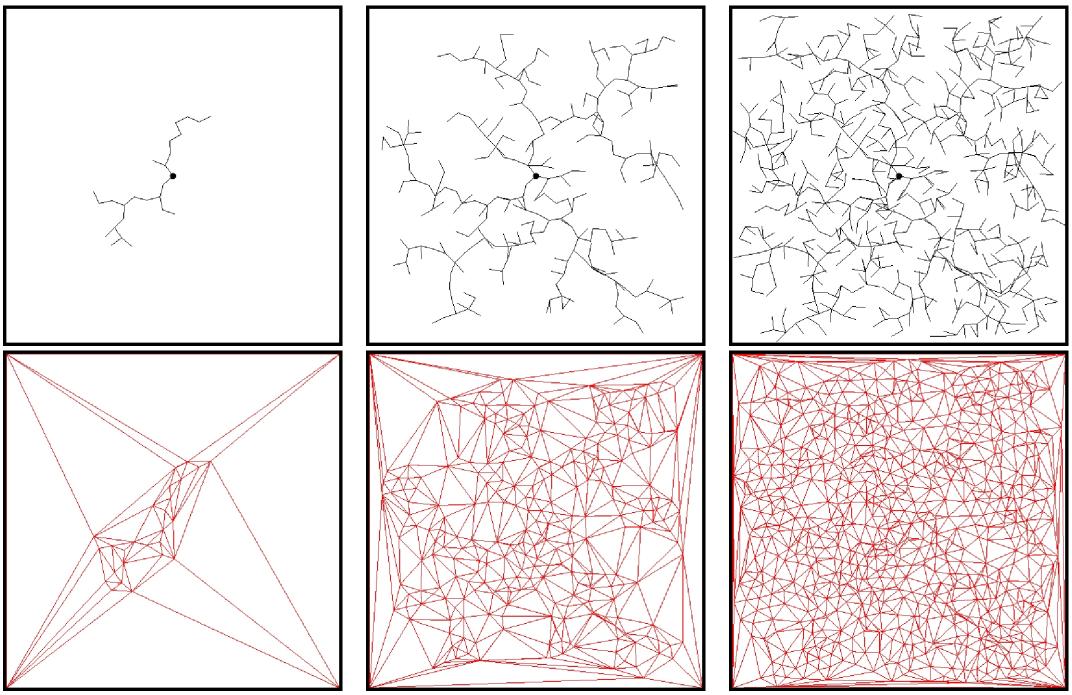


FIGURE 2.2 – Expansion de l'arbre de recherche et triangulation de Delaunay après 25 tirages aléatoires, 275 tirages aléatoires et 775 tirages aléatoires.

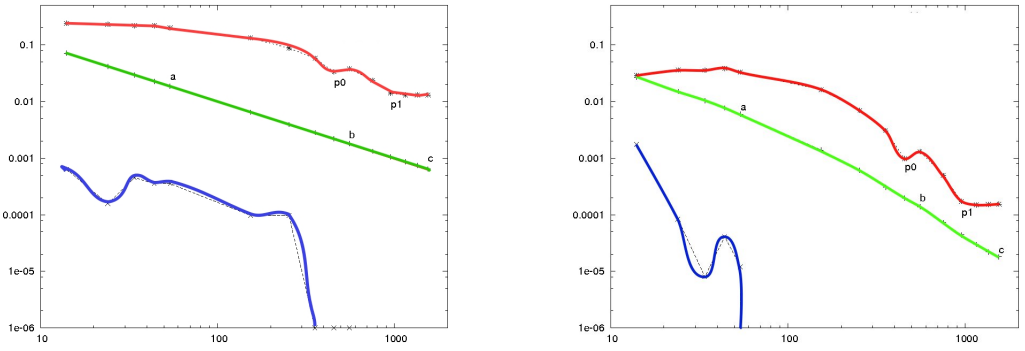


FIGURE 2.3 – Moyenne (à gauche) et écart-type (à droite) des aires des triangles au fil des tirages aléatoires.

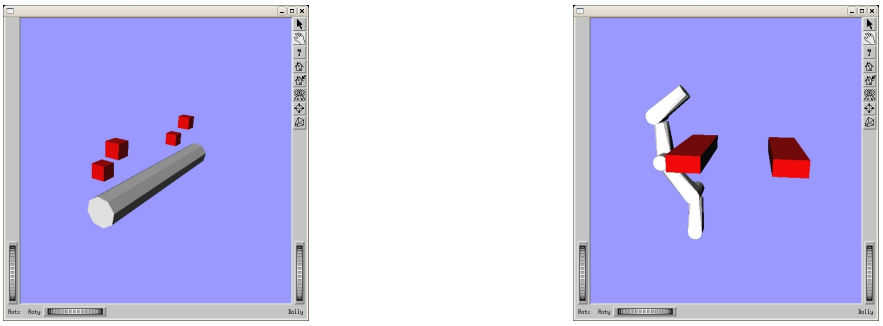


FIGURE 2.4 – Tube et bras articulé à 6 degrés de liberté dans des espaces en $3D$.

au détecteur de collisions diminuent la vitesse de convergence vers la solution. Pour pallier ce problème, nous avons proposé [110, 109] une nouvelle expansion détaillée dans l'algorithme 3.

```
Function clutteredExpand ( q , q_obj , Δt , G )

q_prox ← nearestConfig (q , G );
S ← ∅;
foreach u ∈ U do
    q_new ← integrate ( q_prox , u , Δt );
    d ← distance ( q_new , q_obj );
    S ← S + {q_new, d};
end
sort ( S );
n ← 0;
while n < card ( S ) do
    s ← getTuple ( S , n );
    q_new ← firstElement ( s );
    if isCollisionFree ( q_prox , q_new ) then
        addConfig ( q_new , G );
        addEdge ( q_prox , q_new , G );
        if q_new = q_obj then
            return REACHED;
        end
        return ADVANCED;
    end
end
return TRAPPED;
```

Algorithme 3 : Expansion minimisant les appels au détecteur de collisions.

A chaque nouvelle expansion en espace encombré, `clutteredExpand` retourne :

– *REACHED* si l'objectif est atteint.
– *ADVANCED* si un élément de *G* a été étendu avec succès.
– *TRAPPED* si l'expansion se solde par un échec.

U définit ici l'ensemble des commandes applicables au mobile. *S* regroupe des couples de configurations et de distances à l'objectif. `getTuple` permet d'obtenir le n^{ème} élément de *S*. Après un tri de *S*, l'expansion de *G* s'arrête à la première configuration sans collision.

Cette expansion dédiée aux espaces 3*D* encombrés a été testée sur un ensemble de problèmes importants. Nous avons étudié le comportement de notre algorithme en augmentant linéairement et exponentiellement l'encombrement de l'espace. Les résultats obtenus avec des environnements statiques montrent la robustesse de notre approche au passage à l'échelle. La figure 2.5 montre deux problèmes (à gauche) et leurs solutions (à droite). Le volume balayé du chemin solution est affiché en vert. Comparé à l'approche RRT-Connect [152], notre algorithme produit des solutions avec des facteurs d'accélération allant de 4 à 10.

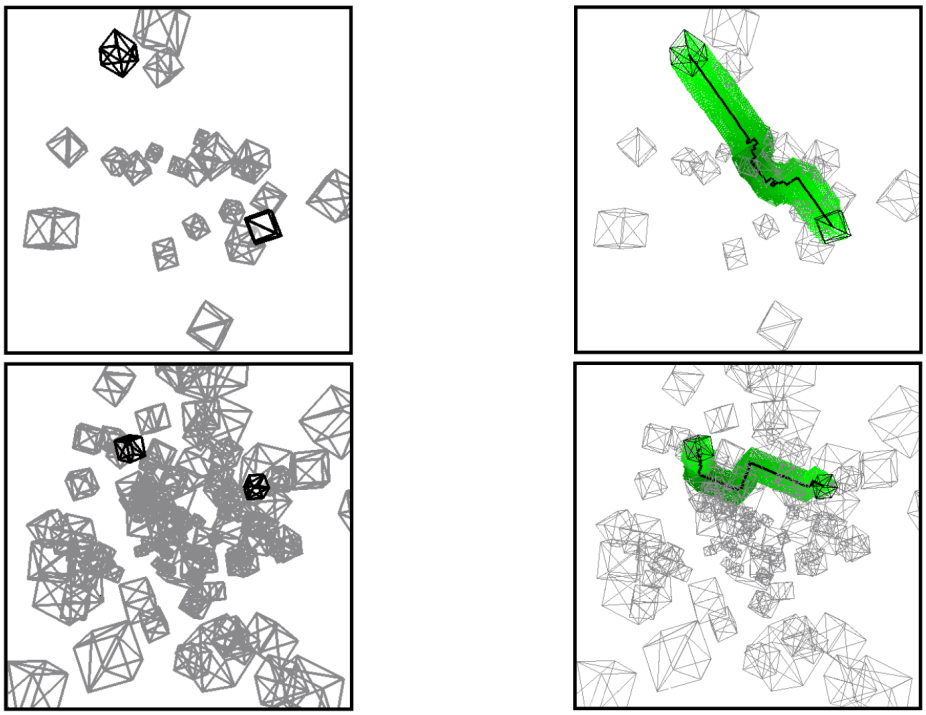


FIGURE 2.5 – Exemples de problèmes et de solutions dans des espaces encombrés en 3*D*.

2.2.2 Parallélisation de recherche statistique

Les algorithmes de recherche statistique sont appliqués pour la première fois à des problèmes de jeux [11] depuis les travaux de B. Bruegmann [29] [12]. Ils apparaissent à la même période que les algorithmes probabilistes dédiés aux problèmes de planification. J. Barraquand et J.C. Latombe [14, 16] proposent la méthode RPP pour la planification de mouvement probabiliste d'un bras articulé en 1991. B. Bruegmann réalise les premiers algorithmes Monte Carlo pour résoudre des problèmes du *Jeu de Go* en 1993. L.E. Kavraki et J.C Latombe [135, 137] proposent la méthode de planification PRM en 1995. S.M. LaValle [153] propose la méthode RRT en 1998. D. Billings *e*t al. [23] utilisent des méthodes d'élagage statistique dans un programme de poker. B. Bouzy et B. Helmstetter [25] testent de nombreuses combinaisons entre recherches arborescentes et algorithmes Monte Carlo dans des programmes de *Jeu de Go* en 2003. T. Cazenave et B. Helmstetter [38] combinent recherches tactiques et algorithmes Monte Carlo pour résoudre des problèmes de *Jeu de Go* en 2005. La recherche statistique permet de pallier la faiblesse de l'algorithme Alpha-beta [13], qui dépend fortement de la connaissance d'une bonne fonction d'évaluation. La recherche arborescente statistique initialement appelée Monte Carlo (pour des variations définies à effectif d'échantillonnage fixe sans mémorisation de l'arbre de recherche), évolue en Monte Carlo Tree Search (MCTS) avec les travaux de R. Coulom [54], de L. Kocsis et C. Szepesvári [141] et de G. Chaslot *et al.* [43] qui réalisent une construction de l'arbre de recherche guidée par une évaluation statistique incrémentale.

Algorithmes MCTS

Les algorithmes MCTS alternent 4 phases qui régissent la construction d'un arbre de recherche T :

11. Différentes classes de jeux apparaissent selon des propriétés de somme-nulle (où la somme des gains de chaque joueur est égale à zéro, autrement dit, où les jeux sont compétitifs, par opposition aux jeux où tous les joueurs peuvent perdre), à information complète ou incomplète (où les informations caractérisant l'état du jeu sont entièrement connues, où toutes les actions possibles de chaque joueur sont clairement définies, par opposition aux cas où l'état du jeu est partiellement observable), déterministe (où la chance n'intervient pas tels que le *Jeu des Échecs* ou le *Jeu de Go*, par opposition aux jeux avec un part de hasard tels que le Poker), séquentiel (où les actions sont appliquées les unes après les autres, par opposition aux jeux simultanés).

12. GOBBLE est le premier programme de *Jeu de Go* 9x9 à utiliser un algorithme Monte Carlo.

13. Alpha-beta est une spécialisation de l'algorithme Minimax ajoutant deux variables α et β permettant de faire des coupes dans la recherche arborescente. Les deux principales variations de l'algorithme Alpha-Beta sont Iterative-Deepening et MTD(f). Iterative-Deepening appelle itérativement Alpha-beta à des profondeurs croissantes. MTD(f) augmente les coupes en réduisant arbitrairement la différence entre les valeurs α et β.

- Sélection : Partant du nœud racine, il s'agit de choisir un nœud q de T jugé intéressant.
- Expansion : Partant de q, on définit un ensemble (de 1 à n) d'éléments Q' successeurs de q. La taille de Q' dépend de la situation et de la politique d'expansion choisie. Pendant cette phase, les éléments de Q' sont insérés dans T.
- Simulation : Pour chaque élément de Q', il s'agit de réaliser une à plusieurs simulations pour obtenir une évaluation statistique du gain possible associé à chaque élément.
- Rétropropagation : Pour chaque élément de Q' maintenant ajouté dans l'arbre, l'évaluation est rétropropagée dans les nœuds parents.

Les principales fonctions des algorithmes MCTS sont la fonction de sélection d'un nœud q et la fonction d'évaluation du gain d'un nœud après n simulations.

Sélection d'un nœud q

G. Chaslot *et al.* [44] ont proposé des stratégies de sélection à l'aide d'une valeur d'urgence égale à la somme d'une valeur de capture et d'une valeur de *pattern* :

- La valeur de capture favorise les coups de capture du joueur et les coups de protection contre les coups de capture de l'adversaire
- La valeur de *pattern* donne une connaissance experte d'une situation restreinte (par exemple dans le cas du *Jeu de Go*, en fonction de 8 intersections voisines).

Ainsi la stratégie de sélection permet de développer de nouvelles positions et de renforcer des positions actuelles, en considérant problèmes locaux et problème global.

Probabilité estimée de gain après n simulations

Dans sa formulation originale (présentée dans l'algorithme 4), la fonction d'évaluation d'un nœud (appelée stratégie de rétropropagation dans [43], opérateur de sauvegarde dans [54] ou encore *banditFormula* [14]) est égale à la proportion

14. Les travaux de recherche sur les algorithmes MCTS ont permis aux programmes de *Jeu de Go* (notamment CrazyStone, Fuego, Mango et MoGo) d'acquérir le niveau des joueurs professionnels. Le *Jeu de Go* possède un facteur de branchement supérieur au *Jeu des Échecs* et les performances d'une recherche Minimax diminuent. Complexité et facteur de branchement sont liés. Au *Jeu des Échecs*, ces valeurs sont de l'ordre de 10^{50} et 35, tandis que pour le *Jeu de Go*, elles sont de l'ordre de 10^{170} et 361. L'étude théorique des algorithmes MCTS a été réalisée simultanément dans des travaux de recherche sur la maximisation des gains face à n bandits manchots [141] (également appelés machines à sous). Il s'agit ainsi de choisir parmi les n bandits possibles, le bandit sur lequel jouer pour maximiser ses chances de gain.

de victoires obtenues après n simulations, soit

$$banditFormula(v,n) = \frac{1}{n}\sum_{n}^{1} r_i$$

avec r_i le résultat de la simulation i variant dans $\{0;1\}$, 0 en cas de défaite et 1 en cas de victoire.

```
Function MonteCarloEvaluation ( q , n)

nb_win ← 0;
for i ← 1 to n do
    q' ← q;
    if playout (q') = WON then
        nb_win ← nb_win + 1 ;
    end
end
return nb_win/n;
```

Algorithme 4 : Évaluation Monte Carlo d'un nœud q avec n simulation.

La fonction `playout` présentée dans l'algorithme 4 réalise une simulation aléatoire à partir de la position q'. Idéalement, cette simulation modifie q' jusqu'à l'obtention d'un nœud terminal. Dans les cas simples, une situation terminale est gagnée ou perdue. Alternativement, l'obtention d'une situation terminale n'est pas possible et la situation q' obtenue peut être indéterminée (si n est petit et qu'aucun gagnant n'apparaît après n itérations), nulle (si une situation peut être sans gagnant et sans perdant) ou encore évaluée à l'aide d'une heuristique.

La première [15] variante UCT [78] (Upper Confidence Tree) utilise la fonction

$$banditFormula(v,d,n) = \frac{v}{(v+d)} + \sqrt{K\log(n)/(v+d)}$$

pour l'évaluation d'une action où K est une constante choisie empiriquement, n est le nombre de simulations achevées et commençant par cette action, v est le nombre de victoires obtenues en commençant par cette action, d est le nombre de défaites obtenues en commençant par cette action. Le premier terme $\frac{v}{(v+d)}$ est dit d'exploitation et favorise les nœuds avec un nombre de succès important. Le second terme est dit d'exploration et favorise les nœuds peu simulés. Pour un petit

15. La première variante UCT utilise cette fonction d'évaluation appelée UCB1 (Upper Confidence Bound 1). P. Auer *et al.* [12] ont présenté les différences entre les variantes UCB1, UCB2, ε_n-GREEDY et UCB1-TUNED dans le cadre de l'étude du problème de la maximisation des gains face à n bandits manchots.

nombre de simulations (et notamment pour $(v+d)=0$), L. Kocsis et C. Szepesvari [141] proposent de définir expérimentalement un seuil de simulation en dessous duquel l'évaluation d'un nœud est égale à une constante [16]. Ils ont également démontré la convergence de la variante UCT vers une recherche Minimax et donc vers une solution optimale.

C. Browne *et al.* [28] ont présenté une revue détaillée des différentes variantes et applications des algorithmes MCTS.

Parallélisation MCTS

L'utilisation itérative d'échantillonnages aléatoires indépendants fait des algorithmes MCTS une cible a priori adéquate à un développement parallèle, permettant de réaliser plus de simulations dans un intervalle de temps équivalent. Cependant le calcul d'une évaluation statistique soulève le problème de la mise en commun des évaluations réalisées en parallèle, achevée par des étapes de synchronisation [17] ou par des séquences de communication [18].

Dans tous les cas d'architectures parallèles, un programme parallèle peut être plus lent qu'un programme séquentiel réalisant un calcul équivalent. Pour nous abstenir des contraintes d'organisation de la mémoire partagée [19], nous avons proposé 3 algorithmes de parallélisation [40, 39] des algorithmes UCT à l'aide du paradigme de parallélisme à passage de messages. Dans chacun de ces algorithmes, un nœud q appliqué au *Jeu de Go* correspond à un ensemble de positions (pour noir et pour blanc), une éventuelle situation de *ko* (annonçant un coup non-autorisé) et une couleur (soit le tour de jeu). Ces algorithmes ont été implémentés sur un cluster d'ordinateurs composé de 16 [20] processeurs Intel dual core 1.86 GHz avec 2GB de RAM reliés sur un réseau Gigabit avec un serveur Intel Xeon

16. Pour le programme Mango, G. Chaslot *et al.* ont utilisé $K = 0.7$ avec un seuil de 30 avec la formule $banditFormula(v,n) = v/n + \sqrt{K\ln(n)/v}$.

17. Adéquates aux architectures à couplage serré.

18. Adéquates aux architectures à couplage lâche.

19. La programmation parallèle à l'aide de la librairie PTHREAD apporte une solution de communication par mémoire partagée à l'aide de sémaphores, mutex et barrières de synchronisation. La mise en œuvre de ces mécanismes étant fastidieuse, il est aujourd'hui plus accessible d'utiliser les fonctionnalités automatiques proposées par OpenMP au travers des directives de compilation. La parallélisation des algorithmes peut se résumer en l'identification des séquences d'instructions d'un programme séquentiel partageables entre plusieurs processeurs indépendants. Pour faciliter la parallélisation des algorithmes, OpenMP propose des primitives de partage de l'exécution entre plusieurs threads, des clauses permettant de définir le comportement de la parallélisation et des primitives de synchronisation. Il reste que le *mapping* de la mémoire est une variable déterminante des possibilités d'accélération d'un programme parallèle réalisé avec OpenMP. Ainsi le for-parallèle sera plus rapide ou plus lent qu'un for séquentiel, en fonction de l'adéquation entre mappage mémoire et opérations des séquences d'instructions. `http://www.openmp.org`.

20. Plus précisément, 8 processeurs en 2007 et 16 en 2008.

3.20 GHz avec 4GB de RAM à l'aide des librairies MPICH [21], LAM-MPI [22] et OpenMPI [23], conformes au spécifications du standard MPI .

Dans l'algorithme 5 appelé Single Run Parallel, chaque *processus-client* réalise une recherche UCT indépendante. Les *n processus-clients* utilisent un générateur de séries aléatoires initialisé avec des valeurs différentes pour chaque *processus-client*, de sorte que chacun développe un arbre UCT légèrement différent. Le développement de l'arbre UCT est réalisé sans communication. Les nœuds les plus intéressants se retrouvant statistiquement dans chacun des arbres UCT des *processus-clients*, un *processus-maître* regroupe les playouts et les victoires obtenus pour les coups possibles évalués par les *processus-clients*, et le *processus-maître* établit le meilleur coup en agrégeant les moyennes des victoires obtenues.

Dans les algorithmes 7 et 8 appelés Multiple Run Parallel, chaque *processus-client* commence par une séquence UCT indépendante et poursuit par l'évaluation de séquences définies par le *processus-maître*. Le développement de l'arbre UCT est dirigé par le *processus-maître*. Les *processus-clients* conservent leurs arbres UCT locaux et retournent les statistiques (nombre de simulations et nombre de victoires) résultant de leur estimation locale. Après collecte des résultats de l'évaluation des *n* séquences par les *processus-clients*, le *processus-maître* établit les *n* nouvelles séquences à évaluer et les répartit. Évaluation et répartition sont répétées durant le temps *t*. A l'issue du temps *t* imparti, le *processus-maître* établit le meilleur coup en sommant playouts et victoires.

21. `http://www.mpich.org`.
22. `http://www.lam-mpi.org`.
23. `http://www.open-mpi.org`.

Function `singleRunParallelUCTMove` (q, t)

$best \leftarrow$ -1;
$(W, G) \leftarrow$ `initialParallelUCTMove` (q, t);
for $i \leftarrow$ `0` **to** `nb_next` *(q)* **do**
  $best \leftarrow$ `max` $(best, W_i/G_i)$;
end
return *best*;

Function `initialParallelUCTMove` (q, t)

`broadcast` (q, t);
$(W, G) \leftarrow$ `receiveUCTSequences` ( );
for $i \leftarrow$ `1` **to** n **do**
  $(W', G') \leftarrow$ `receiveUCTSequences` ( );
  $(W, G) \leftarrow (W+W', G+G')$;
end
return (W, G);

Algorithme 5 : Single Run Parallel (*master*) pour une position q.

Function `singleRunParallelUCTMoveSlaveLoop` ( )

$state \leftarrow CONTINUE$;
while $state = CONTINUE$ **do**
  $state \leftarrow$ `SingleQueryUCTSlaveLoop` ( );
end
return;

Function `SingleQueryUCTSlaveLoop` ( )

if `receive` *(q, t)* = *END_GAME* **then**
  $ret \leftarrow END\_GAME$;
else
  $(W, G) \leftarrow$ `playUCTSequences` (q, t);
  `send` (W, G);
  $ret \leftarrow CONTINUE$;
end
return *ret*;

Algorithme 6 : Single Run Parallel (*slave*) pour une position q.

```
Function multipleRunsParallelUCTMove (q, t)

best ← -1;
(W, G) ← initialParallelUCTMove (q, t_0);
t ← t-t_0;
while t ≥0 do
    broadcast (W, G, t);
    (W', G') ← receiveUCTSequences ( );
    for i ← 1 to n do
        (W', G') ← (W', G') + receiveUCTSequences ( );
    end
    (W, G, t) ← (W+W', G+G', t-t');
end
for i ← 0 to nb_next (q) do
    best ← max (best, W_i/G_i);
end
return best;
```

Algorithme 7 : Multiple Run Parallel (*master*) pour une position q.

```
Function multileRunParallelUCTMoveSlaveLoop ( )

state ← CONTINUE;
while state = CONTINUE do
    state ← SingleQueryUCTSlaveLoop ( );
    while state = CONTINUE do
        if receive (W, G, t) = END_LOOP then
            state ← END_LOOP;
        else
            (W, G) ← continueUCTSequences (t);
            send (W, G);
            state ← CONTINUE;
        end
    end
end
return;
```

Algorithme 8 : Multiple Run Parallel (*slave*) pour une position q.

```
Function atLeavesParallelUCTMove (q, t)

best ← -1;
broadcast (q, state);
(W,G) ← ({0,0,0,...},{0,0,0,...});
while t ≥0 do
    S ← getUCTSequence ( );
    broadcast (S);
    W' ← 0;
    for i ← 1 to n do
        W' ← W' + receive ( );
    end
    W' ← W'/n;
    for i ← 0 to nb_next (q) do
        (W_Si,G_i) ← (W_Si+W'_i, 1);
    end
end
for i ← 0 to nb_next (q) do
    best ← max (best, W_i/G_i);
end
return best;
```

Algorithme 9 : At-The-Leaves Parallel (*master*) pour une position q.

Dans les algorithmes 9 et 10 présentant les deux parties de l'algorithme appelé At-The-Leaves Parallel, les *processus-clients* développent des recherches UCT au niveau des feuilles de l'arbre du *processus-maître*. Pour chaque nouvelle feuille de l'arbre UCT, le *processus-maître* envoie la séquence d'accès à la feuille considérée à chaque *processus-client*. Chacun des *processus-clients* reçoit la même séquence et réalise m playouts. Chaque *processus-client* retourne la moyenne des résultats obtenus et le *processus-maître* établit le meilleur coup par moyenne des moyennes des *processus-clients*. Cet algorithme demande plus de communications que les algorithmes 7, 8 et 9 en réalisant un meilleur partage des informations établies par les recherches UCT locales. L'algorithme At-The-Leaves Parallel est adapté à des problèmes avec des playouts lents.

```
Function atLeavesParallelSlaveLoop ( )

state ← CONTINUE;
while state ≠ END_GAME do
    (q, state) ← receive ( );
    while state ≠ END_LOOP do
        state ← AtLeavesQuerySlaveLoop ( );
    end
end
return;
```

```
Function AtLeavesQuerySlaveLoop ( )

if receive (S) = END_LOOP then
    ret ← END_LOOP;
else
    playSequence (S);
    score ← 0;
    for i ← 0 to m do
        score ← score + randomPlayout ( );
    end
    send (score / m);
    ret ← CONTINUE;
end
return ret;
```

Algorithme 10 : At-The-Leaves Parallel (*slave*) pour une position q.

2.3 Conclusion

Dans ces travaux sur les algorithmes probabilistes appliqués aux problèmes de planification de mouvement et aux problèmes de décision dans les jeux à information complète, nous avons étudié les interactions de leurs composants dans des alternatives séquentielles et parallèles.

Chapitre 3

Robotique humanoïde

Sommaire

Nous décrivons dans ce chapitre les résultats de nos travaux de recherche en robotique humanoïde concernant la génération de mouvement, la vision temps-réel embarquée, l'optimisation morphologique et la génération de stratégies collectives pour des équipes de robots humanoïdes hétérogènes.

Le développement des mouvements des robots humanoïdes [1] est justifié par les besoins en robots dotés de capacités de déplacement sur différentes surfaces (planes ou escamotées, molles ou dures, pentues en descente ou en montée) pouvant ainsi se substituer aux humains pour réaliser des tâches [2]. Dans le contexte

1. Les robots humanoïdes font partie de la catégorie des robots à pattes dont la complexité des déplacements varie inversement à leur nombre de pattes. Les robots avec plus de 4 pattes possèdent des mouvements très stables par compensation des points d'appui entre les différentes pattes. Les robots à 3 pattes et moins impliquent une modélisation des positions d'équilibre déclinant des points d'appui utilisés au cours de leur déplacement.

2. Le *DARPA Robotics Challenge* (DRC) est une compétition de robots sur le thème de l'aide à l'action dans les situations de catastrophes naturelles d'origine humaine. Cette compétition est divisée en 3 parties dans lesquelles matériel et logiciel occupent des places importantes : Le *Virtual Robotics Challenge* qui évalue les possibilités en simulation ; Le *DRC Trials* et le *DRC Finals* sont 2 séries d'épreuves à réaliser avec un robot réel. L'objectif est de réaliser des taches très diverses comme conduire un véhicule utilitaire, se déplacer dans des décombres, retirer des débris du sol,

de la *RoboCup*[3], nous avons utilisé le robot humanoïde *NAO*[4] pour modéliser les mouvements humanoïdes, développer des algorithmes de vision temps réel embarquée, présenter des travaux précurseurs sur l'optimisation morphologique des humanoïdes et définir des comportements collectifs dans le cadre appliqué de la coopération d'une équipe de robots footballeurs. Les expérimentations ont été

ouvrir une porte et entrer dans un bâtiment, monter une échelle, traverser sur une passerelle, utiliser une perceuse, fermer une vanne, connecter des tuyaux d'incendie. Les contraintes fixées sont d'avoir des jambes avec 6 ddl, des bras avec 7 ddl, des mains avec 2 ou 3 doigts, une perception en stéréo-vision et une nappe laser. Le *Virtual Robotics Challenge* s'est déroulé en juin 2013. L'environnement de simulation combinant le simulateur d'environnements physiques et d'interactions *Gazebo* et l'environnement de robotique de service *ROS*, a été fourni par l'*Open Source Robotics Foundation* (OSRF). L'équipe *Institute for Human and Machine Cognition* de Floride a remporté cette première épreuve. Le *DRC Trials* s'est déroulé en décembre 2013. *SCHAFT*, vainqueur de cette seconde épreuve, est un robot humanoïde construit sur une base HRP3L-JSK possédant des moteurs bio-inspirés du principe os-muscle-tendon à refroidissement liquide permettant d'obtenir couple et rapidité sans surchauffe. A cette base constituée de 2 jambes, 2 bras ont été ajoutés au niveau du bassin (ce qui en fait un robot humanoïde sans réel torse). L'ensemble produit un humanoïde à la silhouette trapue. Ces jambes sont connues pour avoir été conçues par J. Urata. La troisième épreuve est prévue pour juin 2015.

3. La *RoboCup* est une compétition de robotique initiée en 1997 dont l'objectif est de confronter Intelligence Artificielle et Robotique dans un but commun. Initialement dédiée de par son nom aux compétitions de robots footballeurs, elle regroupe aujourd'hui des épreuves de sauvetage après désastre naturel, gestion de ville en situation de crise, assistance à domicile, de gestion de stock et de manipulation industrielle, avec pour objectif final de rivaliser avec les humains.

4. *NAO* est un robot humanoïde de 58[cm] développé par *Aldebaran Robotics*. Il possède 14 ddl, une centrale inertielle (avec accéléromètre et gyromètre), de sonars ultrasons, de capteurs de pressions et de bumpers aux pieds, de micros, de haut-parleurs et de 2 caméras indépendantes (l'une permettant de voir le sol devant lui et l'autre permettant de voir plus loin, l'une et l'autre n'étant pas utilisables simultanément). Il est utilisé dans les compétitions de *RoboCup Standard Plateform League* (SPL) depuis 2007. Les matchs sont aujourd'hui sur des terrains de 10[m] par 7[m] avec 5 *NAO* par équipe. Les équipes sont différenciées par des couleurs de *t-shirts* bleu et rouge. L'environnement est simplifié pour limiter les problèmes de perception : le sol est principalement vert, avec des lignes blanches mais les goals sont jaunes pour les 2 équipes (ce qui implique une localisation éprouvée et une relocalisation difficile en cas de chute). Les balles sont oranges et mesurent 6.5[cm] mais changent légèrement selon les années. Les séquences de jeu des matchs sont orchestrées par un contrôleur de jeu relié par connexion wifi à chaque robot. En cas de perte de connexion, les robots continuent à jouer et les séquences de jeu sont sélectionnées manuellement à l'aide d'un bouton-poussoir sur le torse de chaque *NAO*. Les robots sont placés au bord du terrain en début de match et sont invités par le contrôleur de jeu à se placer automatiquement sur le terrain. Les erreurs de placement sont rectifiées par des opérateurs. L'arbitrage est actuellement assuré par un arbitre humain et des assistants, qui relèguent les fautes de jeu au contrôleur de jeu. Les communications entre *NAO* sont possibles mais limitées à une certaine bande passante.

réalisées en *RoboCup SPL*[5] et en *RoboCup 3DSSL*[6].

3.1 Génération de mouvement

Pour un robot humanoïde, se déplacer implique considérer les mouvements de ses différentes articulations et l'équilibre de l'ensemble de son corps dans les différentes phases du mouvement. Parmi les nombreuses approches possibles, les méthodes de suivi de trajectoire ont permis de générer en ligne des mouvements en s'appuyant sur une résolution hors-ligne d'équations dynamiques, en utilisant des articulations à gain élevé pour obtenir des trajectoires prédéfinies. Pour les déplacements du robot P2[7], K. Hirai *et al.* [89] ont utilisé un modèle d'inclinaison dynamique par combinaison des différences des réactions au sol avec les différences des forces d'inertie au torse. Q. Huang *et al.* [91] ont mis en avant le rôle des extrémités des chaînes articulées, en adaptant les mouvements de pied et de hanche au support et au mouvement désiré. Pour réaliser un humanoïde avec des phases de mouvements sans contact avec le support, F. Pfeiffer *et al.* [184] ont proposé de renforcer la mesure des forces de contact au sol. Les résultats montrent des différences entre la solution embarquée et la solution avec calculs déportés sur un ordinateur hors du robot *Johnnie*[8]. Pour permettre la navigation d'un robot humanoïde dans un espace encombré, J. Kuffner *et al.* [147] ont proposé de précalculer un ensemble de trajectoires dont les positions finales sont évaluées par un parcours d'arbre A*. Il apparaît dans ces travaux marquants sur la génération de mouvement pour les robots humanoïdes, l'importance des mouvements de

5. La *RoboCup Standard Plateform League* (SPL) a pour objectif d'opposer des plateformes ayant des caractéristiques mécaniques identiques. Les robots sont entièrement contrôlés par des programmes embarqués. La première plateforme utilisée fut le robot-chien *AIBO-ERS* de *Sony*. La plateforme actuellement utilisée est le robot humanoïde *NAO*.

6. La *RoboCup 3D Soccer Simulation League* (3DSSL) a pour objectif de faciliter le développement de techniques d'apprentissage, d'optimisation et d'algorithmes de coordination multi-robots en robotique humanoïde. Les compétitions sont réalisées dans l'environnement de simulation *SimSpark-rcssserver3d* [178, 207] et impliquent des équipes de 11 robots humanoïdes *NAO* contrôlées par des programmes indépendants.

7. Les performances des premiers robots humanoïdes fabriqués par *Honda* sont très appréciables. P2 fait partie d'une gamme plus large, comprenant son prédécesseur P1 le premier robot humanoïde et son successeur P3, plus petit et plus léger, permettant plus d'interactions. P2 mesure 1.8[m], est large de 60[cm], pèse 210[Kg], est composé de jambes à 6 ddl, de bras à 7 ddl et de mains à 2 ddl. Il est contrôlé par 4 processeurs *SPARC-II* et le système temps-réel *VxWorks*. Sa batterie de 20[Kg] lui octroie une autonomie de déplacement de 20[min].

8. Le robot *Johnnie* comprend 17 ddl dont 6 par jambe. Malgré une taille de 1.8[m], il pèse un poids de 40[Kg] qui en fait un bon candidat pour la réalisation de mouvements de saut. Les vitesses de déplacement obtenues sont de 1.2[km.h^{-1}] avec le calculateur embarqué et de 2.0[km.h^{-1}] avec le calculateur déporté.

jambe et de leur paramétrisation en fonction des articulations impliquées et des déplacements désirés.

Par assemblage de séquences choisies parmi des primitives de mouvement de jambe, S. Behnke [19] a proposé de générer en ligne les trajectoires d'un robot humanoïde de 60[cm] et à 19 ddl, commandé par servomoteurs. Les calculs sont entièrement embarqués. La marche résultante est omnidirectionnelle et modulable selon le type de mouvement désiré (*i.e.* sens de déplacement, vitesse longitudinale et vitesse de rotation). Les résultats montrent une marche composée de pas rasant le sol, avec une vitesse de déplacement en marche avant de 14[m.s^{-1}] avec des pas moyens de 9.14[cm] et une vitesse de déplacement en marche latérale de 3.8[m.s^{-1}] avec des pas latéraux de 2.41[cm] en moyenne. Le déplacement en rotation est de 15[deg] par pas pour une vitesse moyenne de 23.6[deg.s^{-1}].

C. Graf et T. Röefer [83] ont proposé d'éliminer la phase de double support (*i.e.* instant dans la marche où les deux pieds sont au sol) en ajustant les points d'appui de chacun des pas dans le temps. Sur ce principe, ils proposent une marche rapide en boucle ouverte et une marche plus lente en boucle fermée. Les vitesses maximales obtenues avec le robot *NAO* sont de 28[cm.s^{-1}] en marche avant, de 17[cm.s^{-1}] en marche arrière, de 7[cm.s^{-1}] en marche latérale et de 90[deg.s^{-1}] en rotation. Par la suite, ils ont également proposé de développer l'agilité et de renforcer la stabilité de la marche humanoïde permettant de mieux compenser les perturbations extérieures[82]. La position du pendule inversé [134] modélisant le robot est mise à jour en fonction des données capteurs. Les vitesses maximales obtenues avec le robot *NAO* sont de 31[cm.s^{-1}] en marche avant avec des pas de 7[cm], de 12[cm.s^{-1}] en marche latérale avec des pas latéraux de 8[cm], de 22[cm.s^{-1}] en marche arrière, et de 92[deg.s^{-1}] en rotation. En translation avant rapide, les erreurs moyennes de position sur le centre de masse sont de 7.76[mm], pour des erreurs de 5.88[mm] à une vitesse de 10[m.s^{-1}].

F. Xue *et al.* [234] ont proposé une variante lissant les trajectoires du centre de masse et considérant 2 articulations comme élastiques. La marche résultant est omnidirectionnelle. Les expérimentations sur le robot *NAO* montrent une vitesse maximale de 33[cm.s^{-1}] en marche avant, de 20[cm.s^{-1}] en marche arrière, de 11[cm.s^{-1}] en marche latérale et de 90[deg.s^{-1}] en rotation.

A. Schmitz *et al.* [202] ont proposé une amélioration de leurs travaux précédents [19] en ajoutant une politique de planification de pas hors-ligne pour une génération de trajectoires moins réactives et pour un meilleur enchaînement des mouvements de déplacements avec les mouvements d'interaction avec une balle. Le précalcul de 17000 séquences de pas a été réalisé avec A*, en considérant que les séquences finissent avec l'autre pied par symétrie des premières. Avec cet ensemble de chemins, ils ont utilisé un réseau de neurones à 3 couches pour apprendre des politiques permettant de reproduire les chemins correspondants en fonction de l'état courant du modèle. Ils ont ainsi défini 4 politiques pour com-

mencer un mouvement sur un pied et finir une action sur un pied (*i.e.* sur le même ou sur le pied opposé au premier pas). Les résultats montrent des déviations angulaires plus petites, des actions résultantes moins fortes et des chemins plus courts qu'avec une approche réactive.

Pour produire des séquences de mouvements avec une allure plus proche de la démarche des humains, nous avons proposé[98] un algorithme de déplacement basé sur les enchaînements possibles entre des primitives de déplacement. Les primitives sont regroupées dans 2 groupes, l'un pour les mouvements de translation et l'autre pour les mouvements de rotation. La génération de nouvelles primitives spécifiques à la rotation permet d'obtenir des vitesses intéressantes en rotation sur place et autour d'un point situé sur l'axe longitudinal du robot. Les mouvements sont générés en ligne, sans connaissance a priori de la position finale du déplacement en cours. Nous avons étudié des allures de marche avec un calcul du point de moment nul (ZMP) [225] appliqué à un modèle de pendule inversé [134] (abrégé 3D-LIP, ce modèle est représenté par un unique pendule inversé avec une jambe télescopique sans masse, reliant la position d'appui du pied au sol avec le centre de masse de l'ensemble du robot). La hauteur du centre de masse est maintenue constante et aucun couple n'est pris en compte entre la masse et le pied d'appui du robot. Le robot est supposé marcher sur une ligne horizontale-plan, avec une alternance de phases d'appuis sur un seul pied. L'instant de contact avec les deux pieds, appelé point de double support est considéré comme instantané.

Les équations définissant les relations entre une position (x_G, y_G, z_G) de centre de masse et le point ZMP (nommé P^*) sont données par [134] :

$$\ddot{x}_G = \frac{g}{z_G}(x_G - x_{P^*}) \tag{3.1}$$

$$\ddot{y}_G = \frac{g}{z_G}(y_G - y_{P^*}) \tag{3.2}$$

avec g pour valeur de pesanteur.

Dans nos travaux, le point ZMP est fixé pour chaque phase de support. Les formes hyperboliques des courbes suivies par le centre de masse à chaque pas sont définies par :

$$\begin{aligned} x_G(t) &= (x_G^{i(n)} - x_{P^*})\cosh(t/T_C) \\ &\quad + T_C \dot{x}_G^{i(n)} \sinh(t/T_C) + x_{P^*} \end{aligned} \tag{3.3}$$

$$\begin{aligned} \dot{x}_G(t) &= (x_G^{i(n)} - x_{P^*})\sinh(t/T_C)/T_C \\ &\quad + \dot{x}_G^{i(n)} \cosh(t/T_C) \end{aligned} \tag{3.4}$$

avec n le pas de discrétisation des courbes, $x_G^{i(n)}$ et $\dot{x}_G^{i(n)}$ sont respectivement la coordonnée en x initiale du centre de masse et la vitesse initiale sur l'axe Ox de la

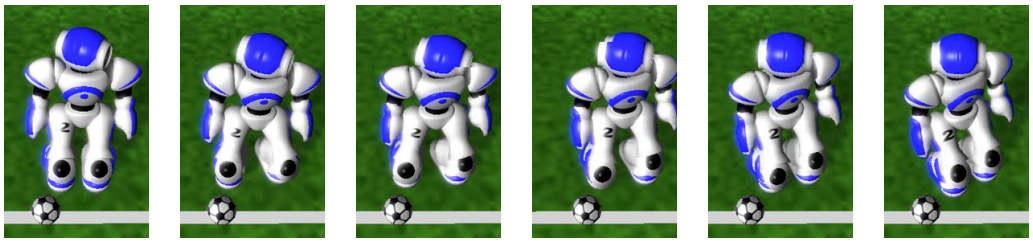
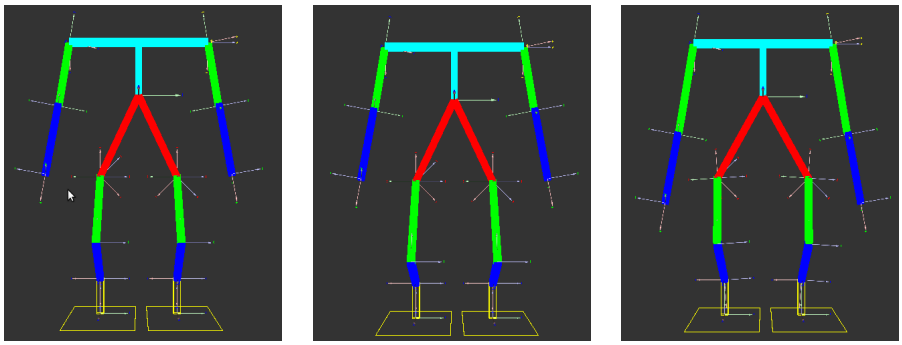


FIGURE 3.1 – Quelques étapes d'un mouvement de rotation sur place.

$n^{\text{ème}}$ valeur sachant $T_C = \sqrt{\frac{z_G}{g}}$.

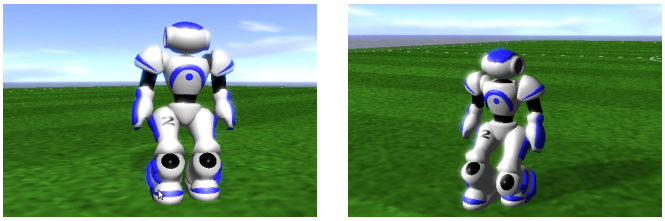
Les expérimentations ont été réalisées avec succès dans l'environnement de simulation *SimSpark-rcssserver3d* [178, 207] et ont été appliquées dans les épreuves qualificatives et les compétitions de *RoboCup-3DSSL* [120] avec des équipes homogènes de 11 robots *NAO* en simulation. La figure 3.1 montre une succession de vues de dessus correspondant à une rotation sur place. La stabilité des déplacements résultants a permis d'interagir avec la balle et d'accomplir des *dribbles* vers les buts adverses afin de marquer. La vitesse résultante est cependant inférieure aux vitesses des 3 meilleures équipes de la compétition. La vitesse en marche avant obtenue est de $0,22$ [m.s^{-1}] et la vitesse angulaire de $1,0$[rad.s^{-1}]. Ces vitesses sont modifiables en fonction du temps choisi pour chaque itération qui définit l'envoi de commandes aux effecteurs. La vitesse et la stabilité de déplacement résultante est supérieure à une première solution de génération des mouvements [243], établie par échantillonnage des mouvements sur un robot *NAO* réel.

Composé de 5 ddl à chaque jambe et d'une articulation commune au niveau des hanches, *NAO* possède la particularité d'avoir une locomotion à 11 ddl. Cette articulation commune implique une dépendance entre les mouvements des jambes. Avec pour contrainte ce sous-actionnement, nous avons proposé la première solution exacte de calcul de l'angle de cette articulation commune [94]. Ces travaux nous ont permis de passer d'un modèle spécifique dédié à *NAO* à un modèle générique valide pour tout humanoïde. Afin de simplifier la génération des primitives de mouvements des humanoïdes, nous avons proposé [95] une procédure de génération automatique des paramètres de modélisation géométrique des chaînes cinématiques. Nous avons utilisé la convention de Denavit-Hartenberg [59] modifiée par W. Khalil et J.F. Kleinfinger [138] (notée DHKK). Notre solution présente l'avantage de calculer automatiquement les paramètres géométriques du modèle à partir de la liste des orientations et des points d'attache successifs des axes des articulations. L'unification du formalisme permet également un gain de temps dans le cas de modification du modèle.



(a) (b) (c)

FIGURE 3.2 – Variantes du squelette du robot *NAO* en position de départ.



(a) (b)

FIGURE 3.3 – Robot *NAO* en position de départ et en mouvement.

Cette procédure de génération automatique a été appliquée avec succès lors de premiers tests dans les épreuves qualificatives et de compétitions de la *RoboCup 3DSSL* pour des équipes de robots homogènes[123] et pour des équipes de robots hétérogènes[118]. Nous avons utilisé des modèles géométriques directs pour valider par un affichage le bon fonctionnement de la procédure. En ajoutant des mouvements simples, nous avons contrôlé les mouvements des différentes branches du squelette. Les figures 3.2 *(a)*, *(b)* et *(c)* montrent différentes variantes du squelette du robot *NAO* en position de départ : le plus à droite est la version standard avec des fémurs de 0.12[m], suit au centre un squelette avec des fémurs allongés de 0.14[m] et à droite un squelette avec des hanches écartées et des bras plus longs. Les figures 3.3 *(a)* et *(b)* montrent un robot *NAO* en position de départ et un autre en cours de déplacement dans l'environnement de simulation *SimSpark-rcssserver3d* [178, 207].

3.2 Vision temps-réel embarquée

Dans le cadre des compétitions de la *RoboCup SPL*, nous avons développé un module de vision temps-réel embarquée. Au cours de la conception d'une première version d'un module de vision embarquée sur le robot humanoïde *NAO* [9], nous avons proposé [97] une procédure semi-automatique de calibration des caméras permettant de réaliser l'ensemble des traitements de vision sur des ensembles de pixels fixés dans un espace composé d'objets monochromes distincts. Les principaux traitements considérés ont été limités à la recherche de groupe de pixels ayant un index commun dans une table de correspondances précaculée et à des calculs de moments pour identifier des formes. Les temps de capture des images et les temps de traitement ont été comparés dans les espaces de couleurs *hsv*, *rgb*, *yuv* et *yuv422* et pour les résolutions *kVGA*, *kQVGA* et *kQQVGA* afin d'assurer la bonne évaluation des temps d'exécution dans le module de vision embarquée. Nous avons réalisé une procédure automatique d'enrichissement des tables de correspondances à partir d'un ensemble d'images. La validation des traitements a été réalisée par expertise sur un ensemble d'images supplémentaires et une procédure permettant de rejouer hors-ligne les traitements réalisés a été utilisée. Ces premiers travaux nous ont permis de réaliser une interface de construction des tables de correspondances [93, 96], connectée au robot *NAO* permettant de :

- Fixer les paramètres de calibration des caméras
- Construire en mode connecté les tables de correspondances
- Visualiser directement le résultat des tables de correspondances
- Visualiser directement le résultat des traitements de vision

Afin de réduire les interférences entre des objets de couleur proche, nous avons étudié les solutions de segmentation automatique en régions.

A.M. Arsenio [11] a proposé pour le robot humanoïde *Cog* [10] d'indexer les couleurs en fonction de leur contexte. Avec une segmentation en régions, les objets sont associés à une apparence (définie comme une hiérarchie de couleurs, de luminances, de formes et de textures), à des caractéristiques physiques (comme le poids et les déformations), à un mode d'actionnement, à des informations cinéma-

9. La première version du robot humanoïde *NAO* utilisée dans le cadre des compétitions de la RoboCup est la V2. Elle possède un processeur AMD Geode cadencé à 550Mhz dédié à la vision et 2 caméras dotées de capteurs OV7670 permettant de récupérer des images de 640x480 à une fréquence de 30fps dans l'espace de couleur YUV422. Les angles de vue de ces caméras sont de 58[deg] dans la diagonale, 48[deg] horizontal et 37[deg] vertical. A partir de la fin de l'année 2010, la version V3.3 possède un processeur Intel Atom cadencé à 1.6 GHz dédié à la vision et 2 caméras dotées de capteurs MT9M114 permettant de récupérer des images de 1280x960 à une fréquence de 30 fps dans l'espace de couleur YUV422.

10. Le robot humanoïde *Cog* apprend par réplication des comportements humains. Dans ce contexte, la vision est un organe de perception très important permettant d'identifier les objets en fonction de leur utilisation : une corde en oscillation devient ainsi un pendule.

tiques, à des dépendances de positionnement dans le monde, à une représentation interne hiérarchique (d'assemblage d'autres objets) et aux actions possibles associées. La reconnaissance des objets est limitée à 20 couleurs. Les objets inconnus sont automatiquement ajoutés dans la nouvelle base de connaissances. Les résultats obtenus avec un jeu de test de 100 images montrent une reconnaissance moyenne de 95% pour les objets communs (tels que canapé, table et chaise) et la nécessité d'ajouter une méthode de reconnaissance de visages.

B. Sheela Rani *et al.* [189] ont proposé un algorithme de traitement d'image pour le suivi des opérations de soudure par thermographie. Un traitement de détection de forme est appliqué sur les groupes regroupés en régions, pour permettre de les identifier sans opération de détection de contours.

S. Gächter[77] a réalisé une étude comparative des méthodes de segmentation (par contours, par seuillage, par *clustering*, par croissance de régions, par fraction/fusion) pour la détection de lignes et de plans dans des images contenant des informations de distance aux objets. Les résultats montrent de meilleurs performances avec des images de ce type par rapport aux résultats obtenus avec des images classiques sans profondeur de champ.

Partant de travaux en vision temps réel sur *Aibo* [31], nous avons proposé [115] l'algorithme 11 de segmentation pyramidale temps-réel en régions en 2 étapes sans paramétrage pour le robot humanoïde *NAO*. La segmentation recherche progressivement des regroupements de 3×3, 9×9 puis 27×27. R' définit l'ensemble des sous-régions ayant des attributs communs. En fonction du temps de calcul disponible, il est possible de rechercher différents attributs pour le regroupement en régions. L'algorithme manipule principalement des ensembles de pixels associés à des attributs, initialement composé d'un pixel avec des attributs simples. Les attributs peuvent différer selon les groupes et le regroupement est défini par comparaison de ses attributs. La segmentation en région est enrichie par l'utilisation d'une table de correspondances.

Les résultats comparent les contours obtenus par segmentation en régions dans l'espace de couleurs *YUV* avec les contours obtenus par un filtrage de Canny en niveau de gris. Les temps de calculs, évalués sur un robot humanoïde *NAO*, montrent des temps comparables. Un sous-échantillonnage est ajouté pour démontrer la robustesse de l'algorithme proposé.

Un algorithme dédié à la détection de lignes a été implémenté dans les dernières versions[100, 99] du module de vision embarqué pour robot humanoïde *NAO*. L'extraction des lignes a permis d'améliorer les fonctions de localisation et de développer des comportements spécialisés pour des séquences de jeu avec des placements particuliers.

Function `regionSegmentation` (*img*)

$R \leftarrow \{R_0, R_1, R_2\}$;
$R_0, R_1, R_2 \leftarrow \emptyset$;
$S = \{3 \times 3,\ 9 \times 9,\ 27 \times 27\}$;
for *i in* 0 *to* `size` *(img)* **do** $R_0 \leftarrow R_0 \cup \{img[i]\}$;
for *i in* 0 *to* 3 **do** **for** *each* $e \in R[i]$ **do**
  $R[i+1] \leftarrow R[i+1] \cup e$;
  $R[i] \leftarrow R[i] - e$;
end
$R' \leftarrow \emptyset$;
for *i in* 0 *to* 3 **do** **for** *each* $e \in R[i]$ **do**
  $nreg \leftarrow FALSE$;
  for *each* $r' \in R'$ **do** **if** `regroup` *(e , r')* **then**
    $R[i] \leftarrow R[i] - e$;
    $R' \leftarrow R' \cup e$;
    $nreg \leftarrow TRUE$;
  end
  if *nreg* **then**
    $R_{new} \leftarrow e$;
    if `regroup` *(e , R[i])* **then**
      $R_{new} \leftarrow R_{new} \cup R[i]$;
      $R[i] \leftarrow \emptyset$;
    end
    $R' \leftarrow R' \cup R_{new}$;
  end
end

Algorithme 11 : Segmentation pyramidale temps-réel en régions.

3.3 Optimisation morphologique

La modélisation de la marche humanoïde en boucle ouverte dépend de nombreux paramètres dont les différentes combinaisons de valeurs impliquent des variations d'allure, de stabilité, d'accessibilité et de vitesse dans la marche. L'utilisation de méthodes d'apprentissage et d'optimisation [11] permet de trouver des valeurs tendant à maximiser les performances réalisées et ainsi dépasser les performances obtenues avec les connaissances d'un expert. Le réglage de la démarche

11. Ces méthodes d'apprentissage et d'optimisation sont nombreuses. Elles regroupent notamment les algorithmes génétiques, le recuit simulé, l'optimisation par colonie de fourmis, l'optimisation par essaim particulaire et les algorithmes évolutionnaires.

des déplacements a de ce fait été largement réalisé à l'aide de ces techniques. Le principe mis en œuvre est d'appliquer itérativement des modifications sur un ensemble de paramètres, conformément à une fonction d'évolution [12].

M. Hebbel *et al.* [88] ont appliqué avec succès différentes primitives à des stratégies évolutionnaires pour mettre au point une marche rapide en avant. Dans le processus de mutation/sélection de l'évolution des individus, ils proposent d'éviter les minima locaux en sélectionnant des enfants ayant des différences marquantes avec leurs parents et de développer plusieurs parents simultanément. Les expérimentations ont été réalisées sur un robot humanoïde *Kondo KHR-1* à 17 degrés de liberté (dont 5 pour les jambes et 3 pour les bras) commandés par servomoteurs. La particularité de ce robot est de ne pas pouvoir exécuter de rotation. Les mouvements possibles sont des translations avant/arrière et gauche/droite. Le processus d'évolution a été appliqué à la marche avec différentes primitives de mouvements (*i.e.* avec des profils de pas calculés selon des trajectoires en demi-cercle, rectangle et ellipse). Avec l'optimisation de 11 paramètres de marche par stratégie évolutionnaire, les résultats montrent des facteurs d'accélération de la marche de l'ordre de 3 à 4 fois plus rapide en simulation et sur un vrai robot.

C. Niehaus *et al.* [176] ont conçu une marche omnidirectionnelle à l'aide d'une méthode d'optimisation par essaim de particules. L'omnidirection est étudiée au travers de 5 sous-parties, qui par symétrie forme l'ensemble des directions possibles. Avec 14 paramètres, la marche obtenue permet des déplacements à différentes vitesses, en fonction de l'angle du déplacement. Les vitesses de déplacement en arrière, en pas latéral et en diagonale-avant sont équivalentes. La vitesse de déplacement en diagonale-arrière est plus lente et la vitesse en marche-avant est plus rapide.

P. MacAlpine et P. Stone [161] ont proposé une marche omnidirectionnelle avec une vitesse de déplacement uniforme pour toutes les directions. Cette marche a été réalisée par application d'une stratégie évolutionnaire [86]. L'évolution des paramètres de marche est guidée par évaluation de plusieurs déplacements vers des positions choisies dans différentes situations : longs déplacements en avant, en arrière et sur les côtés ; déplacements le long d'une courbe ; changements de direction rapide ; arrêts et redémarrages ; alternances entre déplacements d'un côté et déplacements du côté opposé ; déplacements vers une cible avec une position oscillante pour simuler le bruitage des données de perception ; alternances de déplacements avant et arrière avec des angles de 45 degrés ; changements brutaux de direction de déplacement ; combinaisons entre mouvements d'arrêt et déplacements ; déplacements en cercle dans le sens des aiguilles et dans le sens inverse. La distance parcourue, la durée et les chutes sont prises en compte pour définir le

12. En application, selon les ensembles de paramètres et leur valeur initiale, les résultats pourront être différents. Les marches produites seront parfois irréalistes [161], parfois réalistes [122].

taux de réussite des essais. Pour le déplacement en marche-avant, des paramètres supplémentaires permettent d'activer un mode de déplacement plus rapide, pour obtenir un *sprint*. Par optimisation séquentielle selon des politiques de déplacement (aller vers une cible fixe, aller vers une cible mobile, *sprinter*, *dribbler*), 5 ensembles de valeurs de paramètres permettent d'obtenir 5 vitesses de déplacement. Les expérimentations sont réalisées en simulation dans l'environnement de la *RoboCup 3DSSL*.

A. Farchy *et al.* [69] ont proposé une solution permettant de réduire les écarts entre optimisation en simulation et réalité. Des tests sur robot réel sont réalisés pendant le processus d'optimisation en simulation. Ce processus d'optimisation adaptatif permet de réaliser des optimisations sur des modèles légèrement différents des modèles réels. Les expérimentations en simulations utilisent la première version de *NAO* et les tests réels utilisent la dernière version de *NAO* (qui possède des jambes plus longues).

Pour améliorer plus significativement les performances individuelles des robots humanoïdes, nous avons proposé d'inclure les propriétés physiques du modèle de robot humanoïde dans le processus d'optimisation. Considérant le processus d'optimisation comme un procédé d'extraction de caractéristiques cachées dans le modèle de robot humanoïde choisi, nous avons simultanément fait évoluer les caractéristiques physiques du modèle et les paramètres des primitives de ses déplacements permettant ainsi d'établir un processus de croissance humanoïde par optimisation morphologique. Ce processus d'optimisation suit nos travaux sur la génération automatique des paramètres d'un modèle géométrique de robot humanoïde [95]. Dans le cadre de la marche humanoïde, contraints par des résultats contenant des écarts types importants dus à la simulation des lois de la physique combinée avec la complexité des problèmes d'équilibre des robots humanoïdes, nous avons choisi la méthode d'optimisation robuste CLOP [53], dont le principe est de considérer un historique des tests achevés et de ne considérer dans le processus d'optimisation que les valeurs proches de la moyenne actuelle de ces tests. Le processus d'optimisation morphologique [122] devient de fait un processus de décision dans lequel le résultat associé à chaque test devient une valeur de discrimination. La fonction décidant du résultat d'un test est appelée fonction de choix (*i.e.* fonction `pickOut`).

L'algorithme 12 présente la boucle principale d'un processus générique (*i.e.* pour une tâche $< T >$) d'optimisation morphologique constitué de n itérations. Ce processus est défini pour l'optimisation d'un ensemble de paramètres $\mathscr{L}$ contenant leurs valeurs initiales et leurs bornes de variations admissibles. Pendant le processus d'évolution, les résultats sont recueillis dans un historique $\mathscr{H}$ contenant des tuples associant valeurs testées et résultats. Chaque itération du processus d'évolution suit les étapes suivantes :

– La définition des nouveaux paramètres p à tester par combinaison des en-

Function `evolution` $< T >(n, \mathscr{L}, \texttt{pickOut})$

$(v', \mathscr{H}) \leftarrow (\emptyset, \emptyset)$;
for $i = 0$ *to* n **do**
  $p \leftarrow$ `newParams` $< T >(\mathscr{H}, \mathscr{L})$;
  $(v) \leftarrow$ `multipleTrials` $< T >(p)$;
  $(v', h) \leftarrow$ `pickOut` $< T >(v, v')$;
  `insert` $< T >((p, h), \mathscr{H})$;
end
return `paramsFrom` $< T >(v')$;

Algorithme 12 : Principe d'évolution par optimisation morphologique.

sembles $\mathscr{L}$ et $\mathscr{H}$.
- L'exécution de tests par la fonction `multipleTrials` qui permet l'obtention de nouveaux résultats v.
- La décision de la progression réalisée par la fonction de choix `pickOut` qui permet de définir la valeur v' de meilleur résultat obtenu.
- L'insertion des tests réalisés dans l'ensemble $\mathscr{H}$.

Les tests exécutés par la fonction `multipleTrials` permettent d'établir des valeurs moyennes qui comparées au meilleur résultat actuel, permet de définir une évaluation tranchée (*i.e.* un résultat meilleur, équivalent, ou moins bon que le précédent) permettant une convergence contrôlée de l'évolution.

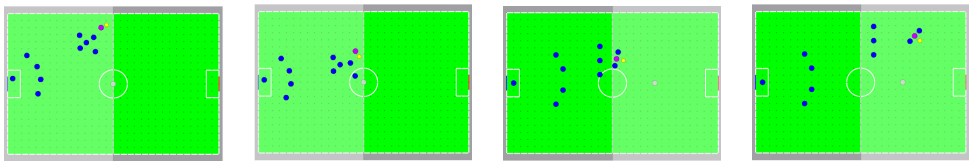
Nous avons réalisé les premières expériences de croissance humanoïde sur le robot humanoïde *NAO* en recherchant uniquement la longueur de fémur idéale. Ces premières expérimentations en simulation ayant permis de retrouver avec succès la taille originale du fémur de *NAO*, nous avons appliqué le même processus pour accélérer les déplacements en marche avant. Avec deux politiques d'optimisation, nous avons établi des marches 1.4 et 1.5 fois plus rapides que la marche originale. Ce processus a également été appliqué à des mouvements de coups de pied dans une balle [121], obtenant ainsi un gain de 1.34 en distance parcourue par la balle. Le processus d'optimisation a été comparé à une séquence de processus d'optimisation et les résultats montrent une meilleure précision par subdivision en sous-processus d'optimisation. Nous avons appliqué l'optimisation morphologique avec des fonctions de choix correspondant à des caractéristiques différentes permettant ainsi de définir 4 profils morphologiques spécialisés pour le jeu en équipe humanoïde hétérogène [119]. L'optimisation morphologique a été appliquée à des situations très différentes [125], permettant d'obtenir des vitesses de déplacement plus rapides, des mouvements de coups de pied plus puissants et plus précis, des profils de joueurs différents.

3.4 Génération de stratégies collectives

Dans le cadre des compétitions de *RoboCup SPL* et *RoboCup 3DSSL*, nous avons étudié le problème de décision dans un système distribué composé de robots humanoïdes autonomes ayant des capacités de communication limitées. Considérant les possibles déconnexions des robots, les stratégies collectives ont été conçues comme une couche additive optionnelle dans les stratégies individuelles. Nous avons présenté l'utilisation de 2 architectures distribuées, la première [100, 99, 123] mettant en avant les capacités de coopérations des robots avec des comportements individuels évolués[124] (*i.e.* contenant plus de 80 états) et la seconde [118] utilisant 2 comportements individuels simplifiés (*i.e.* réduits à une trentaine d'états) permettant plus de réactivité et des interactions contrôlées. La première a été utilisée en simulation pour des groupes de 11 robots humanoïdes *NAO* et en réalité pour des groupes de 4 robots humanoïdes *NAO*. Cette première architecture a montré des limitations dans ses possibilités de gestion des interactions, la combinatoire des états possibles des différents robots créant des problèmes de synchronisation et d'attente non compatibles avec les contraintes de décision et d'action en temps-réel. La seconde n'a été expérimentée qu'en simulation pour des groupes de 11 robots humanoïdes *NAO*. La figure 3.4 montre l'interface développée pour la spécification des stratégies collectives centrées sur des individualités fortes avec des rôles évolués. La capture d'écran dans la partie droite de la figure montre le résultat dans l'environnement de simulation *SimSpark-rcssserver3d* [178, 207]. Les joueurs ont des positions et des comportements qui varient en fonction de la position de la balle. Les comportements sont associés à des zones d'effet. En fonction des comportements, l'aire résultante sera plus ou moins importante que cette zone d'effet. Un comportement attaquant ne prendra pas en considération cette zone. Un comportement de défenseur prendra en compte la largeur de la zone pour se déplacer sur une position d'intersection. Les zones d'effet des comportements sont définies sur une grille dont le pas de discrétisation est variable. Les différentes couleurs représentent les différents joueurs. La figure 3.5 montre l'interface développée pour la spécification des stratégies collectives centrées sur des interactions avec des rôles simples. Les positions des joueurs assistants sont représentées par des points bleus, la position du premier attaquant par un point violet et la position de la balle par un point jaune. Les figures 3.5(a) à 3.5(d) montrent la progression des positions des joueurs en fonction de la balle. La spécification des stratégies en *lua* permet des modifications en temps-réel.

Les stratégies individuelles de chaque robot ont été établies par modélisation des comportements en automate hiérarchique à états finis [92] (HFSM). Nous avons étudié les solutions alternatives de spécification de comportements et les niveaux d'abstraction possibles dans les communications entre robots.

FIGURE 3.4 – Interface de spécification des stratégies collectives centrées sur les rôles individuels.



(a) (b) (c) (d)

FIGURE 3.5 – Interface de spécification des stratégies collectives centrées sur les interactions.

Le développement d'un module de communication [100, 99] nous a permis de connaître l'état courant et l'état désiré de chaque robot. Partant de ces informations, nous avons ajouté des situations collectives à la modélisation HFSM des comportements. Dans le cadre d'une épreuve de la compétition *RoboCup-SPL*, ces états collectifs ont permis de réaliser des comportements de groupe. Les participations en *RoboCup-SPL* et en *RoboCup-3DSSL* démontrent les résultats obtenus par certaines équipes sans communication et en utilisant un échange de rôle dynamique. Les stratégies sans communication, fondées sur l'utilisation d'un leader de jeu et d'un ensemble d'assistants pouvant échanger leur rôle avec le leader à tout moment, permettent d'obtenir des stratégies collectives beaucoup plus réactives. Après une première solution combinant des comportements individuels complexes avec une modélisation HFSM et des rôles correspondants à des placements définis par raisonnement par cas [123], nous avons réalisé une coordination plus réactive avec des comportements individuels simplifiés et une interface de stratégie dynamique en *lua* [118].

3.5 Conclusion

Dans ces travaux sur la robotique humanoïde, nous avons étudié les problèmes de génération de mouvements, de vision temps-réel embarquée et de génération de stratégies collectives. Nous avons présenté des travaux précurseurs en optimisation morphologique humanoïde. Du point de vue concret, nous avons manipulé et testé la plupart des algorithmes proposés sur le robot humanoïde *NAO*. Les nombreuses participations dans les compétitions de *RoboCup-SPL* et en *RoboCup-3DSSL* (*c.f.* Annexe A) ont permis de valider en pratique les idées présentées dans ces travaux.

Chapitre 4

Coordination multi-robots

Sommaire

Nous décrivons dans ce chapitre les résultats de nos travaux de recherche concernant les systèmes *Multiple Mobile Robot Systems* (MMRS) [1] appliqués au transport et à l'exploration.

1. Les systèmes MMRS sont une extension des systèmes *Multi-Robots Systems* (MRS). Les recherches sur les systèmes MRS débutent dans les années 1990, avec des travaux regroupant des robots mobiles rassembleurs d'objets [61], des colonies de robots marcheurs [49, 50]. De nombreux projets de recherche tels que CentiBOTS [142] dont l'objectif était de coordonner une équipe de plus de 100 robots mobiles pour une mission d'exploration en environnement intérieur, ont présenté une taxonomie des travaux existants. G. Dudek *et al.* [67] ont présenté une taxonomie des *Multi-Agent Systems* (MAS) selon leur capacité de communication et de calcul. Y.U. Cao *et al.* [33] classent les systèmes MRS selon 5 critères suivants : l'architecture du groupe, les potentiels conflits de ressources, l'origine de la coopération, l'apprentissage et la géométrie. P. Stone et M. Veloso [211] ont proposé 4 scénarios type, associés à l'utilisation de différents types d'agents dans un problème de poursuite : des agents homogènes non communicants, des agents hétérogènes non communicants, des agents homogènes communicants et des agents hétérogènes communicants. Leurs travaux posent principalement la question de l'apprentissage des systèmes MRS. T. Arai *et al.* [10] ont identifié 7 thèmes de recherche des systèmes MRS : les inspirations biologiques, la communication, les architectures, la localisation / la cartographie / l'exploration, le transport et la manipulation d'objet, la coordination de mouvement et les robots reconfigurables.

L'utilisation de plusieurs robots est justifiée par l'obtention d'un meilleur résultat d'exécution comparativement à une mission impliquant un unique robot. Ce gain est fondé sur la synthèse d'une bonne coordination entre les robots. Il peut se traduire par un temps total inférieur [30], par une consommation globale inférieure [227], par une meilleure robustesse [71, 162, 192] ou encore par un coût inférieur. Sans coordination, l'utilisation d'un système MMRS peut donner des résultats équivalents, plus mauvais, voire conduire à l'échec de la mission dans des cas extrêmes de mauvaise coordination menant à des situation bloquantes. La coordination étant liée à la répartition des tâches correspondant à l'ensemble de la mission, il s'agit de limiter les intersections entre les sous-espaces d'action des différents robots impliqués. Il existe 2 types de mécanismes de répartition de tâches, le premier étant centralisé et le second décentralisé. L'avantage d'un mécanisme centralisé est de pouvoir prétendre à la réalisation d'un plan optimal. Le désavantage d'un tel mécanisme est l'augmentation combinatoire du problème conséquent à l'augmentation du nombre de robots impliqués. Comme son nom le suggère, l'utilisation d'un mécanisme décentralisé implique l'absence de planificateur central. Chaque robot d'un système MMRS utilise des informations locales impactant son propre plan d'exécution. L'avantage des mécanismes décentralisés est d'avoir un bon potentiel d'adaptabilité et une grande robustesse. Le désavantage des mécanismes décentralisés est de proposer des solutions souvent sous-optimales. Indépendamment de ces mécanismes, la coordination peut être réfléchie à l'aide d'un plan (dont les 2 niveaux conceptuels courants sont la planification des tâches [2] et la planification de mouvement [3]).

La décomposition d'une mission en sous-parties et sa répartition dans un système MRS sont équivalentes à un problème du type boîte-poussant multi-robots, dans lequel les robots partagent des actions de transport, de pousser ou de manipulation.

M.J. Matarić *et al.* [166] ont étudié le problème de la coopération avec 2 robots autonomes à 6 pattes [4]. Chaque robot communique avec les autres robots afin de

---

La mobilité étant un caractère important, les systèmes composés de plusieurs robots mobiles sont abrégés MMRS. Avec la notion de mobilité, les robots des systèmes MMRS acquièrent une plus grande autonomie de pensée et d'action. Ils peuvent se déplacer pour réaliser des actions et participer à des actions collectives.

2. Pour une pensée de haut niveau, très abstraite, par action.

3. Pour une pensée de bas niveau, très concrète, proche des consignes de commandes à appliquer à chaque actionneur sur chaque robot.

4. Les robots à 6 pattes sont des hexapodes. Ces robots sont plus stables que les robots bipèdes du fait qu'ils soient statiquement stables dans la plupart des positions. Leur équilibre en marche à petite allure ne dépend pas de contrôleurs temps-réel et d'un calcul d'équilibre. A contrario à grande vitesse de déplacement, les insectes sont dépendants de facteurs dynamiques. Dans l'absolu, la complexité de la morphologie d'une patte est sans limite. En pratique, les pattes sont construites selon un modèle insecte avec 2 à 6 degrés de liberté (créant ainsi de 12 à 36 degrés de

coordonner la livraison de chaque boîte.

C.R. Kube et E. Bonabeau [146] ont présenté des observations empiriques de transport coopératif chez les fourmis et décrivent une étude de cas des problèmes du type boîte-poussant par un essaim de robots [5].

B.P. Gerkey et M.J. Matarić [80] ont proposé un système de répartition des tâches basé sur la notion d'enchères appelé MURDOCH, pour la résolution des problèmes du type boîte-poussant. La solution proposée répartit et coordonne les tâches de manipulation dans un environnement tolérant aux fautes. Les résultats montrent une coopération étroitement couplée avec le nombre de robots.

N. Miyata *et al.* [169] ont proposé une méthode d'affectation des tâches pour le transport coopératif multi-robots dans un environnement inconnu statique en divisant les actions de chaque robot en tâches élémentaires. Ces tâches élémentaires sont affectées par coïncidence entre le nombre de robots disponibles et la nature des tâches à réaliser. La distribution des tâches dans le temps et dans l'espace tend à minimiser les conflits. Cette distribution est réalisée sous hypothèse d'équi-répartition des tâches dans le temps et dans l'espace.

Z.D. Wang *et al.* [226] ont proposé pour le transport avec les systèmes MMRS un algorithme de test rapide permettant de vérifier rapidement les possiblités de saisie d'un objet fixé [6]. Les algorithmes proposés permettent de maintenir une formation composée de 2 robots contribuant au transport d'un objet commun.

Le problème du partage de l'espace dans les systèmes MMRS a été étudié principalement dans les problèmes de planification de mouvement, d'évitement de collision, de congestion et d'interblocage.

M. Jäger et B. Nebel [104] ont décrit une méthode décentralisée pour coordonner des trajectoires indépendantes. Ces travaux permettent d'éviter les collisions entre les robots mobiles tout en évitant l'apparition d'impasses dans les chemins des robots. Les impasses possibles sont évaluées en déplacement standard comme en manœuvre avec des points de rebroussement. Les chemins en commun sont évalués par le biais des intersections entre les chemins des robots.

L'idée présentée dans leurs travaux est la suivante : chaque fois que la distance entre 2 robots est inférieure à une valeur fixée, chaque robot transmet les informations concernant sa trajectoire planifiée et évalue les collisions possibles. Si une collision possible est détectée, alors ils surveillent leurs mouvements et insèrent si nécessaire des temps d'inactivité entre des portions de leurs trajectoires afin d'éviter ces collisions. Chaque fois qu'une impasse est détectée, les planificateurs locaux de chacun des robots impliqués génèrent successivement une trajectoire

liberté pour un robot mobile à 6 pattes).

5. Les premières études sur les comportements collectifs commencent avec les travaux de Deneubourg *et al.* [60].

6. Pour un objet fixé, ce test est défini par la distance à partir de laquelle toute configuration admet un chemin se rapprochant de cet objet et permettant de s'en saisir.

alternative jusqu'à ce que l'impasse disparaisse.

R.C. Luo *et al.* [159] ont proposé une approche de réseau de neurones pour la planification de trajectoire multi-robots. Dédiées à des tâches de nettoyage, les trajectoires des robots sont générées en temps réel en utilisant un réseau de neurones.

L.S. Marcolino et L. Chaimowicz [163] ont proposé un algorithme de coordination décentralisé pour contrôler le trafic d'un essaim de robots, en évitant les situations de congestion lorsque les grands groupes de robots se déplacent en sens opposé. L'algorithme proposé permet aux robots de percevoir la possibilité d'une collision et d'avertir leurs coéquipiers grâce à des communications locales. Les groupes concernés modifient ainsi leur trajectoire pour éviter les congestions. Ils ont également proposé un algorithme de coordination [164] pour le contrôle de trafic quand les robots ont pour objectif local d'atteindre une même cible. Dans cette solution, les robots contrôlent leurs actions en utilisant une machine à états finis probabiliste, en comptant sur les mécanismes de détection de collision et de communication locale pour la coordination.

R. Luna et K.E. Bekris [158] ont proposé un algorithme de planification de trajectoire multi-robots dans un graphe topologique comportant à chaque instant au moins 2 sommets non alloués. L'algorithme proposé utilise 2 primitives : l'une nommée "*push*", dans laquelle un robot se déplace vers son objectif jusqu'à ce qu'aucun progrès ne puisse être fait et l'autre nommée "*swap*", qui permet à 2 robots d'échanger leurs positions, sans altérer la position de tout autre robot.

Le problème du partage de l'espace est un problème important en planification de mouvement. Ce problème apparaît quand plusieurs robots se déplacent vers un même *point de passage* de l'espace de travail, provoquant ainsi une congestion et par là-même une collision de l'espace et un éventuel interblocage, qui peuvent compromettre les performances globales du système. Ce problème fait apparaître une pause assimilable à un état dans lequel tout ou partie du système (*i.e.* un ou plusieurs robots) s'arrête par considération d'un ou plusieurs points de passage en exclusion mutuelle. Cet état de pause ou état d'attente est appelé *situation d'attente*. Il s'agit selon les cas, d'attendre le passage d'un autre robot et dans l'hypothèse d'une congestion excessive, d'attendre une nouvelle planification complète de l'ensemble des trajectoires des robots. Cette hypothèse de congestion pose le caractère déterminant des *situations d'attente* dans le temps global d'exécution d'une mission multi-robots.

4.1 Planification de tâches

Dans les missions du concours *AAAI Robot Challenge* caractéristiques de la robotique, l'objectif est de faire participer un robot à une conférence. Participer

signifie ici s'enregistrer et aider à l'enregistrement des conférenciers au guichet d'accueil, transporter des objets utiles pour les conférenciers, délivrer des messages, surveiller des salles, informer les conférenciers et donner une conférence sur son rôle et ses fonctions. Par analogie à un découpage de ce type en tâches, une mission multi-robots se découpe en sous-tâches, dont l'objectif est de répartir les tâches entre les différents robots à chaque instant. Cette répartition est réalisable par échange de messages ou sans communication par évaluation heuristique.

4.1.1 Par échange de messages

Les approches classiques de répartition des tâches dans un système MRS décentralisé sont principalement basées :

- Sur la notion de marché [210].
- Sur la notion d'enchère [81].

Ces notions sont liées à des mécanismes d'enchère et de soumission. Contrairement à ces approches, nous avons proposé d'utiliser des mécanismes de demande et de répartition. Nous avons proposé un mécanisme, dit d'offre non sollicitée, dont l'articulation est définie par un robot, appelé acheteur, qui réalise une demande de répartition des tâches à un autre robot, appelé vendeur. Après un intervalle de temps fixé, le vendeur analyse les demandes reçues et affecte les tâches aux acheteurs selon un ensemble de règles. Ce mécanisme permet l'assignation de plusieurs tâches à plusieurs robots simultanément. Ces travaux ont été expérimentés sur des problèmes d'exploration multi-robots, dans des environnements structurés inconnus à l'aide d'une équipe de robots mobiles homogènes. Pour répondre à la problématique classique des situations de sauvetage de personnes en environnements dangereux tels que les bâtiments faisant face à des incendies ou des explosions, ou encore dans des situations post-cataclysmiques [105, 13], les résultats montrent des temps d'exécution inférieurs.

Concernant les questions de logistique multi-robots, les problèmes de fourragement [7] sont une instance classique [230] de test des systèmes distribués multi-robots. Ce type de résolution a pour objectif de prouver l'intérêt des approches

7. Le fourragement est défini comme l'activité de collecte de ressources et de stockage de ces ressources dans une base [249]. Pour les colonies de fourmis, il s'agit d'optimiser la recherche et la collecte de nourriture. Par similitude aux comportements de ces colonies, M. Dorigo [64] a démocratisé le développement d'heuristiques permettant de réduire ou d'optimiser des caractéristiques comme la longueur des chemins parcourus, correspondant au comportement global de ces systèmes multi-agents ou multi-robots. Cette approche transposée dans de nombreux problèmes d'optimisation est inspirée simultanément de la création de piste par dépôt de phéromones et d'heuristiques de comportements de groupe. En pratique, il s'agit d'articuler un algorithme de parcours global de l'espace du problème avec une heuristique de recherche locale. Les principales variantes des algorithmes ACO et les principales méthodes alternatives combinant recherche locale avec recherche globale sont presentées dans [79].

impliquant plusieurs agents dans la réalisation d'une tâche distribuée de collecte de nourriture, divisible en plusieurs tâches dans un système multi-robots et par ajout de mécanismes primitifs [8] de coopération [24, 64]. La recherche de nourriture est distribuée entre les différents acteurs, réalisée sans connaissance a priori des sites d'apparition des denrées à collecter et exécutée dans un espace inconnu. L'utilisation de phéromones implique la création de minima locaux [181] produisant des solutions parfois sous-optimales avec des mécanismes simples [9] ayant pour avantage de nombreuses possibilités de transposition dans la réalité [206]. La coopération entre les différents agents est assurée par un algorithme c-marking. La phéronome physique est remplaçée par des pions ce qui simplifie l'expression de la coopération et permet de se focaliser sur l'achèvement rapide d'une recherche de denrée coordonnée [18]. Par application d'un modèle de phéromone intégrant propagation par contact avec les agents et évaporation dans le temps, les simulations réalisées permettent d'obtenir une meilleure coopération dans l'activité de fourragement [248, 246].

G. Pini *et al.* [186] ont proposé une approche de partitionnement de tâches par essaim multi-robots dans le cadre de l'activité de fourragement. Le découpage en sous-tâches et la subdivision des coûts des sous-tâches est réalisée par combinaison d'une fonction de coût et d'une cartographie des sous-tâches. Par isolement des actions spécifiques et recherche des actions par agent dans des rayons d'action limités, les erreurs de positionnement sont limitées et le contrôle de la répartition est optimisé pour minimiser la distance globale parcourue par l'ensemble des robots. Les résultats ont été obtenus avec le simulateur ARGoS [185] en impliquant de 4 à 15 robots foot-bot du projet Swarmanoid [65]. Les résultats mettent en avant la généricité de l'approche dépendant d'une fonction de coût transposable à d'autres problèmes et montrent leur dépendance avec l'odométrie, le nombre de robots et la taille globale de l'environnement.

4.1.2 Par évaluation heuristique

Avec une évaluation heuristique, le problème du transport de marchandises multi-robots d'un lieu de production vers un lieu de consommation devient un problème de minimisation d'un ou plusieurs critères tels que le temps, la consommation d'énergie ou la distance parcourue. Pour diminuer les conflits et les possibles interférences entre les robots, diverses simplifications peuvent être appli-

8. L'intelligence en essaim, initialement définie par G. Beni et J. Wang [20], est une propriété présente dans un système multi-robots et absente dans chacun des robots constituant l'essaim. La définition initiale oppose ainsi collectivité pensante et individualité agissante.

9. Parmis les mécanismes constitutifs de l'intelligence collective des colonies d'insectes tels que les blattes et les fourmies, M. Travers [220] insiste sur les relations étroites entre physiologie et comportement.

quées : unicité des sources en fonction des ressources, unicité des destinations (simplifiant les questions d'optimisation de consommation ou de stockage), emplacements connus et fixés. Dans un souci de réalisme vis à vis de robots transporteurs, les taux de production des marchandises restent inconnus et la capacité des sources est limitée à une unité de ressources. Tous les robots sont placés dans un dépôt au début de la mission et retournent à ce dépôt dès qu'ils sont libres (*i*.e. sans ressource). La consommation d'un robot est supposée nulle dans la zone de dépôt. En dehors de cette zone, la consommation d'énergie est proportionnelle aux perceptions (par estimation de l'énergie nécessaire à l'alimentation des capteurs et de la consommation associée aux traitements des perceptions) et aux déplacements des robots (par estimation de l'énergie nécessaire à la commande des actionneurs et de la consommation associée aux calculs de trajectoire). Par souci de simplification et d'abstraction des modèles de consommation, le taux de cette consommation doit rester proportionnel au temps de déplacement des robots. Pour la réalisation d'une tâche de transport d'une durée de 10 secondes avec un robot, 10 unités d'énergie sont consommées. L'objectif de tels problèmes est d'achever la mission de transport par exploitation du potentiel de collaboration multi-robots en réduisant le temps d'exécution et en limitant la consommation d'énergie à un niveau préalablement fixé.

Dans le cadre du projet *Multiple Autonomous Robots for Transport and Handling Applications* (MARTHA), R. Alami *et al.* [3] ont présenté une solution pour le contrôle d'une flotte de robots mobiles autonomes en utilisant un paradigme de plan-fusion pour la coopération multi-robots au travers d'une solution distribuée, incrémentale dans laquelle les robots mobiles utilisent des informations locales. Chaque robot planifie indépendemment ses actions en combinant 3 plans (1 plan topologique, 1 plan géométrique et 1 plan multi-robots qui spécifie des positions de synchronisation de plan avec les autres robots). Cette approche a été testée en simulation avec une flotte de 10 robots dans des environnements de chargement/déchargement de marchandises et en réalité avec 3 plateformes mobiles de la famille Hilare dans un environnement intérieur [10]. La décomposition des plans permet de réduire les appels au planificateur central définissant les positions de synchronisation. Les résultats montrent l'adéquation d'une approche générique pour la gestion d'une flotte de robots.

R.T. Vaughan *et al.* [224] ont proposé la méthode *Localization-Space Trails* (LOST) permettant une navigation multi-robots utilisant des points d'intérêt dans un environnement inconnu. Les points d'intérêt sont fictifs et sont générés individuellement par les robots par référence à des spécificités de l'environnement, pour être par la suite partagés entre tous les robots dans un repère globale. Cette

10. Les rapports du projet précisent 2 salles pour une surface totale de $70m^2$ contenant lignes, croisements et stations de recharge spécifiées dans une carte topologique.

méthode a été appliquée à un exemple de tâche de transport de ressources, dans lequel plusieurs robots autonomes trouvent et traversent à plusieurs reprises un chemin. Ce chemin est réalisé dans un environnement inconnu entre un domicile connu et une zone de production de ressources dans une position initialement inconnue. Les expérimentations ont été réalisées avec 4 robots *Pionneer 2DX* [11] dans un espace accessible aux utilisateurs [12] contenant salles et couloirs (permettant à 2 robots de passer simultanément). Les résultats montrent les possibilités de convergence du plan multi-robots avec des divergences dans les systèmes de coordonnées des différents robots.

F. Tang et L.E. Parker [215] ont proposé l'approche *Automated Synthesis of Multi-Robot Task Solutions through Software Reconfiguration* (ASyMTRe) pour synthétiser automatiquement des solutions de planification de tâches pour les équipes hétérogènes. Cette approche permet à l'équipe de robots de connecter dynamiquement des schémas d'exécution de différents robots et permet ainsi d'accomplir des tâches distinctes dans un environnement partagé. Cette approche a été expérimentée dans un scénario de transport multi-robots et dans un scénario du type boîte-poussant multi-robots. Les expérimentations réalisées en simulation sur des flottes de 2 à 100 robots et en réel sur des flottes de 2 à 7 robots (dont 6 différents). Les résultats montrent des temps de décision inférieurs à 1[*sec*].

T.S. Dahl *et al.* [57] ont proposé un algorithme pour la répartition des tâches dans des groupes de robots homogènes basé sur les chaînes de vacance. Cet algorithme a été expérimenté en simulation et a été démontré plus performant que les algorithmes de répartition de tâches lorsque les robots individuels sont sujets à des distractions, des pannes ou des changements de priorité dans les tâches pendant leur exécution. T.S. Dahl *et al.* ont défini la notion de problème de transport prioritaire (*Prioritized Transportation Problem*) comme une extension du problème de transport classique dans lequel les sources, les destinations et les objectifs du transport sont divisés en ensembles de tâches de priorités différentes (couramment appelés circuits).

Une des principales raisons de la faible efficacité des transports dans un espace connu est le manque de connaissance concernant le taux de production des ressources dans cet espace. Une estimation régulière de ce taux permet de résoudre ce problème, pour lequel une fonction heuristique donne une solution efficace [101, 179, 182]. Le taux de production, noté T_n, reflète le temps écoulé entre la disparition de la $n^{\text{ème}}$ marchandise [13] et l'apparition de la $(n+1)^{\text{ème}}$ marchandise.

11. Robot mobile de 50x50[*cm*] avec différentes variantes : ceinture de 16 télémètres ultrasons, nappe laser, caméra, bumpers, pince, carte ethernet et carte son.

12. Le rapport précise des expérimentations réalisées en fin de soirée en impliquant des utilisateurs pour simuler l'activité normale d'une agence.

13. Une marchandise disparaît quand elle est emportée. Cette considération est uniquement virtuelle dans la mesure où les marchandises ne disparaissent pas effectivement et sont en cours

T_n est considéré comme une suite de variables aléatoires indépendantes et identiquement distribuées. La loi de probabilité des variables est inconnue, mais peut être déterminée par le test d'hypothèse. La valeur heuristique $h(n)$ est donc obtenue par la fonction de distribution suivante :

$$h(n) = E(T_{n+1}^{'}|T_1,\ldots,T_n) \tag{4.1}$$

L'équation (4.1) définit le temps écoulé depuis la disparition de la $n^{\text{ème}}$ marchandise jusqu'à l'apparition de la $(n+1)^{\text{ème}}$ marchandise par notre prévision. Étant donné T_n une distribution uniforme :

$$h(n) = \frac{1}{n}\sum_{i=1}^{n} T_i \tag{4.2}$$

En effet, l'ensemble du problème est une prise de décision limitée. Pour chaque robot $R_{i(i=0,\ldots,m)}$, l'espace de décision pour chaque étape (également notée R_i) est fini. Par exemple :

$$R_i = \{gotoSource, gotoSink, gotoDepot\} \tag{4.3}$$

Ainsi, l'espace de décision pour chaque étape du planificateur centralisé est défini par :

$$R = \prod_{i=0}^{m} R_i \tag{4.4}$$

Pour N marchandises, l'espace de décision du planificateur centralisé est R^N. Chaque décision $r \in R^N$ est définie par $r(n) \in R(n = 1,\ldots,N)$.

L'équation suivante définit le temps d'exécution noté $X(n)$, pour identifier le temps d'exécution de la mission après achèvement du transport de la $n^{\text{ème}}$ marchandise, par :

$$X(n+1) = X(n) + \min_{t\in R} f(t, h(n), r_n) \tag{4.5}$$

Dans l'équation (4.5), f est une fonction permettant de calculer l'intervalle de temps entre le début des tâches de transport de la $n^{\text{ème}}$ et de la $(n+1)^{\text{ème}}$ marchandise. t représente la prise de décision actuelle. Si f est égale à sa valeur minimum (valeur dépendant de la mission), alors t est égale à $r(n)$. r_n représente l'information de localisation actuelle de chaque robot. Par exemple, t étant également une décision, si la décision prise par un robot est de sortir du dépôt pour se rendre à l'emplacement de l'apparition d'une ressource, alors :

---

de transport.

$$f(t,h(n),r_n) = \max(h(n), T_{DepotToSource}) \tag{4.6}$$

L'équation (4.5) utilise un algorithme glouton, permettant d'obtenir une durée totale minimum pour accomplir la mission, sans pour autant garantir un faible niveau de la consommation d'énergie. Dans le cas de la figure 4.1, `Robot0` et `Robot1` sont 2 robots libres (*i.e.* sans tâche). La mission actuelle est le transport d'une marchandise apparue à la zone `Source`. Les segments rouges et les segments bleus représentent 2 plans (le premier nommé plan rouge et le second plan bleu). Chaque plan (rouge et bleu) est composé d'un mouvement pour chaque robot (soit un total de 2 mouvements par plan). Les 2 plans sont concurrents, dans le sens où il s'agit ici de choisir entre le plan rouge et le plan bleu pour accomplir la mission. Le plan rouge envoie `Robot0` au `Dépôt` et `Robot1` à la `Source`. Le plan bleu envoie `Robot0` à la `Source` et `Robot1` au `Dépôt`. En utilisant l'équation (4.5), le planificateur choisit le plan rouge, en raison de la proximité entre `Robot1` et la `Source`. Pourtant l'évaluation des consommations d'énergie des plans rouge et bleu sont respectivement de 13.0 et de 10.4 (ce qui suggère le choix du plan bleu).

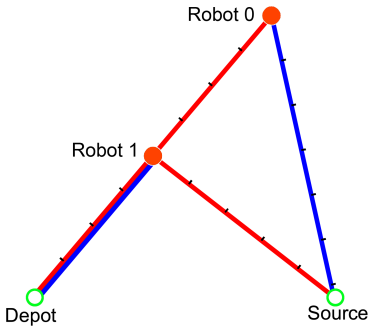


FIGURE 4.1 – Comparaison de 2 plans de répartition des tâches.

Ainsi, l'équation (4.5) ne garantit pas l'obtention d'une solution optimale. Pour pallier ce problème, nous avons proposé l'utilisation d'une évaluation heuristique basée sur un modèle empirique de la consommation d'énergie permettant de remplacer la fonction $\min_{t \in R} f(t,h(n),r_n)$. Ce modèle empirique prend en compte le temps d'exécution des tâches composant la mission. La répartition des tâches découlant de ce modèle est présentée dans l'algorithme 13.

Dans l'algorithme 13, la fonction *h(i)* réalise un test d'appartenance à un intervalle, qui s'ajuste itérativement. Cet ajustement est fonction des caractéristiques des robots mobiles et également des caractéristiques de l'environnement de travail choisi. $T_{toSource}(j)$ indique le temps nécessaire pour un robot j pour se rendre de son emplacement actuel à l'emplacement de la source. Dans l'algorithme 13,

```
Function heuristicTaskAllocation(i)

if ressource (i) == carriedAway then
    allocated (i) ← 0 ;
    update ( h(i) ) ;
    foreach position ( j ) ≠ atDepot and action (j) == free do
        if h(i) < T_toSource(j) and h(i) > 1/2 T_toSource(j) then
            allocated (i) ← 1 ;
            action (j) ← goToSource ;
            break ;
        end
    end
end
if allocated (i) == 0 then
    foreach position ( j ) ≠ atDepot do
        T_wait ← max(0, h(n) − T_DepotToSource(j)) ;
        action (j) ← wait (T_wait) + carry (i) ;
        break ;
    end
end
return ;
```

Algorithme 13 : Répartition heuristique de la ressource i sur le robot j.

$T_{toSource}(j)$ et $T_{DepotToSource}(j)$ sont 2 valeurs estimées pendant l'exécution de la mission. $T_{DepotToSource}$ est une valeur estimée une seule fois, inconnue avant l'exécution de la mission, fixée à la suite de la première livraison et utilisée par la suite dans le calcul de $T_{toSource}(j)$ et $T_{DepotToSource}(j)$. Avant la première livraison, aucune décision n'est nécessaire, ces valeurs ne sont pas utilisées. A la suite de la première livraison, elles sont fixées aux temps constatés lors de cette première itération. L'achèvement de la première tâche fixe les premières valeurs. L'achèvement de tâches supplémentaires met à jour ces valeurs. L'achèvement d'une tâche pour un robot est considéré à la livraison de chaque marchandise dans la zone destination. Une tâche peut commencer par une position au dépôt où une position en cours de retour au dépôt. Après exécution d'une tâche pour le robot j, $T_{toSource}(j)$ est égal au temps nécessaire à ce robot pour rejoindre la source des marchandises. Ce temps peut varier selon les caractéristiques des robots. Cette nouvelle formulation [237] du problème de l'allocation de ressources pour un système décentralisé permet de coordonner plusieurs robots en évaluant heuristiquement les relations entre robots et ressources au fil de l'évolution du système multi-robots et de l'ap-

parition des ressources.

4.2 Planification de mouvement

Dans les problèmes d'exploration et de transport multi-robots, il s'agit principalement de minimiser les interférences entre les différents robots impliqués. Pour résoudre ce problème, l'échantillonnage de l'environnement permet d'identifier des configurations libres proches de *points clés*, identifiés à l'aide d'un *diagramme de Voronoï* et considérés comme des points de passage pour les robots. Par identification de chemin libre entre ces configurations, il s'agit de construire un ensemble de chemins minimisant les interférences entre les robots. Dans cet ensemble de chemins, il reste possible pour 2 robots d'utiliser un ou plusieurs points de passage à des intervalles de temps différents. Dans le cas où tous les passages permettraient à 2 robots de se déplacer simultanément, il est acceptable d'avoir des paires de chemins identiques. Le nombre d'interférences entre les chemins est dépendant de la taille de l'espace des configurations, du nombre de points de passage identifiés, des caractéristiques [14] des robots et du nombre de robots disponibles à chaque instant. Considérant les limitations des solutions par fusion des informations et par segmentation de l'environnement [218, 232] contraintes dans des espaces de petite dimension (*i.e.* 2 ou $2,5$D), les solutions probabilistes sont admises. Elles peuvent être réalisées par échantillonnage dédié à la décision (guidé par les questions de collision, de congestion et d'interblocage) [240, 238, 128, 127] ou par adaptation à l'encombrement de l'espace des configurations [236, 242, 239, 241].

4.2.1 Par échantillonnage dédié à la décision

P. Švestka et M.H. Overmars [214] ont présenté l'approche *Probabilist Road-Map* (PRM) pour la planification de mouvement de robots mobiles non-holonomes dans un espace de travail partagé. Les chemins des robots sont établis par combinaison de chemins dans un graphe des configurations précalculé pour chaque robot. Ces travaux introduisent la notion de *super-graphes* pour la planification de chemin multi-robots. La majeure partie de l'exécution de cette solution réside : pour la partie hors-ligne, dans le calcul du *super-graphe* des configurations libres ; dans la partie en-ligne, dans la recherche des chemins dans ce *super-graphe*. La complétude de cette approche est assurée par l'hypothèse de convergence de la solution dans le temps [15].

14. Principalement vitesse, accélération, commandabilité et encombrement.
15. Avec un temps d'exécution infini, la probabilité de trouver une solution tend vers 1.

M. Moors *et al.* [170] ont présenté un algorithme basé sur la théorie des graphes pour la planification de mouvement multi-robots dans des environnements intérieurs $2D$. Au travers de l'étude de cas pratiques de surveillance [16] en environnement intérieur, ces travaux proposent la génération d'un plan de mouvement coordonné à l'aide de l'algorithme de recherche A* [87], en considérant les limitations et les incertitudes des capteurs. Par modélisation réaliste des capteurs et formulation d'hypothèses pessimistes d'exécution sur les déplacements de l'intrus, ces travaux ont établi une comparaison entre les différentes approches afin d'évaluer les performances de coordination. L'utilisation de l'algorithme A* nécessite une simplification de l'environnement, obtenue par discrétisation de l'espace (sous forme de pavage). Cette simplification, dont le pas de discrétisation influe sur les résultats [17], conduit à un compromis entre capacité de résolution et optimalité de la solution recherchée.

C.M. Clark [52] a présenté une stratégie de planification de mouvement multi-robots par extension de la méthode PRM. Le réseau établi par communication entre les robots permet la coordination entre les différents robots. Ce réseau nommé *Dynamic Robot Network* (DRN) est dit *ad hoc* [18]. La méthode PRM est définie par un planificateur central en tenant compte des requêtes en cours et des réponses précédentes. L'empilement des réponses permet d'augmenter la couverture globale de l'espace des configurations. La génération de jalons est utilisée pour réduire le temps de formulation d'une réponse de planification. L'utilisation d'une séquence particulière pour les configurations proches de l'objectif permet d'augmenter la convergence des solutions proposées vers la configuration finale. Les résultats montrent une réelle efficacité dans les environnements structurés composés d'obstacles statiques. Point faible et point fort de cette approche demeurent dans la nature probabiliste de la méthode PRM, dont les 2 principaux points forts (en contrepartie) sont un petit temps de réponse et une faible dépendance en la dimension du problème [19] et dont le point faible est la convergence probabiliste des calculs vers une solution.

M. Saha et P. Isto [196] ont présenté une stratégie pour la planification de mouvement multi-robots découplée. Cette stratégie propose de fusionner les 2 étapes d'une méthode de planification existante [150] pour améliorer sa fiabilité. La pre-

16. L'activité de surveillance a pour objectif principal la détection d'un robot intrus dans un périmètre prédéfini.

17. Un pas de discrétisation trop grand pourra occulter des solutions et un pas de discrétisation trop petit pourra conduire à une explosion combinatoire, rendant le calcul trop couteux en temps ou en espace.

18. Les réseaux ad hoc sont des réseaux sans fil capables de s'organiser sans infrastructure définie préalablement.

19. Regroupant principalement la dimension de l'espace des configurations, l'espace de travail considéré, le modèle cinématique ou dynamique et le nombre de robots.

mière étape calcule un chemin sans collision par robot. Chaque chemin est établi indépendamment des autres chemins, en ne considérant qu'un robot dans l'environnement. La deuxième étape définit le suivi de chaque chemin pour chaque robot en plaçant des situations d'attentes à chaque croisement avec un autre chemin. Ces interférences entre chemins impliquent des priorités entre robots et des espaces de coordination. En utilisant des priorités fixes et des politiques de gestion des priorités, ces travaux ont comparé une approche centralisée avec une approche décentralisée. L'approche centralisée donne de meilleurs résultats.

Les situations d'attente incluses dans les plans multi-robots sont l'une des raisons les plus importantes limitant les résultats des méthodes de planification de mouvement multi-robots. Ces situations d'attente sont divisées en 3 catégories :

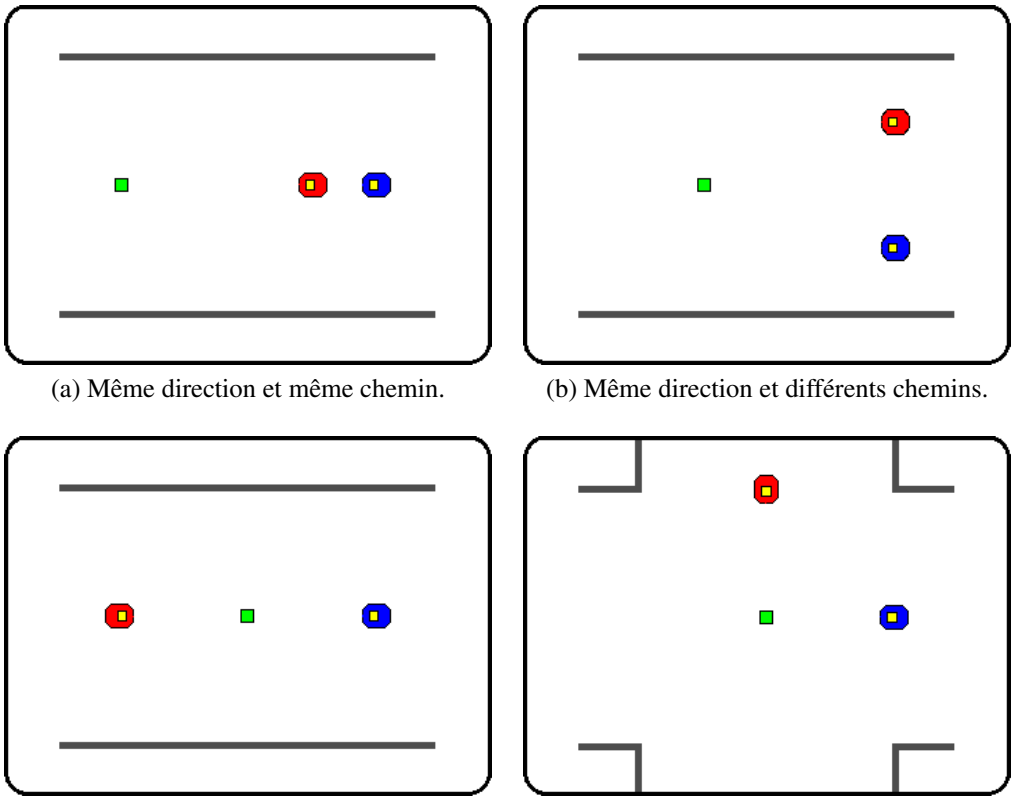
- Les collisions : quand 2 robots possèdent 2 chemins avec un point de passage commun à un même instant t ;
- Les congestions : quand une zone de l'espace devient trop petite pour permettre les mouvements des robots à un instant t ;
- Les interblocages : quand 2 chemins avec des directions opposées partagent un même espace à un instant t ;

L'apparition des situations d'attente est équivalente à l'ajout de contraintes d'exclusion mutuelle au niveau des interférences entre chemins. Ces ajouts se font soit par découpage des chemins sur des intervalles de temps soit lors de l'exécution des différents plans des différents robots, ce qui implique, dans certains cas, l'absence de solution. En conséquence de la génération initiale de chemins, il s'agit dans ces cas de savoir s'il faut composer dans ces chemins d'autres situations d'attente ou s'il faut revoir les chemins contenant des interférences.

Les exclusions mutuelles sont divisées en 4 catégories, comme présenté dans la figure 4.2. Lorsque 2 robots se déplacent vers un même point de passage à un instant t :

- Dans la figure 4.2(a), le robot rouge et le robot bleu se déplacent de la droite vers la gauche séquentiellement (*i.e.* évoluent dans la même direction en suivant un même chemin) ;
- Dans la figure 4.2(b), le robot rouge et le robot bleu se déplacent de la droite vers la gauche simultanément (*i.e.* évoluent dans la même direction en suivant des chemins différents) ;
- Dans la figure 4.2(c), le robot rouge se déplace de la gauche vers la droite et le robot bleu se déplace de la droite vers la gauche (*i.e.* évoluent dans des directions différentes en suivant un même chemin) ;
- Dans la figure 4.2(d), le robot rouge se déplace du haut vers le bas et le robot bleu se déplace de la droite vers la gauche (*i.e.* évoluent dans des directions différentes en suivant des chemins différents).

Deux solutions sont admises pour la gestion des exclusions mutuelles aux points de passage :



(a) Même direction et même chemin.

(b) Même direction et différents chemins.

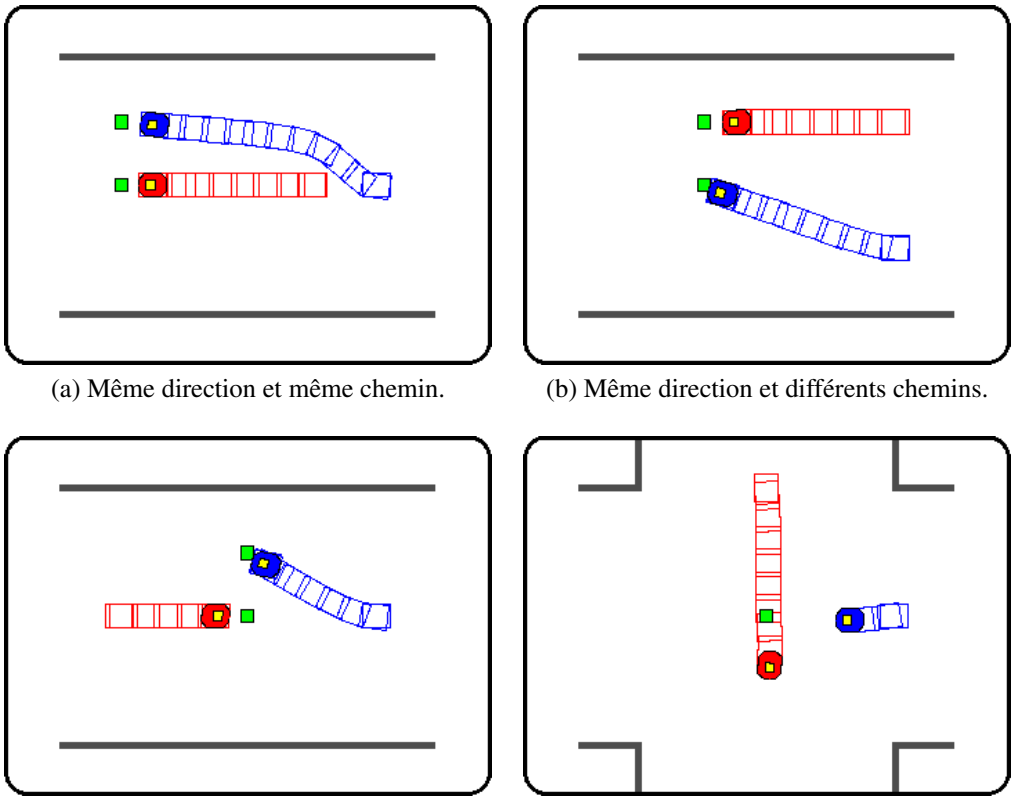
(c) Différentes directions et même chemin.

(d) Différentes directions et différents chemins.

FIGURE 4.2 – 4 catégories d'exclusion mutuelle sur interférence de chemins. Le cube vert de petite taille représente le point de passage. Les robots mobiles sont vus de haut, l'un est rouge et l'autre est bleu.

- La solution *One By One* (OBO) consiste à laisser les robots passer le point de passage un par un [214]. En tendance, cette solution prolonge le temps d'achèvement des missions.
- Les solutions de re-planification locale des portions de chemins pour chacun des robots concernés en combinant planificateur local et mécanisme d'évitement d'obstacles, à l'aide de *Vector Field Histogram Plus* (VFH+) [222] ou de *Nearness Diagram* (ND) [168]. En tendance, la phase de re-planification peut impliquer des solutions locales contenant détours dans les chemains et attentes, plus longues que des solutions globales.

Ces 3 solutions (OBO, VFH+ et ND) prolongent le temps d'exécution des missions. En augmentant le nombre de points de passage, la solution *Alternative Waypoint Generating* (AWG) [127] présente une phase initiale de planification



(a) Même direction et même chemin.

(b) Même direction et différents chemins.

(c) Différentes directions et même chemin.

(d) Différentes directions et différents chemins.

FIGURE 4.3 – Trajectoires de robots évitant les situations d'attente

avec des chemins contenant moins d'interférences. Les expérimentations réalisées en simulation avec Player/Stage dans des environnements $2,5D$ ont démontré de meilleurs résultats que les solutions OBO, VFH+ et ND. La figure 4.3 illustre les résultats de la solution AWG pour 4 situations choisies.

4.2.2 Par échantillonnage adaptatif

Pour réduire l'insertion de multiples situations d'attente dans les solutions de transport de marchandises multi-robots, l'approche *Adaptive Cross Sampling with Probabilistic RoadMap* (ACS-PRM) propose de réaliser un échantillonnage adaptatif de l'espace permettant une répartition des graphes PRM entre les robots en 3 phases :

– Une première phase de génération de configurations en correspondance avec une grille d'occupation.

– Une deuxième phase de construction d'un graphe PRM contenant des cibles potentielles et des jalons.
– Une troisième phase de construction des chemins et de suivi d'exécution.

Il existe 2 principales approches permettant de résoudre le problème de la planification géométrique de mouvement pour un robot mobile. La première consiste à utiliser les algorithmes de recherche incrémentale tels que A* [87], D* [209] et *wavefront* [150]. L'utilisation de ces planificateurs nécessite une discrétisation ou un pavage de l'environnement, ce qui implique un compromis entre pas de discrétisation et temps de réponse attendu. Un grand pas de discrétisation permet une résolution fine et demande un temps important de calcul. Un petit pas de discrétisation permet une résolution approximative (au pas de résolution près choisi) et demande peu de temps de calcul. Un grand pas de discrétisation demande également beaucoup de mémoire et un petit pas de discrétisation demande peu de mémoire. La deuxième utilise une échantillonnage de l'espace des configurations, dont les principales variantes sont les méthodes PRM [136] et RRT [153]. Ces méthodes ont été proposées pour permettre la résolution de problème de plus grande dimension en planification de mouvement. L'idée principale de ces approches est d'éviter la construction explicite de la région obstacle dans l'espace des configurations. Les méthodes basées sur l'échantillonnage de l'espace des configurations obtiennent de bons résultats dans les environnements complexes et dans les espaces de grande dimension. Différentes variantes des méthodes PRM et RRT ont été proposées [7, 228, 208, 112, 85] pour optimiser l'échantillonnage en fonction des caractéristiques des robots et des espaces.

La solution ACM-PRM réalise un échantillonnage représentatif de l'espace des configurations. La notion de représentativité est définie lors de l'exécution par observation de la densité des nouvelles configurations. L'idée principale de cette première étape est de rétracter les valeurs aléatoires en positions à une distance prédéfinie des obstacles. Cette distance aux obstacles est évaluée parallèlement aux axes d'un repère global et par relation à des configurations de C_{obs}. La distance d permettant d'éviter les collisions avec des obstacles est égale à la somme d'un nombre positif w [20] et un rayon r avec son centre sur l'axe de rotation du robot [21] :

$$d = r + w, \ (w > 0) \tag{4.7}$$

C_{obs} représentant l'ensemble des configurations en collision avec les obstacles, $\forall q \in C_{obs}$ il s'agit de définir une direction r_q, puis de déterminer un point de symétrie $S(q)$ tel que :

20. La valeur de w est définie par l'utilisateur en fonction de l'environnement. Cette valeur est une marge d'erreur pour les valeurs fournies par les capteurs.

21. La valeur de r est définie par le cercle englobant du robot dont le profil vu de dessus est englobé par le cercle. Cette valeur majore une distance de non-collision avec les obstacles, qui dépend en réalité de l'angle du robot.

$$S(q) = \{q + t\overrightarrow{r_q} | t > 0\} \cap C_{obs} \tag{4.8}$$

où, si :

$$\{q + t\overrightarrow{r_q} | t > 0\} \cap C_{obs} = \{q\} \tag{4.9}$$

alors :

$$S(q) = \infty \tag{4.10}$$

Sachant $dist(x,y)$ la distance entre le point x et le point y, la fonction de rétraction est définie par :

$$P(q) = \begin{cases} q + d\overrightarrow{r_q} & \text{si } dist(q, S(q)) \geq 2d \\ \frac{q+S(q)}{2} & \text{sinon} \end{cases} \tag{4.11}$$

où $P(q)$ est la position rétractée du point p. Ainsi, l'échantillonnage aléatoire E est adapté (Fig. 4.5(b)) selon l'équation suivante :

$$E = \{P(q) | q \in C_{obs}\} \tag{4.12}$$

L'algorithme 14 présente le principe d'échantillonnage de la solution ACS-PRM. Cet algorithme présente une complexité en temps de $O(n)$ et en espace constant.

La deuxième étape est la construction du graphe PRM, dans lequel les cibles potentielles et les jalons sont extraits et connectés à l'aide du graphe PRM. A partir de la rétraction des contours obtenue dans la première étape, la construction du graphe PRM est définie par :

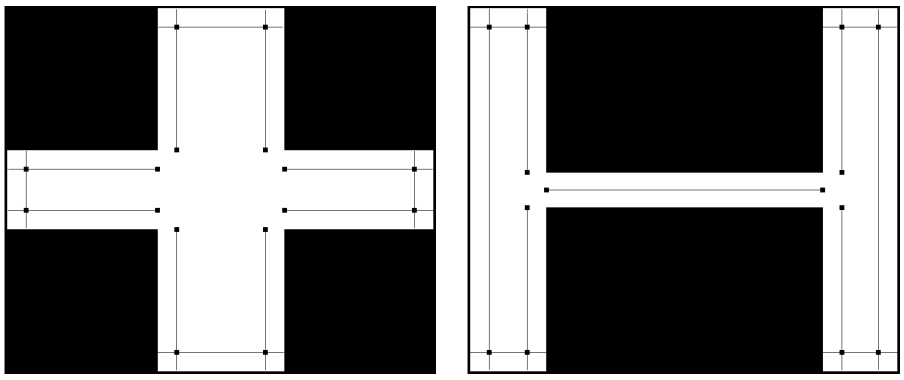
- Une première phase d'extraction de cibles potentielles, qui se déroule comme suit : si $dist(p,q) < d$ alors p se rétracte selon $\frac{q+S(q)}{2}$ et est étiqueté comme appartenant à l'axe médian de C ; un ensemble de segments ayant une longueur l [22] est ainsi défini ; le milieu d'un segment est marqué comme une cible potentielle (Fig. 4.5(c)) ;
- Une deuxième phase d'extraction de jalons à partir des terminaisons des segments ne contenant pas de cible potentielle (Fig. 4.5(d)) ;
- Une troisième phase d'extraction de jalons à partir des intersections des contours (Fig. 4.5(d)).

22. Par simplification, il est courant de fixer la valeur de l à la largeur maximale des obstacles. Si la taille des obstacles varie significativement, il est cohérent de choisir plusieurs valeurs l approchant la largeur de l'obstacle considéré.

Function `buildAcsSet` (E, n)

$nb \leftarrow 0$;
while $nb < n$ **do**
  $p \leftarrow$ `rand` ( ) ;
  if $p \in C_{libre}$ **then**
    foreach `axis-aligned direction` *of* C **do**
      $q \leftarrow$ `closest` (p, C_{obs}) ;
      if $dist(p,q) \geq d$ **then**
        $p \leftarrow q + d\overrightarrow{r_q}$;
      else
        $\{S(q)\} \leftarrow \overrightarrow{qp} \cap C_{obs}$;
        $p \leftarrow (q + S(q))/2$;
      end
    end
    $E \leftarrow E \cup p$;
    $nb \leftarrow nb + 1$;
  end
end
return ;

Algorithme 14 : Construction d'un échantillonnage adaptatif E de taille n.



(a) situation A (b) situation B

FIGURE 4.4 – Cas limite par échantillonnage adaptatif

La figure 4.5 illustre le processus de génération PRM à l'aide d'ACS-PRM pour un exemple de carte grille d'occupation contenant 200000 échantillons aléatoires.

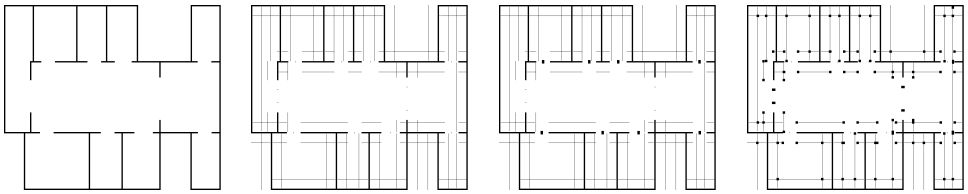


FIGURE 4.5 – Processus de génération PRM à l'aide d'ACS-PRM. (a) présente la carte grille originale. (b) présente le résultat d'échantillonnage croisé adaptatif de C. (c) présente les cibles potentielles extraites (*i.e.* les portes). (d) présente les jalons extraits.

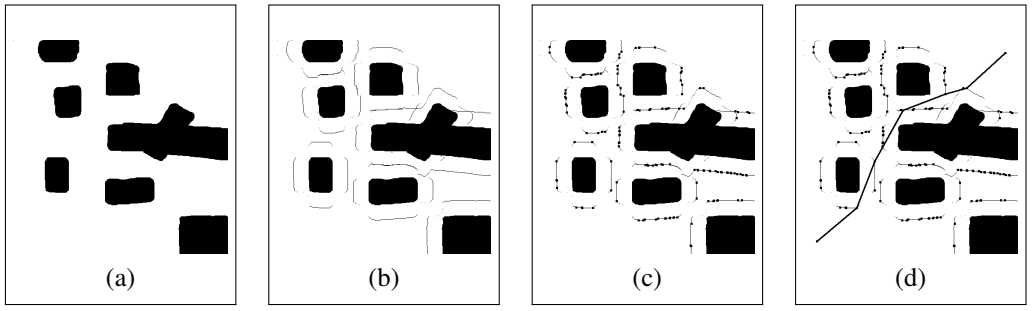


FIGURE 4.6 – Résultats en environnement irrégulier avec l'approche ACS-PRM. (a) présente un environnement irrégulier. (b) présente les configurations générées par l'approche ACS-PRM. (c) présente les jalons extraits. (d) présente un exemple de chemin PRM.

La troisième étape génère un chemin pour chaque robot. Ces chemins sont établis par interrogation du graphe PRM précédemment construit. Cette étape est articulée autour de 3 principes :

- Les cibles potentielles sont considérées comme nœuds objectifs de la planification et comme objets de la répartition de tâches. Ainsi les robots sont affectés aux cibles potentielles.
- L'attribution d'une cible potentielle est définie par proximité à chaque robot et par distance à la cible précédente. Cette stratégie permet de maximiser les différences dans les chemins des différents robots.
- La planification en ligne droite du chemin de chaque robot dans l'espace des configurations entre 2 configurations, correspondant au mécanisme classiquement utilisé dans les méthodes PRM et RRT.

Les résultats obtenus montrent l'absence de gain pour les espaces réduits limités à un unique robot (contenant des intersections ou des passages étroits).

La figure 4.6 présente les résultats obtenus dans un environnement irrégulier non structuré avec 20000 échantillons aléatoires. Dans les environnements irréguliers, les zones d'intersections et les passages étroits sont d'autant plus difficiles à identifier de par la complexité des espaces des configurations.

4.3 Conclusion

Dans ces travaux sur la robotique mobile collective, nous avons étudié les problèmes de fourragement, de transport et d'exploration. Du point de vue abstrait (par action) de la planification de tâches ou plus concret (proche des consignes de commande et des questions de géométrie dans les déplacements) de la planification de mouvement, l'utilisation de plusieurs robots permet de réduire le temps d'exécution des missions.

Chapitre 5

Monte Carlo parallèle et Analyse Rétrograde

Sommaire

Nous décrivons dans ce chapitre les résultats de nos travaux de recherche concernant la parallélisation [1] et la construction de bases de finales par analyse rétrograde. Les jeux concernés sont des jeux à 1 joueur de type réussite (avec pour objectif de trouver la plus grande suite de coups possibles) et de type puzzle (avec pour objectif de trouver la solution la plus rapide en temps ou la plus courte en nombre de coups), des jeux à 2 joueurs compétitifs en tour par tour, à information complète et imparfaite [2]. Plus généralement, ces travaux concernent la résolution

1. Dans la suite des résultats de nos travaux sur la parallélisation de MCTS appliquée au Go présentés dans la section 2.2.2, la parallélisation concerne dans cette partie d'autres algorithmes de recherche arborescente appliqués au domaine du jeu en utilisant des ordinateurs multi-cœurs ou des réseaux d'ordinateurs.

2. La notion de connaissance parfaite et complète permet de définir la complexité d'un jeu. Un jeu est à connaissance parfaite quand tous les joueurs connaissent tous les mouvements précédemment réalisés par les autres joueurs. Un jeu est à connaissance complète quand tous les joueurs

de problèmes de jeu [3].

5.1 Jeux à 1 joueur

Nous présentons dans cette partie, nos travaux concernant la parallélisation de résolution Monte Carlo et l'identification de fins de parties prématurées par analyse rétrograde dans 2 jeux à 1 joueur, avec pour objectif la diminution du temps de résolution de ces problèmes. Avec la parallélisation, l'objectif est de réduire linéairement le temps de calcul avec le nombre de processus impliqués. Avec l'analyse rétrograde, l'objectif est de pré-calculer des états bloquants et de réduire l'espace nécessaire au stockage de ces états pour les exploiter dans une recherche Monte Carlo.

connaissent les stratégies et les gains de tous les joueurs. Un jeu est statistique quand une part de hasard intervient pour déterminer l'état suivant du jeu.

3. La résolution de problèmes de jeu est définie selon 3 catégories : *ultra-weak* pour lesquels victoire ou défaite sont prédictibles par application de stratégies sans preuve constructive et sans confrontation avec un joueur parfait (ce qui suppose de disposer d'une évaluation théorique ou pratique de la position initiale) ; *weak* pour lesquels un programme donne la meilleure réponse face à un joueur parfait (ce qui implique d'avoir un joueur parfait ou de le spécifier et d'avoir une stratégie permettant au programme considéré de jouer) ; *strong* pour lesquels un programme donne la meilleure réponse face à tout joueur (ce qui implique : la possibilité d'erreurs dans les coups précédents des 2 joueurs et dans les réponses futures du joueur adverse ; d'avoir une stratégie applicable dans tous les états possibles.). Un jeu *strongly solved* implique d'énumérer tous les états possibles de ce jeu. Le nombre de jeux résolus est de l'ordre de la trentaine. Parmi les plus connus : *Hex* est *ultra-weakly solved* ; Le *Jeu de Dames* 8x8 (*i.e. Checkers*, *Draughts*) est *weakly solved* depuis 2007 [200, 148] ; Le jeu *Awari* (*i.e. Awalé*, *Awélé*) est *strongly solved* depuis 2003 [191]. Ces résolutions impliquent des puissances de calculs importantes. Pour le *Jeu de Dames*, J. Schaeffer *et al.* ont utilisé jusqu'à 90 ordinateurs et 1 super-calculateur *BBN TC2000*. Pour le jeu *Awalé*, J.W. Romein et H.E. Bal ont utilisé un cluster IBM de 144 processeurs.

5.1.1 Résolution parallèle au *Morpion Solitaire*

Le *Morpion Solitaire* (MS) est un jeu à 1 joueur NP-difficile [58] [4]. A chaque tour, l'objectif est de trouver un mouvement valide. Ce mouvement doit permettre de continuer à jouer par la suite, sinon le jeu s'arrêtera. Un mouvement consiste à placer un nouveau cercle aligné avec d'autres cercles déjà présents sur le jeu. Les cercles doivent être placés sur une grille fictive infinie. L'alignement des cercles forme une nouvelle ligne sur le jeu. Chaque nouvel alignement de cercles est valide uniquement s'il ne recouvre pas une nouvelle ligne. Les lignes peuvent se croiser ou partager un cercle mais pas se recouvrir. Ainsi les lignes parallèles avec 2 cercles ne sont pas valides. Les lignes peuvent être horizontales, verticales ou diagonales. La figure 5.1 montre la position de départ contenant 36 cercles. Les règles classiques définissent le MS Touchant. La variante MS Disjointe implique une contrainte supplémentaire : les lignes parallèles ne peuvent pas se toucher.

4. La complexité d'un algorithme (ou d'une méthode de résolution) est un critère d'évaluation et de comparaison des algorithmes. Pour remédier aux différences entre ordinateurs, la définition de la complexité s'accompagne d'un modèle abstrait d'ordinateur, visant à définir l'unité élémentaire d'espace mémoire et l'unité élémentaire opératoire. L'unité élémentaire d'espace mémoire définit les variables occupant une unité de stockage en mémoire. L'unité élémentaire opératoire définit les opérations exécutées en une unité de temps. De ce fait, la complexité s'exprime classiquement en espace ou en temps. La complexité en espace (respect. en temps) d'un algorithme est le nombre d'unités élémentaires d'espace (respect. opératoires) requis pour son exécution. La résolution d'un algorithme dépendant des données assignées à la définition du problème, la complexité est au pire ou en moyenne. La complexité au pire donne la borne supérieure de la complexité. La complexité en moyenne donne la moyenne du nombre d'unités élémentaires utilisées. La complexité d'un problème est définie par la complexité de sa résolution. Elle est notée $O(f(m))$ si le nombre d'unités élémentaires nécessaires (en temps ou en espace) suit asymptotiquement les variations définies par la fonction $f(m)$. m définit dans ce cas la nature des données initiales du problème. Si $f(m)$ est un polynôme de degré constant indépendant de m, la complexité est polynomiale. Une complexité non polynomiale est exponentielle. La complexité des problèmes est couramment divisée en cinq classes, allant des plus simples aux plus difficiles : la classe *P* pour les problèmes polynomiaux déterministes en temps, la classe *NP* pour les problèmes polynomiaux non-déterministes en temps, la classe *PSPACE* pour les problèmes polynomiaux non-déterministes en espace (tels que *Hex*, *Havannah* et *Amazons*), la classe $EXPTIME$ pour les problèmes exponentiels en temps (tels que le *jeu des Échecs* et le *Go*) et la classe $EXPSPACE$ pour les problèmes exponentiels en espace. Un problème est *P* s'il est associé à un algorithme de longueur polynomiale en la taille de ses données. Un problème est *NP* s'il n'a pas d'algorithme de ce type, mais la validité d'une solution à ce problème est vérifiée par un algorithme de ce type. Un problème polynomial non-déterministe n'est pas difficile en soi puisque sa résolution peut se limiter à l'énumération de tous les états possibles et au choix du meilleur état. La problématique d'une telle résolution se situe dans le nombre de ces configurations et des liens reliant ces états. Les règles des jeux influent particulièrement les liens possibles entre les états et la nature des états (dans leurs possibilités d'être des états non valides ou non atteignables), la complexité des jeux est souvent mesurée selon le nombre de positions légales de l'espace d'états. Cette complexité peut être exacte ou statistiquement approximée par des méthodes Monte Carlo en générant des positions et en regardant le ratio des positions légales et la profondeur théorique de l'arbre de recherche.

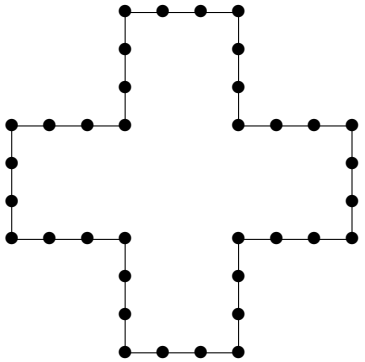


FIGURE 5.1 – Position initiale de MS. Les points noirs représentent les 36 cercles initiaux.

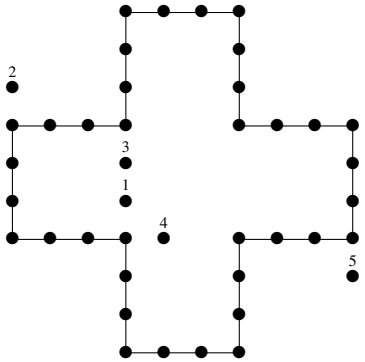


FIGURE 5.2 – 5 premiers mouvements MS d'une des solutions de notre résolution parallèle [41].

La variante Disjointe est de fait plus difficile que la variante Touchante. Pour la version à 4 cercles, les meilleurs scores possibles sont de 62 pour la variante T et de 35 pour la variante D. Pour la version à 5 cercles, le meilleur score réalisé par un humain est de 68 mouvements [58]. La figure 5.2 montre les premiers mouvements d'une des solutions proposées par notre résolution parallèle de la variante D à 5 cercles.

Pour résoudre des problèmes similaires, H. Hyyrö et T. Poranen [102] ont proposé des algorithmes de recuit simulé avec échantillonnage aléatoire pour les variantes T et D et ont obtenu les meilleurs résultats pour la variante D. Les mouvements parmi les positions précédentes et suivantes sont choisis aléatoirement sans heuristique.

F. Guerriero et M. Mancini [84] ont proposé deux algorithmes parallèles utilisant des stratégies de résolutions appliquées au problème du voyageur de commerce (TSP) et le problème d'ordonnancement séquentiel (SOP). Avec des stratégies spéculatives, ils ont obtenu une accélération moyenne de 2.53 avec 8 processeurs pour résoudre les problèmes SOP et TSP. Ils ont obtenu des gains en vitesse de résolution allant de 3.89 à 6.64 avec 8 processeurs.

T. Cazenave [37] a proposé une amélioration de la recherche Monte Carlo en imbriquant des recherches Monte Carlo à chaque descente d'un niveau de profondeur. Cette variante est adaptée à la résolution des problèmes sans heuristique. Ces travaux montrent que l'utilisation imbriquée de recherches Monte Carlo permet d'amplifier les résultats statistiques obtenus dans les descentes et ainsi d'améliorer la recherche classique.

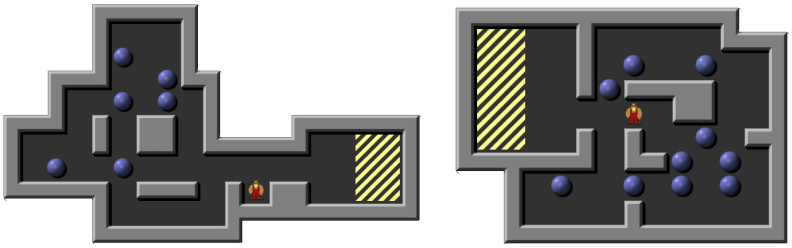
Afin de paralléliser les recherches Monte Carlo imbriquées[41], nous avons proposé un algorithme parallèle organisé selon 4 types de processus : le *processus-maître*, les *processus-médians*, un *processus-dispatcher* et des *processus-clients*.

Ces processus fonctionnent à différents niveaux d'imbrication : le *processus-maître* au 1er niveau (le plus haut niveau d'imbrication), les *processus-médians* au 2ème niveau et les *processus-clients* au 3ème niveau. Le *processus-maître* joue un jeu au 1er niveau et invite les *processus-médians* à jouer à des jeux de 2ème niveau. Les *processus-médians* demandent aux *processus-clients* de jouer à des jeux de 3ème niveau en parallèle. Les *processus-clients* sont soit en attente de requête de calcul, soit en cours de résolution d'une requête de calcul d'un *processus-médian*. Le *processus-dispatcher* est utilisé pour indiquer aux *processus-médians* les *processus-clients* libres. Avec cette organisation, nous avons proposé 2 algorithmes : l'un utilisant le principe du tourniquet et l'autre exploitant la disponibilité des *processus-clients*. Les expérimentations ont été réalisées sur un cluster d'ordinateurs composé de 20 ordinateurs *dual-core* 1.86 Ghz, de 12 ordinateurs *dual-core* 2.33 Ghz et d'1 serveur reliés par un réseau *Gigabit*. Les communications entre processus ont été réalisées avec OpemMPI [76] dont le principe de communication par message est adapté aux réseaux hétérogènes. Les communications ont été réalisées dans le communicateur global `MPI_COMM_WORLD`. Le serveur a supporté l'exécution d'1 *processus-maître*, d'1 *processus-dispatcher* et de 40 *processus-médians*. Chacun des ordinateurs du cluster a supporté l'exécution simultanée de 2 *processus-clients*. Toutes nos expérimentations concernent la variante D avec 3 et 4 niveaux d'imbrication de la recherche Monte Carlo. Avec 4 niveaux d'imbrication, nous avons trouvé 2 nouvelles séquences de mouvements de taille 80 avec un facteur d'accélération de 56 en utilisant 64 *processus-clients*.

5.1.2 Analyse rétrograde à *Sokoban*

Sokoban est un problème *PSPACE*-complet [55] comportant de nombreuses situations bloquantes (*i.e. deadlock*) qui en font un problème difficile, même dans le cas de solutions non-optimales. L'identification des *deadlock* constitue donc une étape importante dans la résolution de *Sokoban*. L'objectif du jeu est de placer des caisses sur des emplacements de stockage précis dans un entrepôt. Le jeu se joue sur une grille avec des déplacements verticaux et horizontaux. Les cases sont de deux types : espace libre ou obstacle. Les espaces libres peuvent contenir soit un personnage soit une caisse. Le personnage peut pousser une caisse et seulement une à la fois. Le personnage ne peut se déplacer et ne peut pousser les caisses que dans les espaces libres. Le problème est résolu quand toutes les caisses sont sur des emplacements de stockage. La figure 5.3 montre les problèmes 1 et 2 des 90 problèmes standards proposés par A. Junghanns et J. Schaeffer [131]. Les caisses sont représentées par des puces bleues et les emplacements de stockage par des hachures jaunes. Les espaces libres sont en gris foncé et les obstacles en gris clair.

Dans les solutions proposées par A. Junghanns et J. Schaeffer [130, 132, 133], les *deadlock* sont assimilées à des motifs de taille 5x4 et un algorithme de re-

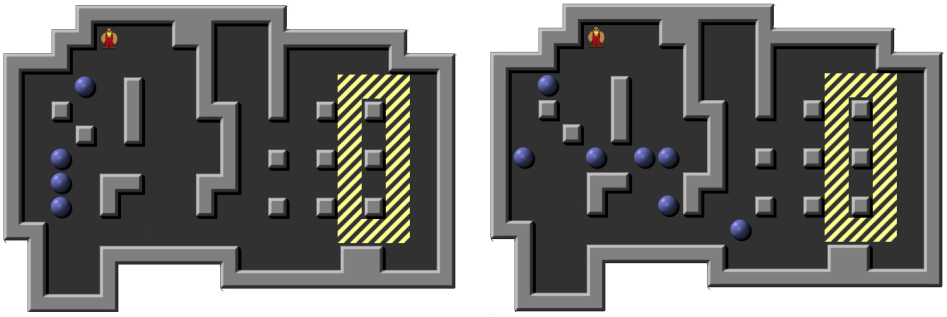


(a) Problème 1. (b) Problème 2.

FIGURE 5.3 – Exemples de problèmes *Sokoban*.

cherche spécifique identifie les possibilités de *deadlock* supplémentaire par proximité de motifs. Les motifs sont identifiés au fil de l'exécution et sont insérés dans une base de connaissance.

Afin d'extraire la détection des *deadlock* de la recherche des chemins [42], nous avons proposé un algorithme permettant d'identifier les *deadlock* par analyse rétrograde (utilisée avec succès dans des problèmes à 1 joueur [143, 56, 144, 70]). Les *deadlock* initiales sont établies par un ensemble de règles simples (pierres dans les coins, pierres dans une concavité, pierres jointes et adjacentes à un bord).



(a) *deadlock* à 4 pierres. (b) *deadlock* à 7 pierres.

FIGURE 5.4 – Exemples de *deadlock* du problème 13.

La figure 5.4 montre deux exemples de *deadlock* du problème 13 ainsi obtenues. Nous avons calculé des bases de *deadlock* de 1 à 12 pierres. Chaque base est spécifique à chaque problème. Combinées avec un algorithme glouton, elles permettent de réduire le nombre d'itérations nécessaires à la résolution du problème. Pour le problème 13, le nombre d'itérations est supérieur à 1 million avec les *deadlock* à 5 pierres et passe à 1372 avec les *deadlock* à 10 pierres. Le nombre d'itérations reste cependant identique avec les *deadlock* à 11 pierres et à 12 pierres. Les résultats montrent que les bases de *deadlock* contribuent à des simplifications

par palier et permettent de résoudre des problèmes précédemment non-résolus par l'algorithme IDA*.

5.2 Jeux à 2 joueurs

Nous présentons dans cette partie, nos travaux concernant la parallélisation de la recherche en meilleur d'abord et la construction de bases de finales pour les jeux à 2 joueurs. Les principaux algorithmes en meilleur d'abord pour les jeux à 2 joueurs sont B*[21], *Conspiracy Number Search* [4, 199] et *Proof Number Search* [6] (PN). Ces algorithmes évaluent les nœuds pour définir les nœuds les plus prometteurs au fil de la construction de l'arbre. Ainsi B* propose d'utiliser deux évaluations (une pessimiste et une optimiste) pour classer les coups possibles entre eux. Les coups ayant des valeurs pessimistes supérieures à toutes les valeurs optimistes d'un coup sont de fait meilleurs que ce coup. Avec *Conspiracy Number Search*, le développement de l'arbre est guidé vers les feuilles qui, avec un minimum de changement, influeront sur les valeurs minimum des nœuds-parents. Avec PN, il devient possible de prouver ou de réfuter des objectifs binaires (tels que victoire et défaite) dans des arbres. A chaque descente dans l'arbre, PN recherche l'évaluation minimale du nœud courant en utilisant les nœuds-fils. A chaque nœud, un *proof number* (PN) définit le coût de prouver la victoire et un *disproof number* (DN) définit le coût de prouver la défaite. Par défaut, les nœuds possèdent un couple (PN, DN) avec des valeurs (1, 1). Pour une fin de partie gagnante, un nœud prends les valeurs (0, ∞). Pour une fin de partie perdante, un nœud prends les valeurs (∞, 0). Dans un arbre ET/OU [5] : Un état impliquant un ensemble de nœuds-fils OU a pour valeurs (PN, DN) le couple (A, B) avec A minimum des valeurs PN des fils et B somme des valeurs DN des fils ; Un état impliquant un ensemble de nœuds-fils ET a pour valeurs (PN, DN) le couple (A', B') avec A' somme des valeurs PN des fils et B' minimum des valeurs DN des fils. Aux nœuds OU, réduire la résolution équivaut à choisir le plus petit PN. Aux nœuds ET, réduire la résolution équivaut à choisir le plus petit DN. Avec la parallélisation de PN, l'objectif est de réduire le temps de calcul. En distribuant PN sur un réseau d'ordinateurs, l'objectif est de partager les résolutions et ainsi résoudre des problèmes ayant une plus grande complexité en espace.

5. Les arbres ET/OU sont adaptés à la représentation des problèmes en conjonctions et disjonctions de sous-problèmes. Dans la formulation générale, conjonctions et disjonctions peuvent être à des profondeurs identiques. Dans le cas des jeux à 2 joueurs, le nœud racine représente le problème de l'un des joueurs, les mouvements adverses sont représentés par des conjonctions de sous-problèmes et les mouvements du joueur considéré sont représentés par des disjonctions de sous-problèmes. Les coups du joueur adverse sont donc représentés par des nœuds ET et les coups du joueur considéré par des nœuds OU.

5.2.1 Parallélisation de PN : PPN^2

Pour résoudre les jeux à 2 joueurs, de nombreuses variantes [6] séquentielles et parallèles de PN ont été proposées [140].

L.V. Allis et S. Keetman [5] ont proposé l'algorithme PN^2 qui remplace l'évaluation des feuilles de la position prometteuse par une nouvelle recherche PN pour ces feuilles (prenant la position prometteuse comme racine). PN^2 réalise donc une deuxième recherche PN, pour sélectionner les feuilles les plus prometteuses, permettant ainsi de réduire la mémoire nécessaire en augmentant le temps de calcul. Les résultats des expérimentations sur *Awalé* et le *Jeu de Dames* montrent un nombre de nœuds explorés 3 fois plus grand qu'avec PN.

M. Seo [204] a proposé l'algorithme PN* qui réalise une recherche en profondeur d'abord sur les valeurs PN afin de réduire la mémoire nécessaire. Un seuil définit le nombre de feuilles maximales à prouver. Au delà de ce seuil, les nœuds-parents deviennent non prometteurs et les parties des sous-arbres correspondants sont élaguées.

A. Nagai [171] a proposé l'algorithme *Proof-number and Disproof-number Search* (PDS) qui applique PN* aux valeurs DN. Le principe d'une valeur seuil est appliqué aux valeurs DN pour stopper le développement de certaines parties de l'arbre. Un échec lors de la phase de sélection des feuilles prometteuses implique la redéfinition de l'un des deux seuils.

M.H.M. Winands *et al.* [229] ont proposé l'algorithme PDS-PN reprenant le principe de PN^2 pour l'appliquer à PDS. L'évaluation des feuilles de PDS est remplacée par une recherche PN pour ces feuilles. Seules les feuilles les plus prometteuses de la recherche PDS sont sélectionnées et les arbres PN développés dans la seconde recherche ne sont pas mémorisés. En limitant le nombre de nœuds des recherches, les expérimentations montrent de meilleurs résultats (en nombre de problèmes résolus) à *Lines of Action* (LOA) [7].

A. Nagai [172] a proposé l'algorithme *Depth First Proof Number* (DF-PN) utilisant les couples (PN, DN) comme PDS et limitant le recalcul de ces couples

6. Le nombre de variantes est justifié par l'existence de 3 défauts dans l'utilisation d'un algorithme PN : Le besoin important de capacité mémoire ; La dépendance entre évaluation et précédence d'un nœud (un nœud atteignable par 2 chemins pourra avoir 2 valeurs d'évaluation distinctes. Ce problème est appelé *Graph History Interaction* et implique des possibilités de sous-estimation et de surestimation des valeurs PN-DN.) ; L'élimination des cycles dans les parties par souci de rapidité de résolution. Le nombre de variantes est également justifié par l'inadéquation d'*alpha-beta* concernant la recherche de coup sans parade en fin de partie. Selon la complexité des jeux, les bases de finales ou les extensions d'*alpha-beta* ne pourront pas établir de séquences de *mat* assez grandes. La recherche PN remédie à ce problème.

7. *Lines of Action* se joue sur un échiquier 8x8 avec 12 dames blanches et 12 dames noires, initialement placées centrées sur les bords. L'objectif pour noir est de déplacer les dames noires pour les connecter en utilisant les 8 cases voisines de chaque case. La longueur et les possibilités des déplacements sont conditionnées par les positions des autres dames sur l'échiquier.

dans les nœuds-parents aux feuilles supérieures à la valeur du nœud plus prometteur courant. La recherche est itérée tant que PN et DN sont inférieurs à des valeurs de terminaison.

T. Ueda *et al.*[221] ont proposé de réduire les problèmes de surestimation des valeurs de PN avec l'algorithme *Weak Proof-Number Search* (WPNS). Les valeurs PN des nœuds ET et les valeurs DN des nœuds OU deviennent le nombre de nœuds-fils ajouté aux valeurs des meilleurs nœuds-fils. En réduisant des valeurs PN et DN à 1, WPNS sous-estime les coûts d'évaluation des nœuds.

J.T. Saitoa *et al.*[197] ont proposé l'algorithme *Randomized Parallel Proof-Number Search* qui s'applique à PN (RP-PNS) et à PN^2 (RP-PN^2). L'ensemble de l'arbre est en mémoire partagée et un ensemble de recherches parallèles est réalisé à proximité de la variation principale selon une probabilité de distribution. Les expérimentations sur LOA avec 8 *threads* montrent un facteur d'accélération de 3.5 avec RP-PNS et de 4.7 avec RP-PN^2.

I.C. Wu *et al.*[231] ont proposé l'algorithme *Job-level proof-number search* (JL-PN) qui étend les nœuds les plus prometteurs en parallèle dans des ordinateurs clients. En fonction de l'occupation processeur des clients, il est possible d'évaluer des nœuds prometteurs supplémentaires. Les expérimentations ont été réalisées sur les débuts de partie de *Connect6* avec les stratégies *virtual-win*, *virtual-loss* [8] et une stratégie gloutonne [9], en utilisant de 1 à 8 *processus-clients*. Les résultats montrent des facteurs d'accélération linéaires et supra-linéaires selon les cas.

M.P.D. Schadd *et al.*[198] ont résolu *Fanorana* [10] en combinant PN^2 avec une base de fins de parties à 7 pièces établie par analyse rétrograde. PN^2 a été utilisé pour prouver qu'avec 2 joueurs parfaits, le score d'une partie est toujours nul (*i.e.* égalité des scores en fin de partie).

A. Saffidine et T. Cazenave[193] ont proposé l'algorithme *Generalized proof number search* (GPNS) qui étend l'utilisation de PN aux jeux à somme nulle dont les scores dépassent la simple notion de *gagné/perdu*. Pour m valeurs de score possibles [11], chaque nœud est associé à $2n$ *effort numbers* possibles avec n dans $[1,m]$. Les expérimentations montrent des temps de résolution 21 fois plus petit que PN pour *Connect Four* et des temps de résolution équivalents à PN pour *Woodpush*.

Pour étendre les possibilités de résolution parallèle avec PN, nous avons proposé l'algorithme ParallelPN^2 qui parallélise la recherche PN^2 [195]. Destiné à un environnement distribué, ParallelPN^2 exécute la première recherche PN dans

8. Les stratégies *virtual-win* et *virtual-loss* permettent ici de considérer des nœuds comme systématiquement vainqueurs ou perdants afin de dévier la sélection des nœuds les plus prometteurs.
9. La stratégie gloutonne applique *virtual-win* ou *virtual-loss* en fonction d'une évaluation.
10. *weakly solved* pour les problèmes 9x5 ou moins.
11. m valeurs ou m intervalles de valeurs dans les entiers ou les réels.

```
Function masterLoopParallelPN2 ( )

Q ← MostPromisingLeaves ( ) ;
sendAndReservePositionsToClients ( Q ) ;
while not proved ( root ) do
    ( res_id , id ) ← recvFromAnyClient ( ) ;
    q ← getReservedLeaf ( id ) ;
    if isPartial ( res_id ) then
        ( PN, DN )_q ← update ( (PN, DN)_q, res_id ) ;
        backPropagateFrom ( q ) ;
    else
        discard ( q ) ;
        expandTree ( ) ;
        ( PN, DN )_q ← update ( (PN, DN)_q, res_id ) ;
        backPropagateFrom ( q ) ;
        if not proved ( root ) then
            q ← newUnreservedMostPromisingLeaf ( ) ;
            reserve ( q ) ;
            sentPositionToClient ( q, id ) ;
        end
    end
end
GatherFromClientsAndDiscardLeaves ( ) ;
return ;
```

Algorithme 15 : Résolution PN du *processus-maître* de ParallelPN^2.

le *processus-maître* et les recherches PN des feuilles dans des *processus-clients*. L'algorithme 15 montre la boucle principale du *processus-maître*. Après une première recherche PN, le *processus-maître* envoie une feuille à évaluer à chaque *processus-client*. A chaque envoi d'une position correspondant à un nœud, le *processus-maître* réserve ce nœud pour indiquer qu'un *processus-client* exécute une recherche PN sur ce nœud (et ainsi ne pas affecter ce nœud à un autre *processus-client*). Les *processus-clients* retournent des résultats partiels au *processus-maître* à une fréquence fixée [12]. La fonction `recvFromAnyClient` permet de récupérer le résultat de la recherche PN (*i.e.* variable res_{id}) du *processus-client id*. La fonction `getReservedLeaf` permet de retrouver le nœud q associé à ce *processus-client*. La réception d'un nouveau résultat final (différencié d'un résul-

12. Fréquence fixée expérimentalement à 100 itérations de recherche PN dans les *processus-clients* pour réduire le temps global de résolution de *Breakthrough* 4x5.

tat final par la fonction `isPartial`) implique la libération du nœud précédemment réservé, l'expansion de l'arbre, sa mise à jour et la rétropropagation des valeurs (PN, DN) à partir de *q*. Ensuite, si le nœud *root* n'est pas prouvé, alors le *processus-maître* sélectionne, réserve et envoie une nouvelle feuille prometteuse *q* au *processus-client id*. Si le nœud *root* est prouvé, alors la recherche PNS prend fin et le *processus-maître* collecte les réponses restantes des *processus-clients*.

Les expérimentations de ParallelPN^2 ont été réalisées sur *Breakthrough*, un jeu de stratégie à 2 joueurs se déroulant sur un damier de taille variable [13]. Ce jeu est uniquement composé de simples pions. Un pion peut se déplacer en avançant, sur les 3 cases devant lui si elles sont libres ou se déplacer en capturant sur les 2 cases en diagonale devant lui si elles sont occupées par des pions adverses. Le 1er joueur à atteindre le bord adverse gagne la partie. Si un joueur n'a plus de pièce à jouer, il perd la partie. Les mouvements de capture ne sont pas obligatoires à jouer. La figure 5.5 montre la position de départ sur un plateau 8x8. Pour le joueur blanc, avancer un pion sur la ligne 8 équivaut à gagner. Pour le joueur noir, avancer un pion sur la ligne 1 équivaut à gagner. La figure 5.6 montre une position avancée après 86 coups. Le tour est à blanc et ses mouvements possibles sont `e5-f6`, `e3-e4` et `e3-f4`. Si blanc joue `e3-f4`, les mouvements possibles de noir seront `f6-e5` et `f6-f5`.

Dans la pratique, résoudre *Breakthrough* est équivalent à trouver un chemin vers la ligne opposée. Le contrôle du centre en début de partie permet de construire une bonne position pour l'établissement d'un tel chemin. L'utilisation de positions *zugzwang* permet de forcer l'adversaire à jouer des coups non désirés et ainsi d'affaiblir sa position. En utilisant des sacrifices de pions, le joueur qui définit le chemin le plus court vers la ligne adversaire est gagnant. En exploitant ce principe, nous avons proposé la notion de *race pattern* qui permet, par enchaînement de motifs, de définir des chemins gagnants. Les chemins sont généralisés dans des matrices à 2 dimensions contenant des éléments définissant l'état des positions [14].

Les expérimentations de ParallelPN^2 ont été réalisées avec un réseau de 17 ordinateurs standards [15]. Avec 1k itérations par évaluation PN d'une position dans les *processus-clients*, le facteur d'accélération en temps de la résolution de *Breakthrough* 4x5 est de 18 avec 64 *processus-clients*. L'exécution de recherches PN

13. Classiquement 8x8.

14. Les motifs sont définis pour blanc et obtenus par symétrie pour noir. Les états des positions varient dans les valeurs suivantes : *occupée*, *libre*, *passive* (*i.e.* sans pion blanc et inutile pour noir), *franchissable* (*i.e.* en position capturable), *sans importance* (*i.e.* ne contribuant pas à la construction d'un chemin vers la base noire, ce qui est le cas des pions contribuant uniquement à la construction de chemins vers la base blanche). En fonction du placement des pièces, les valeurs possibles varient.

15. 17 ordinateurs avec processeur Intel i5 quad core 3.2 GHz avec 4GO de RAM (soit 1 ordinateur pour le *processus-maître* et 16 ordinateurs pour les *processus-clients*).

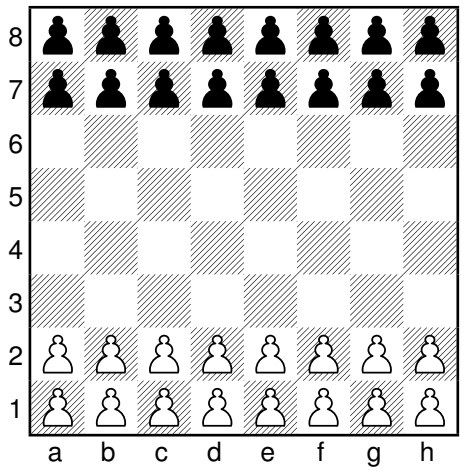


FIGURE 5.5 – Position initiale d'une partie classique 8x8.

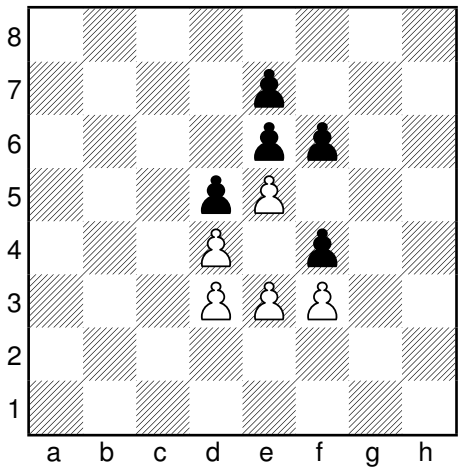


FIGURE 5.6 – Position avancée d'une partie classique 8x8.

disjointes dans les *processus-clients* impliquant des itérations inutiles, le nombre de nœuds créés est multiplié par 2.11 et le nombre de nœuds développés est multiplié par 2.17. Avec 64 *processus-clients*, la résolution de *Breakthrough* 4x5 a été achevée en 216[*sec*] avec 10k itérations par évaluation PN et en 577[*sec*] exécutant 100k itérations par évaluation PN. ParallelPN^2 a permis de résoudre *Breakthrough* 6x5, qui est toujours une partie gagnante pour le 2$^{\text{ème}}$ joueur.

5.2.2 Analyse rétrograde

Dans le cas des jeux à 2 joueurs, les bases de finales sont principalement utilisées pour améliorer le niveau des joueurs et pour résoudre les jeux. Elles sont calculées avec l'analyse rétrograde en partant des positions finales. Plus largement, par analyse rétrograde il est possible d'établir une séquence de coups optimaux dans un sous-ensemble spécifique d'un jeu (tel que la fin de partie mais également aux ouvertures et en milieu de partie) et ainsi de contribuer à la résolution des problèmes à 2 joueurs [217, 35, 36, 148, 191]. Les bases de finales ont été utilisées au *Jeu des Échecs* sur des positions particulières contenant 6 pièces [217, 216] ou optimisant des combinaisons de 3 à 6 pièces [173], au *Jeu de Dames* pour les positions à 10 pièces [201] [16], à *Awari* pour toutes les positions atteignables à 48 pierres [191] (*i.e.* 204 milliards de positions (soit 2x10^{11}) dans une base de finales de 178[Go]), au *Jeu des Échecs Chinois* (*i.e. Xiang Qi*) pour des combinaisons de

16. Les finales du *Jeu de Dames* ont été calculées sur 10 années : de 2 à 8 pièces pour 444 milliards de positions (soit 4.4x10^{11}) en 1993, à 9 pièces et 5-5 à 10 pièces pour 13 milliers de milliards de positions (soit 1.3x10^{13}) en 2003, compressées dans une base de finales de 215[Go] prévue pour un accès rapide permettant à un programme de l'utiliser dans un temps imparti.

pièces particulières [68] (*i.e.* 2 pièces en attaque, 4 pièces en défense et 2 rois) et au *Jeu de Go* pour des sous-problèmes particuliers [36] (*i.e.* obtenir un œil, vivre, connecter 2 groupes, connecter 2 intersections et capturer un groupe.). Plus récemment l'analyse rétrograde a été à *Kriegspiel* [17] pour la construction de finales à 3 pièces [51].

Les métriques utilisées dans les bases de finales varient. Les mesures les plus courantes sont une valeur indiquant si la position est résolue (*i.e.* toujours gagnée, toujours perdue ou indéterminée), le nombre de coups menant à la victoire et le nombre de coups permettant une modification de matériel (*i.e.* la capture d'une pièce adverse ou la transformation d'une pièce amie).

La taille importante de l'espace mémoire nécessaire pour stocker les bases de fins de partie obtenues par analyse rétrograde étant une contrainte limitant leur utilisation, J. Schaeffer *et al.* [201] ont proposé une solution de compression par suppression de positions de capture combinée à un algorithme de codage par plages permettant de représenter 155 positions avec 1 bit.

Nous avons proposé [194] une approche par symétries dans le matériel. Les symétries sont de 2 types : Dans les positions des pièces (qui permettent des symétries par rotation et par symétrie axiale) et dans le matériel en jeu (qui permet de définir des équivalences entre des ensembles de pièces). Ces équivalences ont été établies dans des graphes pour les jeux *Chinese Dark Chess*, *Skat* et au *Jeu des Dominos*. Ayant défini par équivalence des ensembles de représentants, les bases de fins de parties ont été calculées pour ces représentants par analyse rétrograde pour 8 pièces à *Chinese Dark Chess* et pour 6 pièces au *Jeu des Dominos*. Les résultats au *Jeu des Dominos* montrent un facteur de compression de 1541 pour les bases de finales à 6 pièces. Les résultats à *Chinese Dark Chess* montrent un facteur de compression de 9.92 pour les bases de finales à 4 pièces et de 3.68 pour les bases de finales à 8 pièces.

5.3 Conclusion

Dans ces travaux sur les problèmes de jeu à 1 et 2 joueurs, nous avons étudié les algorithmes parallèles et la construction de bases de finales par analyse rétrograde. Les gains en vitesse de résolution obtenus par parallélisation de MCTS sont quasi-linéaires jusqu'à 64 processeurs [18]. Les gains en vitesse de résolution obtenus par parallélisation de PN sont infra-linéaires [19]. Les algorithmes paral-

17. *Kriegspiel* (*i.e. Blind Chess*) est une variante à information incomplète du *Jeu des Échecs*. Chaque joueur ne voit pas les pièces de l'adversaire. Il résulte la possibilité de mouvements illégaux qui impliquent de demander au joueur concerné de jouer un autre coup.

18. Pour un facteur d'accélération de 56 avec 64 *processus-clients*.

19. Pour un facteur d'accélération de 18 avec 64 *processus-clients*.

lèles proposés ont permis des résolutions plus rapides à MS et la résolution de nouveaux problèmes à *Breakthrough*.

L'utilisation de l'analyse rétrograde dans le calcul de bases de sous-problèmes sans solution et de fins de parties a permis de résoudre de nouveaux problèmes à *Sokoban*, de proposer des solutions de compression générique pour *Chinese Dark Chess*, *Skat* et le *Jeu des Dominos*. Les résultats montrent des taux de compressions de 1541 pour les bases de finales à 6 pièces au *Jeu des Dominos* et de 3.68 pour les bases de finales à 8 pièces.

Les nombreuses participations dans les compétitions internationales *Computer Olympiad* et dans les compétitions de jeux abstraits (*c.f.* Annexe A) nous ont permis de valider en pratique les idées présentées dans ces travaux.

Chapitre 6

Conclusion et perspectives

Sommaire

6.1 Conclusion

Dans nos travaux principalement sur la robotique mobile, la coordination multi-robots, la robotique humanoïde et les jeux, nous avons abondamment exploré les solutions probabilistes, reposant sur un échantillonnage aléatoire de l'espace du problème et sur une hypothèse de convergence dans le temps. Séquentielles, parallèles, distribuées, évolutionnaires, nous avons présenté des contributions dans des domaines proches et différents. De nombreuses pistes telles que les algorithmes probabilistes, les méthodes de coordination implicites et explicites, les algorithmes MCTS à budget prédéfini, les algorithmes PN, les bases de connaissances, restent à explorer dans de nombreux domaines. Nos derniers travaux concernant la robotique humanoïde et les jeux stochastiques sont présentés en annexes. Nos derniers projets, toujours en cours, concernent la robotique aérienne, la robotique médicale et la parallélisation des algorithmes MCTS.

6.2 Perspectives

Les perspectives de nos travaux de recherche présentent des algorithmes protéiformes : partant de description de problèmes, de description de jeux, de modèles géométriques, cinématiques ou dynamiques, les algorithmes impliquent l'utilisation de modèles différents. Nous avons étudié des algorithmes séquentiels, parallèles et distribués. La coordination des algorithmes distribués est réalisée par heuristique, par communication stigmergique et par échange de messages dirigés. Nous avons appliqué des modèles descriptifs, cinématiques, dynamiques et flous. En utilisant des procédures automatiques, nous avons généré des modèles paramétriques. Nous avons utilisé des méthodes d'échantillonnage, des méthodes heuristiques, des méthodes optimales et des méthodes évolutionnaires. Hors ligne, nous avons calculé des tables permettant d'obtenir des résolutions complètes optimales. Une perspective est de tendre vers l'unification de ces algorithmes à l'aide de représentations internes évoluant avec les itérations. A l'avenir, nous pensons augmenter l'homogénéité des modèles initiaux, les transformer en modèles paramétriques par des procédures automatiques, établir des algorithmes adéquats à ces modèles paramétriques et les faire évoluer pour transformer leur structure et présenter de meilleures solutions.

6.2.1 Robotique

Nous présentons ici nos perspectives en robotique articulée, mobile, aérienne et médicale.

Robotique articulée

Nos premiers résultats [122, 119] sur l'optimisation morphologique nous permettent de prévoir des évolutions intéressantes de généralisation des modèles de robotique articulée. Notre proposition d'une procédure de génération automatique des paramètres de modélisation géométrique des chaînes cinématiques [95] peuvent s'appliquer plus généralement à tout type de chaîne articulée. Ainsi nous pensons intéressant d'étudier les modes de déplacements des systèmes articulés avec 1 et 3 pattes. Avec des modèles cinématiques, nous prévoyons d'étudier des modes de déplacements avec un nombre de points de contact variable dans le temps. Avec des modèles dynamiques, nous prévoyons d'étudier des modes de déplacements avec des phases sans appui au sol. Ces différents cas d'études pourront nous permettre de comparer les modes de déplacements, dans leurs espaces de travail, leurs consommations et les mobilités résultantes. Pour reprendre les propositions d'intelligence du corps de R. Pfeifer et F. Iida [183], nous étudierons

la dépendance entre point de contact au sol et morphologie des robots articulés. L'espace atteignable par les pieds variant en fonction de la morphologie des robots, nous pensons étudier les possibilités de minimisation de la consommation d'énergie des modèles articulés tout en maximisant les possibilités d'action des mouvements. Ces travaux nous permettront d'étudier différents modes de déplacements, mettant l'accent sur la rapidité ou sur la sécurité des déplacements des robots dans les espaces encombrés. Une autre piste intéressante de recherche concerne la coopération de chaîne articulée pour la réalisation de mouvements coordonnés dans une optique de réduction globale de l'énergie consommée.

Robotique mobile

Avec un nombre de plateformes disponibles conséquent, la robotique mobile évolue ces dernières années vers les questions de coopération et de coordination multi-robots. Dans ce contexte, une extension de nos travaux sur les mécanismes d'allocation de tâches distribuées [238] serait intéressante pour les architectures multi-robots hétérogènes. Dans le cas de missions difficiles [1], il est admis qu'un unique robot ne possède que peu de chance de pouvoir accomplir une mission avec succès. Ce constat souligne l'importance des missions multi-robots, dans lesquelles la composition des compétences des robots impliqués est déterminante pour la mission. Plusieurs scénarios sont possibles, d'un déploiement entièrement distribué composé de comportements complémentaires [238] vers une solution centralisée optimale [128], l'identification d'action-clés pondérées sur les capacités du groupe et sur la nécessité de l'action permettrait de proposer des solutions adaptatives intéressantes.

Robotique aérienne

La profusion récente des plateformes de *drones* et les nouvelles possibilités offertes par les processeurs multi-cœurs basse consommation nous poussent vers l'étude des possibilités de suivi et de supervision automatique de l'agriculture dans les régions à couverture satellite réduite ou coûteuse. Dans ce contexte, nous avons initié un projet en collaboration avec l'université *Jomo Kenyatta University of Agriculture and Technology* de Nairobi au Kenya sur ce sujet. L'objectif est de prendre en compte l'état d'avancement des cultures pour permettre une meilleure gestion des cultures par les agriculteurs et de permettre une meilleure supervision des surproductions et des sous-productions afin de prévenir des manques en approvisionnement des denrées produites. Les drones permettront de prendre part à la collecte des données temps-réel dans des missions d'inspections des terres

1. telles que l'assistance à l'action dans les cas de catastrophes nucléaires.

agricoles. Dans la suite de nos travaux en robotique mobile sur les systèmes MMRS, nous pensons appliquer des méthodes probabilistes pour la création de missions adaptées aux résultats passés et présents, des missions plus courtes ou plus spécifiques basées sur des modèles d'échantillonnage probabiliste. Dans des travaux plus récents établis en collaboration avec des chercheurs du *Laboratoire des Sciences et des Technologies de l'Information et de la Communication* de l'Université de Guelma en Algérie et du *Dipartimento di Informatica, Elettronica e Sistemistica* (DIES) de *Università della Calabria* en Italie, nous avons également proposé des solutions de coordination multi-agents sans message [247] réalisées en simulation dans des expériences de couverture et de recherche multi-cibles [2]. Nous avons proposé un nouvel algorithme inspiré de la dynamique des vortex d'eau pour donner une dynamique de groupe en fonction des informations locales permettant à chaque agent de définir ses prochains déplacements sans communication directe avec les autres agents. Les résultats obtenus montrent la robustesse de notre algorithme face à l'encombrement de l'espace et les relations entre le nombre d'agents et la taille de l'espace de recherche. Nous pensons évaluer les possibilités d'extension de ce concept à la collecte des informations dans le cadre du projet de suivi de l'agriculture à l'aide de plateformes de *drones*.

Robotique médicale

L'avènement récent des technologies d'impression 3D dans des applications courantes a facilité les possibilités d'études de nouveaux *design* en robotique humanoïde [3]. En ce sens, les travaux précurseurs de R. Pfeifer et F. Iida [183] [4] soulignent les contraintes posées par la morphologie et les limitations de l'intelligence possible d'un corps induites par sa forme. Simple à produire, facile à reproduire en intégralité ou par parties, robuste, facilitant les expérimentations, diminuant les risques au cours d'interaction physique avec les robots, de nombreuses plateformes humanoïdes artisanales ont été réalisées au cours des 10 dernières années. Dans le cadre de la *RoboCup*, l'équipe *NimbRo* [167] propose en 2004 de construire des robots humanoïdes, au rythme de 1 par an sur une période de 5 ans et d'étudier les possibilités d'apprentissage de robots humanoïdes (joueurs

2. Les problèmes de fourragement, de couverture et de recherche multi-cibles diffèrent : Le fourragement est l'activité de collecte de nourriture combinant recherche, ramassage et rapatriement vers une base ; La couverture est l'activité d'exploration de l'ensemble d'un espace en un minimum de temps ; La recherche multi-cibles est l'activité de découverte de n cases considérées comme cibles.

3. Pour R. Brooks [27], les progrès les plus importants à venir en robotique humanoïde concernent la résistance et l'adaptabilité, qui sont des capacités intrinsèques au corps humain.

4. Dans leur travaux précurseurs, parlent "d'intelligence du corps" et montrent les liens entre le modèle de robot utilisé et les résultats possibles.

de football ou guides de musée). Au cours des années suivantes, les déclinaisons et modifications de cette plateforme initiale prendront de nombreux noms, dont *Toni*, *Fritz*, *Rudi*, *Max*, *Jupp*, *Sepp*, *Paul*, qui auront pris part à des compétitions de la *RoboCup* dans les catégories *KidSize*, *TeenSize* et dans la catégorie d'assistants à domicile *AtHome*. Avec la plateforme *Poppy*, M. Lapeyre *et al.* [149] ont conçu une plateforme humanoïde permettant d'étudier la morphologie humanoïde, l'ensemble des interactions corporelles d'un humanoïde avec l'humain et les questions de robustesse et de facilitation de la reproduction d'un humanoïde. Leurs travaux présentent une comparaison de 13 plateformes dont *NAO*, *Darwin-Op*, *NimbRo-Op* et *iCub*. Le robot *Poppy* a cependant besoin d'être soutenu pour marcher. Depuis janvier 2012, G. Langevin a initié le projet *InMoov* dont l'objectif est de construire un humanoïde de taille humaine par impression 3D. Avec la plateforme *MU-L8*, A.B. Stroud *et al.* [212] proposent une évolution de la plateforme *NimbRo* entièrement conçue par impression 3D. En collaboration avec le *Laboratoire Energétique Mécanique Electromagnétique* (LEME) de l'Université Paris Ouest (UPO), nous avons débuté en janvier 2014 une étude des solutions existantes de prothèses de main non intrusives et commencé à évaluer les possibilités de prothèse conçue par impression 3D. A l'aide de capteurs de surface non intrusifs placés sur l'avant bras et sur les muscles dorsaux, l'objectif de ce projet est double : définir des traitements pour le contrôle d'une prothèse de main et étudier la commande d'une prothèse par muscle artificiel. Un premier prototype de pince à 2 doigts a été réalisé. Commandé par un ensemble de modules *ROS*, les premiers tests nous ont permis d'évaluer les capacités de perception [5] d'un capteur de pression *FlexiForce*. En partenariat avec des biomécaniciens et des ostéopathes, nous avons prévu pour l'année à venir d'adapter le mécanisme de préhension et de réaliser un prototype de main complet.

6.2.2 Jeux

Nous présentons ici nos perspectives de recherche dans le domaine des jeux abstraits, sur la parallélisation des algorithmes, sur les jeux non déterministes, sur l'Analyse Retrograde et sur les jeux généraux.

Parallélisation et complexité

Précédemment exclusivement réservée aux supercalculteurs, la parallélisation des algorithmes s'applique aujourd'hui sur une grande variété de plateformes, des

5. Principalement le domaine de définition des valeurs, la précision, la linéarité et l'hystérésis pour évaluer l'adéquation du capteur avec l'asservissement de la main pour des actions de préhension.

ordinateurs classiques, aux cartes GPU et à des architectures parallèles spécialisées HPC [6]. La parallélisation des algorithmes de parcours d'arbre ou de recherche dans un espace permet de réduire linéairement les temps de calcul et de trouver de nouvelles solutions. En parallélisant les algorithmes MCTS pour le *Jeu de Go*, les algorithmes PN et les algorithmes *Nested*-MCTS, nous avons linéairement augmenté le nombre de simulations réalisées, permettant ainsi d'obtenir des programmes de meilleur niveau. Une poursuite logique de nos travaux précédents concerne la parallélisation de ces algorithmes dans les cartes GPU [7] et des nouveaux processeurs MPPA [8], qui nécessite une réécriture en tout ou partie [9] de ces algorithmes. Nous pensons également étendre nos recherches en parallélisation des algorithmes à des problèmes combinatoires tels que le calcul des *Weak Shur Numbers*, l'affectation quadratique et des problèmes de type *puzzle* tels que *Sokoban* et *Twixt*.

Dans le cadre d'un contrat de collaboration recherche avec un constructeur d'ordinateurs MPPA, nous avons initié des travaux de recherche sur la parallélisation des algorithmes MCTS sur MPPA de 256 cœurs. Ce contrat fait suite à des titres obtenus dans les compétitions internationales [10] par parallélisation des algorithmes Monte Carlo. La compétition de GGP [11] a pour objectif d'étendre les résultats obtenus dans les jeux spécialisés à des jeux plus généraux et de fait à

6. *High Performance Computing*.

7. L'évolution des rapports de puissance, de mémoire et de transfert dans les architectures de calcul implique aujourd'hui un parallélisme plus explicite dans les programmes : le nombre de transistors possibles dépasse le nombre de transistors utiles ; le temps de lecture et d'écriture en mémoire dépasse le temps des opérations arithmétiques ; la répartition d'instructions indépendantes supplémentaires augmente le temps global de calcul. Les architectures CPU sont construites sur l'exploitation de peu d'unités de calcul capables de réutiliser des résultats intermédiaires. Les architectures GPU reposent sur l'exploitation d'un grand nombre d'unités de calcul associées à des flots de données distincts.

8. Les architectures Massively Parallel Processor Array regroupent un grand nombre d'unités de calcul reliées par un réseau d'interconnexion (RI) reconfigurable. Par exploitation de ce RI, les MPPA sont particulièrement adaptés aux traitements de séquences de flots de données ayant une largeur bornée et avec des dépendances inter-séquences clairement établies avant le début du calcul. Pour des flots à largeur non bornée ou sans dépendance inter-séquences pré-établie, la parallélisation sur MPPA devient un problème de division hiérarchique en séquences et de reconfiguration du RI.

9. L'exploitation de symétrie de répartition des données sur les grilles d'unités de calcul GPU permettra d'obtenir des gains intéressants et de tirer partie du nombre d'éléments de calcul impliqués ; ce qui nécessite réécriture.

10. Golois est un programme de *Jeu de Go* dont les auteurs sont T. Cazenave et N. Jouandeau. Golois a remporté de nombreuses médailles d'or dans la catégorie *phantom-go* de la compétition *Computer Olympiad*. Ary est un programme de GGP dont les auteurs sont J. Méhat et T. Cazenave. Ary a remporté 2 fois la compétition de GGP.

11. *General Game Playing.*

des problèmes plus généraux [12]. Pour envisager la parallélisation des algorithmes MCTS sur MPPA, il a été nécessaire de réduire l'espace-mémoire utilisé pour les boucles d'interprétation GDL [13]. Les premiers tests de parallélisation sur un algorithme de maximum de produit-matriciel montrent que l'architecture nécessite plus de cœurs [14] pour obtenir des facteurs d'accélération supérieurs aux architectures multi-cœurs classiques.

Problèmes non-déterministes

Dans une vision plus large de la résolution de problèmes, une extension de nos travaux sur les jeux à information complète concerne les jeux non-déterministes. Les problèmes sont des situations plus réelles avec une connaissance complète du problème et une part de hasard dans les états suivants. Les adversaires sont capables de réponses imprévisibles. Les solutions proposées sortent d'un cadre idéal de résolution. Le jeu de *Backgammon* est un bon exemple de jeu non-déterministe à 2 joueurs. A chaque tour, les joueurs lancent des dés et déplacent des pions en fonction des résultats du tirage de dés. Nous avons initié des travaux de recherche sur l'utilisation des algorithmes MCTS pour ce type de problème. Avec un espace de 10^{37} états, pour un facteur de branchement moyen de 32 et un arbre complet de 10^{135} nœuds, il serait intéressant d'évaluer l'évolution de la complexité en fonction des incertitudes. Nos premiers essais MCTS montrent des résultats inférieurs à l'approche classique *-minimax. En comparant des variantes de regroupement de nœuds MCTS, nous avons établi [117] les gains de *playout* sous-contraints et d'évaluations heuristiques. Dans des travaux très récents, nous avons présenté une évaluation des différentes politiques possibles de regroupement des nœuds et leurs performances sans fonction d'évaluation heuristique [116]. Il serait intéressant d'évaluer le comportement des algorithmes MCTS sur les différentes variantes que nous avons proposées [111]. Dans la poursuite de nos travaux sur les bases de finales [194], il serait intéressant de construire des bases d'ouverture pour les problèmes non-déterministes.

12. Pour J. Pitrat, la résolution de problèmes généraux est une composante essentielle de l'intelligence [187].

13. *Game Description Language*.

14. Les prévisions des constructeurs annoncent des processeurs MPPA à plus de 1024 cœurs à partir de 2015. Nous sommes très enthousiastes sur les futures possibilités de calcul scientifique offertes par ces architectures ayant des régimes de consommation inférieurs aux architectures multi-cœurs actuelles. Un processeur MPPA à 256 cœurs consomme moins de 10[W] quand son équivalent multi-cœurs Intel Core i7 consomme 130[W].

6.2.3 Informatique médicale

L'informatique médicale est un domaine en pleine expansion. La collecte des données et leur traitement massif permet l'élaboration de nouvelles pistes de recherche. L'imagerie rétinienne et les questions de segmentation pour l'aide au diagnostic sont amplement traitées dans la littérature et possèdent des solutions variées [74] notamment par apprentissage, par modélisation experte ou encore par morphologie mathématique. Parmi les solutions existantes, les solutions non supervisées ayant été très peu proposées, nous avons proposé [129] une méthode probabiliste non supervisée pour la segmentation des images de fond de l'œil permettant une aide au diagnostic médical dans le cadre d'une collaboration internationale [15] avec le laboratoire *Computer Vision and Virtual Reality Technology* de l'université *Central South University* de Changsha en Chine. Il serait intéressant d'éprouver les résultats préliminaires expérimentaux dans de nouvelles expérimentations comparatives en utilisant des *benchmarks* reconnus ou dans des plateformes collaboratives de compétitions internationales [165] [16]. Les paramètres de seuillage adaptatif pourraient également être établis en ligne, en relation avec une évaluation du bruitage des images à traiter. Dans les chapitres précédents, nous avons présenté nos contributions concernant l'évolution des idées et des connaissances sur les algorithmes de type Monte Carlo. Naturellement intéressés par les problèmes impliquant un grand nombre de données, nous pensons les appliquer à l'informatique médicale. Les premiers résultats obtenus en segmentation des images de fond de l'œil nous permettent de penser à des extensions dans le domaine plus général de la segmentation du corps humain.

Une extension intéressante de nos travaux concerne les applications à la modélisation des vaisseaux et à leur extraction à partir de données tomodensitométriques [17]. La robustesse et la rapidité des méthodes probabilistes couplées à des approches temporelles pourraient permettre de produire des modèles d'évolution des maladies et aider à la mise en place de procédures de biopsies spécialisées en fonction des données perçues et des stades des maladies. L'utilisation des méthodes probabilistes pourrait également permettre d'adapter les informations nécessaires à la modélisation et au diagnostic en fonction des maladies, des examens ou des patients et de fait permettre de réduire les temps d'exposition aux traitements ionisants [18].

15. Ces travaux ont été financés par les PHC 28011WC et 30068QH du ministère des Affaires étrangères.

16. `https://www.kaggle.com/competitions`

17. La tomodensitométrie appelée *Computed Tomography* (CT) est une méthode d'imagerie médicale permettant de créer une série d'images. Elle est actuellement utilisée dans le traitement du Cancer pour la détection d'excroissances et de tumeurs.

18. La tomodensitométrie est une procédure d'analyse médicale reconnue comme étant à dose élevée en émission de radiations pour les patients. Bien que constituant 4% des examens médicaux,

6.2.4 Coordination

Nous présentons ici nos perspectives en coordination par stigmergie, en synthèse de stratégies et planification distribuée et sur les interactions *Ad Hoc*.

Coordination par stigmergie

Nos premiers travaux sur les méthodes probabilistes [114], sur la coordination multi-robots avec Z. Yan [127] et sur l'utilisation de la notion de stigmergie avec O. Zedadra [247] nous permettent de prévoir des extensions concernant les systèmes MMRS et l'assistance à la navigation avec les périphériques mobiles. L'imposante part prise par la géolocalisation dans les applications au cours des 10 dernières années nous pousse à étendre les méthodes de navigation des robots mobiles aux périphériques mobiles pour la gestion des mouvements de masse dans les espaces urbains. Nous avons donc initié une réflexion sur la navigation par information de champs-proches (nommée *Near Field Navigation*) pour les architectures multi-robots et pour l'assistance à la navigation. Les méthodes de localisation par champs-proches nous permettent aujourd'hui de prévoir de nouvelles méthodes de navigation basées sur l'exploitation d'informations de proximité entre plusieurs acteurs d'un même système. L'objectif est l'étude des algorithmes de navigation basés sur la notion de proximité dans un ensemble de systèmes autonomes indépendants. Pour un système distribué, composé de plusieurs agents (mobiles et/ou fixes), l'optimisation de la navigation implique la mutualisation d'informations (*i.e.* de données perçues et d'actions initiées) et le développement de formes de coopération. Dans ce contexte, l'utilisation d'une équipe coordonnée de robots est souvent suggérée, le nombre étant un supposé avantage. L'approche multi-agents présente des atouts de robustesse et de rapidité d'exécution, conditionnés par la gestion des communications pour la concertation et la délibération. Cette approche fait également apparaître de nouveaux problèmes, qui sont notamment le raisonnement distribué et la gestion des interférences des différents acteurs dans l'environnement. Ces interférences sont de plusieurs natures. Les interférences d'actions caractérisent les problèmes de gestion de l'espace commun pour la mission, les interférences physiques de perceptions identifient les problèmes de fonctionnement des capteurs (la diaphonie des ultrasons) et les interférences logiques de perceptions identifient les problèmes de délibération collective. Face à cette diversité d'interférences, le raisonnement collectif doit synthétiser les actions à effectuer. En pratique, l'approche par identification de frontières domine la

elle représente selon des études [103] globalement plus de 40% des expositions reçues par les patients. Les question de dose de radiations transmises et d'optimisation des protocoles d'examens associés sont donc deux questions importantes dans les considérations d'impact de ces traitements sur les patients.

résolution du problème de l'exploration des environnements inconnus. L'espace est discrétisé sur une grille, et chaque cellule est considérée comme libre, occupée ou inconnue. L'agent autonome produit une synthèse de sa situation faisant apparaître au fil de l'exploration, la notion de frontières comme limite entre les zones explorées et non explorées. Le gain d'informations escompté guide les déplacements, permettant de maximiser l'information acquise dans le temps. Pour des espaces de grandes dimensions avec des partitions complexes, la notion de frontière devient insuffisante et une planification de plus haut niveau devient nécessaire. Les méthodes probabilistes nous permettront de définir de bons critères de construction de graphes topologiques, le séquençage de ces nœuds pour des foules d'agents non connectés, l'identification de leurs possibles interactions et interférences.

Stratégie et planification distribuée

Suite à nos travaux sur les possibilités de modélisation HFSM des comportements collectifs, incluant des états collectifs [100, 99] dans les compétitions de *RoboCup SPL* et sur la génération de stratégies statiques de coordination de comportements complexes [123] et de stratégies dynamiques de comportements simplifiés [118] dans les compétitions de *RoboCup 3DSSL*, nous pensons étudier les possibilités de mélange des actions individuelles. A l'aide des automates à états finis non-déterministes (*i.e. Nondeterministic Finite State Machines* (NFSs) [219]) nous pensons générer des structures fédératives, créant des automates à plusieurs états actifs au même instant. Ces nouveaux états seront des compositions des états possibles des différents robots. Les décisions étant réalisées en autonomie, chaque robot devra éliminer en-ligne les compositions les moins prometteuses selon une évaluation statistique similaire à nos travaux précédents sur les algorithmes PN [195] et plus récemment dans des variantes MCTS pour les jeux non-déterministes [117].

Interactions *Ad Hoc*

L'augmentation de l'autonomie des robots, des programmes et des joueurs virtuels va de pair avec l'augmentation des possibilités d'interaction *Ad Hoc* des systèmes décisionnels. Pour un robot seul dans un espace, il s'agira d'évaluer les parties de l'environnement qui pourront l'aider à achever sa mission. Pour plusieurs robots dans un espace commun, il s'agira de différencier amis et ennemis, de différencier les comportements collectifs des comportements individuels. Pour un joueur dans un puzzle, il s'agira d'évaluer les indices ou des évidences du problème. Pour plusieurs entités impliquées dans un problème à somme nulle,

il s'agira d'identifier les capacités de coopération des autres entités [19]. Pour plusieurs entités impliquées dans un problème à somme non-nulle, il s'agira d'identifier les comportements des autres pouvant l'aider à marquer plus de points. Dans le contexte appliqué des jeux et de la robotique, il serait intéressant d'étudier la coopération et la compétition sans prédéfinition. Cette question implique une première phase d'évaluation de l'environnement, de l'objectif, de l'autre, des autres et de leur comportements. Pour M. Bowling and P. McCracken [26], le problème est soit prédictif (*i.e.* évaluer toutes les combinaisons des actions individuelles possibles) soit adaptatif (*i.e.* apprendre ce qui marche, pondérer son comportement). Dans le cadre de la *RoboCup 3DSSL*, nous avons participé [160] à des épreuves de coopération *Ad Hoc* sans connaissance à priori des facultés des coéquipiers. Dans des travaux récents, nous avons étudié les mécanismes de réduction dans les arbres MCTS pour les problèmes non-déterministes à information complète [116]. Dans une solution par combinaison des actions individuelles, le regroupement des combinaisons nous permettra de façon similaire de faire face à la combinatoire du problème. Dans une solution adaptative, nous pensons utiliser la logique floue pour comprendre les comportements des autres entités. Dans des travaux récents, nous avons étudié les sentiments d'utilisateurs de média sociaux [174] en proposant une classification floue et démontré l'importance d'une catégorie neutre dans les mécanismes d'évaluation. Des travaux récents sur le traitement des données obtenues dans les jeux tentent de comprendre les joueurs, de synthétiser les comportements des joueurs [66] par clustering ou d'identifier des styles de joueurs non conformes [1]. Dans une approche plus générale de qualification des comportements et des interactions, nous pensons que la logique floue pourrait permettre d'établir une classification par adéquation entre situation et action (*i.e.* requête et réponse) en généralisant nos travaux préliminaires [175] sur une mesure d'évaluation universelle des contenus multimédia par classification floue.

19. Et réciproquement, l'identification par les autres de nos propres capacités de coopération.

Annexe A

Participations et résultats obtenus en compétitions internationales

- 2014
  - Joao Pessao, *RoboCup*, catégorie *3D Soccer Simulation League*, 9ème à 12ème place.
  - Joao Pessao, *RoboCup*, *technical challenge*, catégorie *2D/3D Simulation*, 6ème place.
  - Magdeburg, *German Open*, catégorie *3D Soccer Simulation League*, 5ème place.
  - Taipei, *Taiwan Computer Game Tournament*, catégorie *Dark Chinese-Chess*, 9ème place.
- 2013
  - Eindoven, *RoboCup*, catégorie *3D Soccer Simulation League*, 2ème tour, 12ème à 13ème place.
  - Yokohama, *Computer Olympiad*, catégorie *Phantom-Go*, médaille d'or.
  - Yokohama, *Computer Olympiad*, catégorie *Dark-Chess*, 7ème place.
  - Magdeburg, *German Open*, catégorie *Standard Plateform League*, 8ème à 16ème place.
  - Taipei, *Taiwan Computer Game Tournament*, catégorie *Dark-Chess*, 7ème place.
- 2012
  - Mexico, *RoboCup*, catégorie *Standard Plateform League*, tour intermediaire, 16ème à 24ème place.
  - Mexico, *RoboCup*, catégorie *Standard 3D Soccer Simulation League*, 1er tour, 12ème à 13ème place.
  - Eindoven, *Dutch Open*, catégorie *3D Soccer Simulation League*, 4ème place.
  - Magdeburg, *German Open*, catégorie *Standard Plateform League*, 8ème à

16ème place.
- 2011
  - Tilburg, *Computer Olympiad*, catégorie *Phantom-Go*, médaille d'or.
  - Tilburg, *Computer Olympiad*, catégorie *Hex*, 3ème place.
  - Istambul, *RoboCup*, catégorie *Standard Plateform League*, tour intermediaire, 17ème à 25ème place.
  - Istambul, *RoboCup*, catégorie *3D Soccer Simulation League*, 21ème à 22ème place.
  - Magdeburg, *German Open*, catégorie *Standard Plateform League*, 8ème à 16ème place.
- 2010
  - Kanazawa, *Computer Olympiad*, catégorie *Phantom-Go*, médaille d'or.
  - Singapore, *RoboCup*, catégorie *Standard Plateform League*, 3ème place du 2ème tour, 9ème à 12ème place.
  - Rome, *Mediterranean Open*, catégorie *Standard Plateform League*, 3ème à 4ème place.
  - Magdeburg, *German Open*, catégorie *Standard Plateform League*, 5ème à 8ème place.
- 2009
  - Pamplona, *Computer Olympiad*, catégorie *Phantom-Go*, médaille d'or.
  - Graaz, *RoboCup*, catégorie *Standard Plateform League*, 3ème place du 2ème tour, 9ème à 12ème place.
  - Hannover, *German Open*, catégorie *Standard Plateform League*, 8ème à 16ème place.
- 2008
  - Beijing, *Computer Olympiad*, catégorie *Phantom-Go*, médaille d'or.
  - Beijing, *Computer Olympiad*, catégorie *Go9x9*, 13ème place.
- 2007
  - Amsterdam, *Computer Olympiad*, catégorie *Phantom-Go*, médaille d'or.
  - Amsterdam, *Computer Olympiad*, catégorie *Go9x9*, 5ème place.

Bibliographie

[1] M.A. Ahmad, B. Keegan, J. Srivastava, D. Williams, and N. Contractor. Mining for gold farmers : Automatic detection of deviant players in mmogs. In *IEEE Int. Conf. on Computational Science and Engineering, pp. 340–345, 2009.* 90

[2] J.M. Ahuactzin-Larios. *Le Fil d'Ariane : Une Méthode de Planification Générale. Application à la Planification Automatique de Trajectoires*. PhD thesis, Inst. Nat. Polytechnique de Grenoble, 1994. 12

[3] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. Multi-robot Cooperation in the MARTHA Project. *IEEE Robotics and Automation Magazine*, 5(1) :36–47, 1998. 51

[4] D. Mc Allester. A New Procedure for Growing Minimax Trees. In *Technical Report, Artificial Intelligence Laboratory, MIT, 1985.* 72

[5] L.V. Allis. Searching for Solutions in Games an Artificial Intelligence. In *Phd thesis, Vrije Universitat Amsterdam, Maastricht, 1994.* 73

[6] L.V. Allis, M. van der Meulen, and H.J. van den Herik. Proof-Number Search. In *Artificial Intelligence, 66(1) :91–124, 1994.* 72

[7] N.M. Amato, O.B. Bayazit, L.K. Dale, C. Jones, and D. Vallejo. OBPRM : An Obstacle-Based PRM for 3D Workspaces. In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 155–168, Houston, TX, USA, 1998. 61

[8] N.M. Amato, O. Bayazita, L. Dale, C. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilisitc roadmap methods. In *Int. Conf. Robotics and Automation (ICRA), 1998.* 12

[9] N.M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *Int. Conf. Robotics and Automation (ICRA), 1996.* 12

[10] T. Arai, E. Pagello, and L.E. Parker. Advances in multi-robot systems. In *IEEE Trans. on Robotics and Automation, 18(5) :655–661, 2002.* 45

[11] A.M. Arsenio. Object Recognition from Multiple Percepts. In *IEEE-RAS/RSJ Int. Conf. on Humanoid Robots, 2004.* 36

[12] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. In *Machine Learning, 47, pp. 235–256, 2002.* 22

[13] S. Balakirsky, S. Carpin, A. Kleiner, M. Lewis, A. Visser, J. Wang, and V. Amos Ziparo. Towards heterogeneous robot teams for disaster mitigation : Results and performance metrics from robocup rescue. In *Journal of Field Robotics, 24(11–12) : 943–967, 2007.* 49

[14] J. Barraquand. Automatic Motion Planning for Complex Articulated Bodies. Technical report, Laboratoire de Robotique de Paris (LRP), 1991. 9, 20

[15] J. Barraquand and J.C. Latombe. Nonholonomic multibody mobile robots : Controllability and motion planning in the presence of obstacles. In *Algorithmica, 1993.* 11

[16] J. Barraquand and J.C. Latombe. Robot motion planning : A distributed representation approach. In *Int. Journal of Robotics Research (IJRR), 1991.* 9, 20

[17] M.R. Barrios, M.A. Kalem, Y. Kosuge, P. Lamarche, B. Schmolke, G. Sohos, J.J. Vasquez, and T. Wherle. Inside the RS/6000 SP. In *IBM Int. Technical Support Organization, SG24-5145-00, 1998.* 8

[18] A. Bautin, O. Simonin, and F. Charpillet. Stratégie d'exploration multi-robot fondée sur les champs de potentiels artificiels. In *Journée Francophones sur les Systèmes Multi-Agents (JFSMA), 2011.* 50

[19] S. Behnke. Online trajectory generation for omnidirectional biped walking. In *Int. Conf. on Robotics and Automation (ICRA'06), pp. 1597–1603, 2006.* 32

[20] G. Beni and J. Wang. Swarm Intelligence in Cellular Robotic Systems. In *NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, 1989.* 50

[21] H. Berliner. The B* Tree Search Algorithm. A Best-First Proof Procedure. In *Artificial Intelligence Vol. 12, Issue 1, pp. 23–40, 1979.* 72

[22] J. Bialkowski, S. Karaman, and E. Frazzoli. Massively Parallelizing the RRT and the RRT$*$. In *Int. Conf. on Intelligent Robots and Systems (IROS), 2011.* 8

[23] D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. The challenge of poker. In *Artificial Intelligence, Vol. 134, pp. 201–240, 2002.* 20

[24] E. Bonabeau, M. Dorigo, and G. Theraulaz. Swarm Intelligence - From Natural to Artificial Systems. In *Oxford University. Press, USA, 1999.* 50

[25] B. Bouzy and B. Helmstetter. Monte-carlo go developments. In *ACG, Vol. 263 of IFIP, pp. 159–174. Kluwer, 2003.* 20

[26] M. Bowling and P. McCracken. Coordination and adaptation in impromptu teams. In *Multi-Agent Systems, Robotics, AAAI, 2005.* 90

[27] R.A. Brooks. The next 50 years. In *Commun. ACM 51(1) : 63-64, 2008.* 83

[28] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A Survey of Monte Carlo Tree Search Methods. In *IEEE Trans. on Computational Intelligence and AI in Games, Vol. 4, no. 1, 2012.* 2, 23

[29] B. Bruegmann. Monte carlo go. In *Unpublished, 1993.* 20

[30] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative Multi-Robot Exploration. In *Int. Conf. on Robotics and Automation (ICRA), 2000.* 46

[31] A. De Cabrol, P. Bonnin, T. Costis, V. Hugel, and P. Blazevic. A New Video Rate Region Color Segmentation and Classification for Sony Legged RoboCup Application. In *LNCS, RoboCup 2005 : Robot Soccer World Cup IX., Springer-Verlag, 2005.* 37

[32] J.F. Canny. *The complexity of robot motion planning*. PhD thesis, Massachusetts Institute of Technology. Artificial Intelligence Laboratory, 1987. 8

[33] Y.U. Cao, A.S. Fukunaga, and A.B. Kahng. Cooperative mobile robotics : Antecedents and directions. In *Autonomous Robots, 4(1) :7–27, 1997.* 45

[34] S. Carpin and E. Pagello. On Parallel RRTs for Multi-robot Systems. In *8th Conf. of the Italian Association for Artificial Intelligence (AI*IA), 2002.* 2, 8, 11

[35] T. Cazenave. Automatic Acquisition of Tactical Go Rules. In *Game Programming Workshop in Japan '96, pp. 10–19, Kanagawa, Japan, 1996*. 77

[36] T. Cazenave. Generation of Patterns With External Conditions for the Game of Go. In *H.J. van den Herik and B. Monien, editors, Advances in Computer Games 9, pp. 275–293. Universiteit Maastricht, Maastricht, 2001.* 77, 78

[37] T. Cazenave. Nested Monte-Carlo search. In *19th Int. joint Conf. on Artificial Intelligence (IJCAI), 2009.* 69

[38] T. Cazenave and B. Helmstetter. Combining tactical search and monte-carlo in the game of go. In *IEEE Conf. on Computational Intelligence and Games, pp. 171–175, 2005.* 20

[39] T. Cazenave and N. Jouandeau. A Parallel Monte-Carlo Tree Search Algorithm. In *Int. Conf. on Computers and Games (CG), 2008.* 2, 23

[40] T. Cazenave and N. Jouandeau. On the Parallelization of UCT. In *Computer Games Workshop (CGW), 2007.* 2, 23

[41] T. Cazenave and N. Jouandeau. Parallel Nested Monte-Carlo Search. In *12th Int. Workshop on Nature Inspired Distributed Computing (NIDISC), 2009.* 69

[42] T. Cazenave and N. Jouandeau. Towards deadlock free Sokoban. In *Board Game Studies XIIIth Colloquium (BGA), 2010.* 5, 71

[43] G. Chaslot, J.T. Saito, B. Bouzy, J.W.H.M. Uiterwijk, and H. Jaap van den Herik. Monte-carlo strategies for computer go. In *18th BeNeLux Conf. on Articial Intelligence, Namur, Belgium, pp. 83–91, 2006.* 2, 20, 21

[44] G. Chaslot, M.H.M. Winands, H.J. van den Herik, J.W.H.M. Uiterwijk, and B. Bouzy. Progressive Strategies for Monte-Carlo Tree Search. In *New Math. Nat. Comput., Vol. 4, no. 3, pp. 343–357, 2008.* 21

[45] P. Cheng. *Reducing RRT metric sensitivity for motion planning with differential constraints*. PhD thesis, Iowa State University, 2001. 10

[46] P. Cheng and S.M. LaValle. Reducing Metric Sensitivity in Randomized Trajectory Design. In *Int. Conf. on Intelligent Robots and Systems (IROS), 2001.* 10

[47] P. Cheng and S.M. LaValle. Resolution Complete Rapidly-Exploring Random Trees. In *Int. Conf. on Intelligent Robots and Systems (IROS), 2002.* 11

[48] P. Cheng, Z. Shen, and S.M. LaValle. RRT-Based Trajectory Design. In *Archives of Control Sciences, Vol. 11, 2001.* 10

[49] A. Ali Cherif. Collective behavior for a micro-colony of robots. In *2nd European Conf. on Artificial Life (ECAL), 1993.* 45

[50] A. Ali Cherif. A constitution and an economic model for the organisation and emergence of collective behaviour in a colony of robots. In *From Perception to Action conference, pp. 334–337, Lausanne, Switzerland, 1994.* 45

[51] P. Ciancarini and G. Favini. Solving kriegspiel endings with brute force : the case of kr vs. k. In *Advances in Computer Games, pp. 136–145, 2010.* 78

[52] C.M. Clark. Probabilistic Road Map Sampling Strategies for Multi-Robot Motion Planning. *Robotics and Autonomous Systems*, 53(3–4) :244–264, 2005. 57

[53] R. Coulom. CLOP : Confident Local Optimization for Noisy Black-Box Parameter Tuning. In *Advanced Computer Games (ACG), Vol. 7168, pp. 146–157, LNCS, Springer, 2011.* 40

[54] R. Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In *5th Int. Conf. Comput. and Games, Turin, Italy, pp. 72–83, 2006*. 2, 20, 21

[55] J. C. Culberson. Sokoban is PSPACE-complete. In *L. Pagli, E. Lodi and N. Santoro Editors, Conf. on Fun With Algorithms, pp. 65–76, Waterloo, Carleton Scientific, 1999.* 70

[56] J. C. Culberson and J. Schaeffer. Pattern Databases. In *Computational Intelligence, 4(14), pp. 318–334, 1998*. 71

[57] T.S. Dahl, M.J. Matarić, and G.S. Sukhatme. Multi-robot task allocation through vacancy chain scheduling. *Robotics and Autonomous Systems*, 57(6–7) :674–687, 2009. 52

[58] E. D. Demaine, M. L. Demaine, A. Langerman, and S. Langerman. Morpion solitaire. In *Theory Comput. Syst., 39(3) :439–453, 2006.* 68, 69

[59] J. Denavit and R.S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. In *Trans. ASME J. Appl. Mech. 23 pp. 215–221, 1955.* 34

[60] J.L. Denebourg, J.M. Pasteels, and J.C. Verhaeghe. Probabilistic Behaviour in Ants : a Strategy of Errors ? In *Journal of Theoretical Biology, num. 105, 1983.* 47

[61] J.L. Deneubourg, R. Beckers, and O.E. Holland. From local actions to global tasks : Stigmergy and collective robotics. In *4th Int. Workshop on the Synthesis and Simulation of Living Systems, pp. 181–189, Cambridge, MA, USA, 1994.* 45

[62] D. Devaurs, T. Simeon, and J. Cortes. Parallelizing RRT on Distributed-Memory Architectures. In *Int. Conf. Robotics and Automation (ICRA), 2011.* 8

[63] B.R. Donald, P.G. Xavier, J.F. Canny, and J.H. Reif. Kinodynamic Motion Planning. In *Journal of the ACM, 1993*. 10

[64] M. Dorigo and M. Birattari. Swarm Intelligence. In *Scholarpedia, 2007.* 49, 50

[65] M. Dorigo, D. Floreano, L.M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, D. Burnier, A. Campo, A.L. Christensen, A. Decugniere, G. Di Caro, F. Ducatelle, E. Ferrante, A. Forster, J.M. Gonzales, J. Guzzi, V. Longchamp, S. Magnenat, N. Mathews, M. Montes de Oca, R. O'Grady, C. Pinciroli, G. Pini, P. Retornaz, J. Roberts, V. Sperati, T. Stirling, A. Stranieri, T. Stutzle, V. Trianni, E. Tuci, A.E. Turgut, , and F. Vaussard. Swarmanoid : a novel concept for the study of heterogeneous robotic swarms. In *IEEE Robotics & Automation Magazine, 2013.* 50

[66] A. Drachen, R. Sifa, C. Bauckhage, and C. Thurau. Guns, swords and data : Clustering of player behavior in computer games in the wild. In *IEEE Conf. on Computational Intelligence and Games, pp. 163–170, 2012.* 90

[67] G. Dudek, M.R.M. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multiagent robotics. In *Autonomous Robots, 3(4) :375–397, 1996.* 45

[68] H.R. Fang, T.S. Hsu, and S.C. Hsu. Construction of chinese chess endgame databases by retrograde analysis. In *Computers and Games, pp. 96–114, 2001.* 78

[69] A. Farchy, S. Barrett, P. MacAlpine, and P. Stone. Humanoid Robots Learning to Walk Faster : From the Real World to Simulation and Back. In *Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS), 2013.* 40

[70] A. Felner, R. E. Korf, R. Meshulam, and R. C. Holte. Compressed pattern databases. In *Artif. Intell. Res. (JAIR), Vol. 30, pp. 213–247, 2007.* 71

[71] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A Probabilistic Approach to Collaborative Multi-Robot Localization. In *Autonomous Robots, 8(3) :325–344, 2000.* 46

[72] T. Fraichard. Dynamic trajectory planning with dynamic constraints : a "state-time space" approach. In *Int. Conf. Robotics and Automation (ICRA), 1993.* 10

[73] Th. Fraichard and H. Asama. Inevitable collision states - a step towards safer robots ? In *Advanced Robotics, pp. 1001–1024, 2004.* 11

[74] M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A. Rudnicka, C. Owen, and S. Barman. Blood vessel segmentation methodologies in retinal images - a survey. In *Computer Methods and Programs in Biomedicine, Vol. 108, no. 1, pp. 407–433, 2012.* 87

[75] E. Frazzoli, M.A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. In *IEEE Trans. on Robotics, 21(6) :1077–1091, 2005.* 11

[76] E. Gabriel, G.E. Fagg, G. Bosilca, T. Angskun, J.J. Dongarra, J.M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R.H. Castain, D.J. Daniel, R.L. Graham, and T.S. Woodall. Open MPI : Goals, concept, and design of a next generation mpi implementation. In *11th European PVM/MPI Users' Group Meeting, pp. 97–104, Budapest, Hungary, 2004.* 70

[77] S. Gächtero. Results on Range Image Segmentation for Service Robots. In *Laboratoire de Systèmes Autonomes (LSA), Ecole Polytechnique Fédérale de Lausanne (EPFL), EFPL-LSA-2005-01, 2005.* 37

[78] S. Gelly, Y. Wang, R. Munos, and O. Teytaud. Modification of uct with patterns in monte-carlo go. In *Rapport de recherche INRIA RR-6062, 2006.* 22

[79] M. Gendreau and J.Y. Potvin. Handbook of metaheuristics. In *Int. Series in Operations Research, Management Science, Vol. 146, 2nd ed. 2010.* 49

[80] B.P. Gerkey and M.J. Matarić. Pusher-watcher : An approach to fault-tolerant tightly-coupled robot coordination. In *Int. Conf. on Robotics and Automation (ICRA), 2002.* 47

[81] B.P. Gerkey and M.J. Matarić. Sold ! : Auction methods for multi-robot coordination. In *IEEE Trans. on Robotics and Automation, 18(5), pp. 758–768, 2002.* 49

[82] C. Graf and T. Röefer. A Center of Mass Observing 3D-LIPM Gait for the RoboCup Standard Platform League Humanoid. In *Robot Soccer World Cup XV, LNAI, Vol. 7416, Springer, pp. 101–112, 2012.* 32

[83] C. Graf and T. Röefer. A Closed-loop 3D-LIPM Gait for the RoboCup Standard Platform League Humanoid. In *4th Workshop on Humanoid Soccer Robots, IEEE-RAS Int. Conf. on Humanoid Robots, 2010.* 32

[84] F. Guerriero and M. Mancini. Parallelization strategies for rollout algorithms. In *Computational Optimization and Applications, 31(2) :221–244, 2005.* 69

[85] J. Guitton, J.L. Farges, and R. Chatila. Cell-RRT : Decomposing the Environment for Better Plan. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 5776–5781, St. Louis, MO, USA, 2009. 61

[86] N. Hansen. The cma evolution strategy : A tutorial. In `https://www.lri.fr/~hansen/cmatutorial.pdf`, *2011.* 39

[87] P.E. Hart, N.J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. on Systems Science and Cybernetics*, 4(2) :100–107, 1968. 57, 61

[88] M. Hebbel, R. Kosse, and W. Nistico. Modeling and Learning Walking Gaits of Biped Robots. In *1st Workshop on Humanoid Soccer Robots, IEEE-RAS Int. Conf. on Humanoid Robots (HSR), 2006.* 39

[89] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of Honda humanoid robot. In *Int. Conf. on Robotics and Automation, pp. 1321–1326, 1998.* 31

[90] D. Hsu, J.C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Trans. on Robotics and Automation, 1997.* 12

[91] Q. Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi, and K. Tanie. Planning Walking Patterns for a Biped Robot. In *IEEE Trans. on Robotics and Automation, 17(3), 2001.* 31

[92] V. Hugel, G. Amouroux, T. Costis, P. Bonnin, and P. Blazevic. Specifications and Design of Graphical Interface for Hierarchical Finite State Machines. In *LNCS, RoboCup 2005 : Robot Soccer World Cup IX., Springer-Verlag, 2005.* 42

[93] V. Hugel, H. Martinez Barbera, N. Jouandeau, and F. Blanes Noguera. French-Spanish L3M SPL Team Description. In *Robocup Standard Plateform League (RCUP), 2010.* 36

[94] V. Hugel and N. Jouandeau. Analytical Solution for Joint Coupling in NAO Humanoid Hips. In *18th annual RoboCup Int. Symp. 2014 (RCUP-2014).* 4, 34

[95] V. Hugel and N. Jouandeau. Automatic generation of humanoid's geometric model parameters. In *17th annual RoboCup Int. Symp. (RCUP), 2013.* 4, 34, 40, 81

[96] V. Hugel and N. Jouandeau. L3M Joint Team report, Participation in the 2010 Robocup SPL League. In *Annual Team Report, Robocup (RCUP), 2010.* 36

[97] V. Hugel and N. Jouandeau. Les Trois Mousquetaires Team Description. In *Robocup Standard Plateform League (RCUP), 2009.* 36

[98] V. Hugel and N. Jouandeau. Walking Patterns for Real Time Path Planning Simulation of Humanoids. In *21st IEEE Int. Symp. on Robot and Human Interactive Communication (RO-MAN), 2012.* 4, 33

[99] V. Hugel, F. Blanes Noguera, N. Jouandeau, J. Jose Alcaraz Jimenez, and H. Martinez Barbera. SPL Team Description Paper. In *Robocup Standard Plateform League 2012 (RCUP), 2012.* 37, 42, 43, 89

[100] V. Hugel, F. Blanes Noguera, N. Jouandeau, J. Jose Alcaraz Jimenez, H. Martinez Barbera, Z. Y. Bayraktaroglu, and P. Boyraz. SPL Team Description Paper. In *Robocup Standard Plateform League (RCUP), 2011.* 37, 42, 43, 89

[101] G.J. Hwang and S.S. Tseng. A Heuristic Task Assignment Algorithm to Maximize Reliability of a Distributed System. *IEEE Trans. Reliability*, 42(3) :408–415, 1993. 52

[102] H. Hyyrö and T. Poranen. New heuristics for morpion solitaire. Technical report, Department of Computer Sciences, University of Tampere, 2007. 69

[103] ImPACT Group. Estimating patient dose on current CT scanners : Results of the ImPACT* CT dose survey. In `http://www.impactscan.org/dosesurveysummary.htm`, *2000.* 88

[104] M. Jäger and B. Nebel. Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots. In *Int. Conf. on Intelligent Robots and Systems (IROS), 2001.* 47

[105] J.S. Jennings, G. Whelan, and W.F. Evans. Cooperative search and rescue with a team of mobile robots. In *8th Int. Conf. on Advanced Robotics (ICAR), pp. 193–200, Monterey, CA, USA, 1997.* 49

[106] N. Jouandeau. Planification de mouvement. In *4ème Journées Nationales de la Recherche en Robotique, (session poster), (JNRR), 2003.* 13

[107] N. Jouandeau. Planification de mouvement randomisé localisée. In *18ème Journées des Jeunes Chercheurs en Robotique (JCR), 2004.* 13

[108] N. Jouandeau. Planification de trajectoires par visibilité et échantillonnage. In *MAnifestation des JEunes Chercheurs STIC (MAJECSTIC), 2003.* 13

[109] N. Jouandeau. Rapidly Exploring Sorted Random Tree, a self adaptive random motion planning algorithm. In *Lecture Notes in Electrical Engineering, Informatics in Control, Automation and Robotics, Vol. 24, pp. 63-73, 2007.* 18

[110] N. Jouandeau. RSRT : Rapidly Exploring Sorted Random Tree. In *4th Int. Conf. on Informatics in Control, Automation and Robotics (ICINCO), 2007.* 18

[111] N. Jouandeau. Varying Complexity in CHINESE DARK CHESS Stochastic Game. In *Int. Taiwan Computer Game Association Computer Game Workshop (TCGA), 2014.* 6, 86

[112] N. Jouandeau. *Algorithmique de la planification de mouvement probabiliste pour un robot mobile*. PhD thesis, Université Paris 8, 2004. 2, 12, 15, 61

[113] N. Jouandeau and A. AliCherif. An Efficient Environmental Motion Planning Strategy. In *35th Int. Symp. on Robotics (ISR), 2002.* 12

[114] N. Jouandeau and A. AliCherif. Fast Approximation To Gaussian Obstacle Sampling For Randomized Motion Planning. In *5th Symp. on Intelligent Autonomous Vehicles (IAV), 2004.* 2, 13, 88

[115] N. Jouandeau, P. Bonnin, V. Hugel, and P. Blazevic. From Color Groups to Scene Interpretation. In *Workshop of Humanoid Soccer Robots (WHSR), 2009.* 4, 37

[116] N. Jouandeau and T. Cazenave. Monte-Carlo Tree Reductions for Stochastic Games. In *19th Int. Conf. on Technologies and Applications of Artificial Intelligence, Taiwan, Taipei, (TAAI-2014).* 86, 90

[117] N. Jouandeau and T. Cazenave. Small and large MCTS playouts applied to CHINESE DARK CHESS stochastic game. In *21th European Conf. on Artificial Intelligence, Computer Games Workshop (CGW-2014).* 6, 86, 89

[118] N. Jouandeau and V. Hugel. 3DSSL Team Description Paper. In *Robocup 3D Soccer Simulation League (RCUP), 2014.* 35, 42, 43, 89

[119] N. Jouandeau and V. Hugel. Enhancing Humanoids Walking Skills through Morphogenesis Evolution Method. In *4th Int. Conf. on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR 2014).* 4, 41, 81

[120] N. Jouandeau and V. Hugel. L3M-SIM Team Description Paper. In *Robocup 3D Soccer Simulation League 2012 (RCUP), 2012.* 34

[121] N. Jouandeau and V. Hugel. Optimization of Parametrised Kicking Motion for Humanoid Soccer Player. In *IEEE Int. Conf. on Autonomous Robot Systems and Competitions (ICARSC), 2014.* 4, 41

[122] N. Jouandeau and V. Hugel. Simultaneous evolution of leg morphology and walking skills to build the best humanoid walker. In *IEEE-RAS Int. Conf. on Humanoid Robots, 8th Workshop on Humanoid Soccer Robots (HSR), 2013.* 4, 39, 40, 81

[123] N. Jouandeau, V. Hugel, and T. Da Costa. 3DSSL Team Description Paper. In *Robocup 3D Soccer Simulation League (RCUP), 2013.* 35, 42, 43, 89

[124] N. Jouandeau, V. Hugel, and T. Da Costa. From Low Level to High Level Humanoid's Behaviors. In *3DSSL Open Challenge, Robocup 3D Soccer Simulation League (RCUP), 2013.* 42

[125] N. Jouandeau, V. Hugel, and T. Da Costa. Humanoid Soccer Optimization. In *3DSSL Open Challenge, Robocup 3D Soccer Simulation League 2014, (RCUP-2014).* 41

[126] N. Jouandeau, Y. Touati, and A. AliCherif. Incremental Motion Planning With Las Vegas Algorithms. In *New Developments in Robotics Automation and Control, ISBN 978-953-7619-20-6, 2008.* 16

[127] N. Jouandeau and Z. Yan. Decentralized Waypoint-based Multi-robot Coordination. In *IEEE Int. Conf. on Cyber Technology in Automation, Control and Intelligent Systems (CYBER), 2012.* 3, 56, 59, 88

[128] N. Jouandeau and Z. Yan. Improved Trade-based multi-robot coordination. In *6th IEEE Joint Int. Information Technology and Artificial Intelligence Conf., ISBN 978-1-4244-8622-9, (ITAIC), 2011.* 3, 56, 82

[129] N. Jouandeau, Z. Yan, P. Greussay, B. Zou, and Y. Xiang. Retinal Vessel Segmentation Based on Adaptive Random Sampling. In *Journal of Medical and Bioengineering (JOMB, ISSN : 2301-3796), 2014.* 5, 87

[130] A. Junghanns. Pushing the limits : New developments in single-agent search. In *Phd thesis, University of Alberta, 1999.* 70

[131] A. Junghanns and J. Schaeffer. Sokoban : A Challenging Single-Agent Search Problem. In *Workshop on Using Games as an Experimental Testbed for AI Research, IJCAI-97, Nagoya, Japan, August 1997.* 70

[132] A. Junghanns and J. Schaeffer. Sokoban : Enhancing general single-agent search methods using domain knowledge. In *Artif. Intell., 129(1-2) :219–251, 2001.* 70

[133] A. Junghanns and J. Schaeffer. Sokoban : improving the search with relevance cuts. In *Theor. Comput. Sci., 252(1-2) :151–175, 2001.* 70

[134] S. Kajita. Humanoid robot. In *Ohmsha Ltd, 3-1 Kanda Nishikicho, Chiyodaku, Tokyo, Japan, 2005.* 32, 33

[135] L.E. Kavraki. *Random networks in configuration space for fast path planning*. PhD thesis, Stanford University, 1995. 1, 9, 20

[136] L.E. Kavraki, M. Kolountzakis, and J.C. Latombe. Analysis of probabilistic roadmaps for path planning. In *Int. Conf. Robotics and Automation (ICRA), 1996.* 12, 61

[137] L.E. Kavraki, J.C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot path planning. In *Symp. on Theory of Computing, 1995*. 1, 9, 20

[138] W. Khalil and J.F. Kleinfinger. A new geometric notation for open and closed-loop robots. In *IEEE Int. Conf. on Robotics and Automation, pp. 1174–1180, 1986.* 34

[139] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Int. Conf. on Robotics and Automation (ICRA), 1985*. 9

[140] A. Kishimoto, M.H.M. Winands, M. Müller, and J.T. Saito. Game-Tree Search using Proof Numbers : The First Twenty Years. In *ICGA Journal, Vol. 35, No. 3, 2012*. 73

[141] L. Kocsis and C. Szepesvári. Bandit-based monte-carlo planning. In *European Conf. on Machine Learning (ECML), pp. 282–293, 2006.* 2, 20, 21, 23

[142] K. Konolige, C. Ortiz, R. Vincent, A. Agno, M. Eriksen, B. Limketkai, M. Lewis, L. Briesemeister, E. Ruspini, D. Fox, J. Ko, B. Stewart, and L. Guibas. CENTIBOTS, large scale robot teams. Technical report, DARPA SOFTWARE FOR DISTRIBUTED ROBOTICS, 2002. 45

[143] R. E. Korf. Finding optimal solutions to rubik's cube using pattern databases. In *AAAI-97, pp. 700–705, 1997.* 71

[144] R. E. Korf and A. Felner. Disjoint pattern database heuristics. In *Artif. Intell., 134(1-2), pp. 9–22, 2002.* 71

[145] E. Kruse, R. Gutsche, and F. Wahl. Efficient, iterative, sensor based 3-d map building using rating functions in configuration space. In *Int. Conf. Robotics and Automation (ICRA), 1996.* 12

[146] C.R. Kube and E. Bonabeau. Cooperative transport by ants and robots. In *Robotics and Autonomous Systems, 30(1–2) : 85–101, 2000.* 47

[147] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Motion planning for humanoid robots. In *Robotics Research, Springer, 2005.* 31

[148] R. Lake, J. Schaeffer, and P. Lu. Solving large retrograde-analysis problems using a network of workstations. In *H.J. van den Herik, I.S. Herschberg, and J.W.H.M. Uiterwijk, editors, Advances in Computer Chess 7, pp. 135–162. University of Limburg, Maastricht, The Netherlands, 1994.* 67, 77

[149] M. Lapeyre, P. Rouanet, and P.Y. Oudeyer. Poppy Humanoid Platform : Experimental Evaluation of the Role of a Bio-inspired Thigh Shape. In *13th IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids), 2013.* 84

[150] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991. 57, 61

[151] J.P. Laumond, S. Sekhavat, F. Lamiraux, A. Bellaiche, F. Jean, J.J. Risler, P. Souères, J.D. Boissonnat, A. De Luca, G. Oriolo, C. Samson, P. Svestka, M.H. Overmars, P. Jiménez, F. Thomas, and C. Torras. Robot Motion Planning and Control. In *Lectures Notes in Control and Information Sciences. Springer, 1998.* 11

[152] S. LaValle and J. Kuffner. Rapidly-exploring random trees : Progress and prospects. In *Workshop on the Algorithmic Foundations of Robotics (WAFR), 2000.* 19

[153] S.M. Lavalle. Rapidly-exploring random trees : A new tool for path planning. Technical Report 98-11, Dept. of Computer Science, Iowa State University, 1998. 2, 9, 20, 61

[154] S.M. LaValle and J.J. Kuffner. Randomized kinodynamic planning. In *Int. Conf. Robotics and Automation (ICRA), 1999.* 10

[155] S.M. Lavalle and J.J. Kuffner. Randomized kinodynamic planning. In *Int. Journal of Robotics Research (IJRR), 2001*. 12

[156] S.R. Lindemann, A. Yershova, and S.M. LaValle. Incremental Grid Sampling Strategies in Robotics. In *6th Int. Workshop on the Algorithmic Foundations of Robotics (WAFR), 2004.* 15

[157] T. Lozano-Pérez. Spatial Planning : A Configuration Space Approach. In *Trans. on Computers, 1983*. 10

[158] R. Luna and K.E. Bekris. Efficient and complete centralized multi-robot path planning. In *Int. Conf. on Intelligent Robots and Systems (IROS), 2011*. 48

[159] R.C. Luo and J.H. Tzou. The development of distributed web-based rapid prototyping manufacturing system. In *Int. Conf. on Robotics and Automation (ICRA), 2003*. 48

[160] P. MacAlpine, K. Genter, S. Barrett, and P. Stone. The RoboCup 2013 Drop-In Player Challenges : Experiments in Ad Hoc Teamwork. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2014*. 90

[161] P. MacAlpine and P. Stone. Using Dynamic Rewards to Learn a Fully Holonomic Bipedal Walk. In *Adaptive Learning Agents Workshop (ALA), 2012*. 39

[162] R. Madhavan, K. Fregene, and L.E. Parker. Distributed Heterogeneous Outdoor Multi-robot Localization. In *Int. Conf. on Robotics and Automation (ICRA), 2002*. 46

[163] L.S. Marcolino and L. Chaimowicz. Traffic control for a swarm of robots : Avoiding group conflicts. In *Int. Conf. on Intelligent Robots and Systems (IROS), 2009*. 48

[164] L.S. Marcolino and L. Chaimowicz. Traffic control for a swarm of robots : Avoiding target congestion. In *Int. Conf. on Intelligent Robots and Systems (IROS), 2009*. 48

[165] M. Garcia Martineza and B. Walton. The wisdom of crowds : The potential of online communities as a tool for data analysis. In *Technovation, Vol. 34, Issue 4, pp. 203–214, 2014*. 87

[166] M.J. Matarić, M. Nilsson, and K.T. Simsarian. Cooperative Multi-Robot Box-Pushing. In *Int. Conf. on Intelligent Robots and Systems (IROS), 1995*. 46

[167] S. Behnke N. Mayer, J. Müller, and M. Schreiber. NimbRo 2004 Team Description. In *Team Description Paper, Robocup Kid Size League (RCUP), 2004*. 83

[168] J. Minguez and L. Montano. Nearness Diagram (ND) Navigation : Collision Avoidance in Troublesome Scenarios. *IEEE Trans. on Robotics and Automation*, 20(1) :45–49, 2004. 59

[169] N. Miyata, J. Ota, T. Arai, and H. Asama. Cooperative Transport by Multiple Mobile Robots in Unknown Static Environments Associated With

Real-Time Task Assignment. In *IEEE Trans. on Robotics and Automation, 18(5) :769–780, 2002.* 47

[170] M. Moors, T. Röhling, and D. Schulz. A Probabilistic Approach to Coordinated Multi-Robot Indoor Surveillance. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'05)*, pages 3447–3452, Alberta, Canada, 2005. 57

[171] A. Nagai. A new depth-first search algorithm for AND/OR trees. In *Master's thesis, University of Tokyo, 1999.* 73

[172] A. Nagai. Df-pn algorithm for searching AND/OR trees and its applications. In *Phd thesis, Department of Information Science, University of Tokyo, 2002.* 73

[173] E.V. Nalimov, G. McCrossan Haworth, and E.A. Heinz. Space-efficient indexing of chess endgame tables. In *ICGA Journal, 23(3) :148–162, 2000.* 77

[174] L. Nderu, N. Jouandeau, and H. Akdag. Importance of the neutral category in fuzzy clustering of sentiments. In *Int. Journal of Fuzzy Logic Systems (IJFLS), Vol.4, No.2, 2014.* 90

[175] L. Nderu, N. Jouandeau, and H. Akdag. Towards Universal Rating of Online multimedia content. In *Int. Conf. on Artificial Intelligence Applications (ARIA), 2014.* 90

[176] C. Niehaus, T. Röfer, and T. Laue. Gait Optimization on a Humanoid Robot using Particle Swarm Optimization. In *2nd Workshop on Humanoid Soccer Robots, IEEE-RAS 7th Int. Conf. on Humanoid Robots (HSR), 2007.* 39

[177] N. J. Nilson. Shakey The Robot. Technical Report 323, AI Center, SRI Int., 333 Ravenswood Ave., Menlo Park, CA 94025, 1984. 7

[178] O. Obst and M. Rollmann. Spark - A Generic Simulator for Physical Multi-Agent Simulations. In *Multiagent System Technologies, LNCS, Vol. 3187, pp. 243-257, 2004.* 31, 34, 35, 42

[179] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi. Estimation of Essential Interactions from Multiple Demonstrations. In *IEEE Int. Conf. on Robotics and Automation (ICRA'03)*, pages 3893–3898, Taipei, Taiwan, 2003. 52

[180] J. O'Rourke. Computational Geometry in C (Second Edition). In *Cambridge University Press, 1998.* 8

[181] L. Panait and S. Luke. A pheromone-based utility model for collaborative foraging. In *Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS'04), 2004.* 50

[182] M. Pardowitz, R. Zollner, and R. Dillmann. Incremental Acquisition of Task Knowledge applying Heuristic Relevance Estimation. In *IEEE Int. Conf. on Robotics and Automation (ICRA'06)*, pages 3011–3016, Orlando, FL, USA, 2006. 52

[183] R. Pfeifer and F. Iida. Morphological computation : connecting body, brain, and environment. In *Japanese Scientific Monthly, Vol. 58, No 2, pp. 48–54, 2005.* 81, 83

[184] F. Pfeiffer, K. Löffler, and M. Gienger. The concept of jogging johnnie. In *Int. Conf. on Robotics and Automation (ICRA'02), Vol. 3, pp. 3129–3135, 2002.* 31

[185] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L.M. Gambardella, and M. Dorigo. ARGoS : A modular, parallel, multi-engine simulator for multi-robot systems. In *Swarm Intelligence, 6(4), 271–295, 2012.* 50

[186] G. Pini, A. Brutschy, C. Pinciroli, M. Dorigo, and M. Birattari. Autonomous task partitioning in robot foraging : an approach based on cost estimation. In *Adaptive Behavior, 21(2) 118–136, 2013.* 50

[187] J. Pitrat. Ai systems are dumb because ai researchers are too clever. In *ACM Comput. Surv. 27(3) : 349-350, 1995.* 86

[188] E. Plaku and L.E. Kavraki. Distributed Sampling-Based Roadmap of Trees for Large-Scale Motion Planning. In *Int. Conf. Robotics and Automation (ICRA), 2005.* 2, 8

[189] B. Sheela Rani, N. Nandhitha, and N. Manoharan. Euclidean Distance Based Color Image Segmentation Algorithm for Dimensional Characterization of Lack of Penetration from Weld Thermographs for On-line Weld Monitoring in GTAW. In *17th World Conf. on Nondestructive Testing, Shanghai, China, 2008.* 37

[190] H.J. Reif. Complexity of the Mover's Problem and Generalizations. In *20th IEEE Symp. on Foundations of Computer Science, pp.421-427, 1979.* 8

[191] J. W. Romein and H. E. Bal. Solving awari with parallel retrograde analysis. In *IEEE Computer, 36(10) :26–33, 2003.* 67, 77

[192] S.I. Roumeliotis and G.A. Bekey. Distributed Multirobot Localization. In *IEEE Trans. on Robotics and Automation, 18(5) :781–795, 2002.* 46

[193] A. Saffidine and T. Cazenave. Generalized proof number search. In *6th Journées Francophones Modèles Formels de l'Interaction (MFI), pp. 131–138, 2011.* 74

[194] A. Saffidine, N. Jouandeau, C. Buron, and T. Cazenave. Material Symmetry to Partition Endgame Tables. In *Computers and Games (CG), Yokohama, 2013.* 5, 78, 86

[195] A. Saffidine, N. Jouandeau, and T. Cazenave. Solving Breakthrough with Race Patterns and Job-Level Proof Number Search. In *Advanced Computer Games 2011, Tilburg (ACG), 2011.* 5, 74, 89

[196] M. Saha and P. Isto. Multi-Robot Motion Planning by Incremental Coordination. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'06)*, pages 5960–5963, Beijing, China, 2006. 57

[197] J.T. Saitoa, M.H.M. Winands, and H.J. van den Herik. Randomized Parallel Proof-Number Search. In *12th Int. Conf. Advanced Computer Games (ACG), 2009.* 74

[198] M.P.D. Schadd, M.H.M. Winands, J.W.H.M. Uiterwijk, H.J. van den Herik, , and M.H.J. Bergsma. Best play in fanorona leads to draw. In *New Mathematics and Natural Computation, 4(3) :369–387, 2008.* 74

[199] J. Schaeffer. Conspiracy Numbers. In *Artificial Intelligence, Vol. 43, No. 1, pp. 67–84, 1990.* 72

[200] J. Schaeffer, Y. Bjornsson, N.Burch, A. Kishimoto, M. Muller, R. Lake, P. Lu, and S. Sutphen. Solving Checkers. In *Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 292-297, 2005.* 67

[201] J. Schaeffer, Y. Björnsson, N. Burch, R. Lake, P. Lu, and S. Sutphen. Building the checkers 10-piece endgame databases. In *Advances in Computer Games 10, pp. 193–210, 2003.* 77, 78

[202] A. Schmitz, M. Missura, and S. Behnke. Real-time trajectory generation by online footstep planning for a humanoid soccer robot. In *15th annual RoboCup Int. Symp. (RCUP), 2011.* 32

[203] J.T. Schwartz and M. Sharir. On the piano movers problem :I, II, III, IV, V. Technical report, New York University, Courant Institute, Department of Computer Sciences, 1983. 8

[204] M. Seo. The C* algorithm for AND/OR tree search and its application to a tsume-shogi program. In *Master's thesis, Departement of Information Science, University of Tokyo, 1995.* 73

[205] J. Signorini and P. Greussay. Object-oriented Wound Healing in the Liver : a Class-Structured View of Fibrogenesis and a Glimpse of its Evolution. In *ACM SIGAPP Symposium on Applied Computing, 2005.* 5

[206] O. Simonin, F. Charpillet, and E. Thierry. Collective construction of numerical potential fields for the foraging problem. In *Trans. on Autonomous and Adaptive Systems, ACM TAAS 2010.* 50

[207] SimSpark. A generic physical multiagent simulator system for agents in three-dimensional environments. In `http://simspark.sourceforge.net/`. 31, 34, 35, 42

[208] G. Song and N.M. Amato. Randomized Motion Planning for Car-like Robots with C-PRM. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 37–42, Maui, HI, USA, 2001. 61

[209] A. Stentz. Optimal and Efficient Path Planning for Partially-Known Environments. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3310–3317, San Diego, CA, USA, 1994. 61

[210] A. Stentz and M.B. Dias. A free market architecture for coordinating multiple robots. In *CMU-RI-TR-99-42, Carnegie Mellon University, Pittsburgh, PA, USA, 1999.* 49

[211] P. Stone and M. Veloso. Multiagent systems : A survey from a machine learning perspective. In *Autonomous Robots, 8(3) :345–383, 2000.* 45

[212] A.B. Stroud, M. Morris, K. Carey, J.C. Williams, C. Randolph, and A.B. Williams. MU-L8 : The Design Architecture and 3D Printing of a Teen-Sized Humanoid Soccer Robot. In *13th IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids), 2013.* 84

[213] P. Svestka. *Robot Motion Planning using Probabilistic Roadmaps*. PhD thesis, Utrecht University, 1997. 1, 9

[214] P. Švestka and M.H. Overmars. Coordinated Motion Planning for Multiple Car-Like Robots Using Probabilistic Roadmaps. In *IEEE Int. Conf. on Robotics and Automation (ICRA'95)*, pages 1631–1636, Nagoya, Japan, 1995. 56, 59

[215] F. Tang and L.E. Parker. ASyMTRe : Automated Synthesis of Multi-Robot Task Solutions through Software Reconfiguration. In *IEEE Int. Conf. on Robotics and Automation (ICRA'05)*, pages 1501–1508, Barcelona, Spain, 2005. 52

[216] K. Thompson. 6-piece endgames. In *ICCA Journal, 19(4) :215–226, 1996.* 77

[217] K. Thompson. Retrograde analysis of certain endgames. In *ICCA Journal, 9(3) :131–139, 1986.* 77

[218] S. Thrun. Learning Metric-Topological Maps for Indoor Mobile Robot Navigation. *Artificial Intelligence*, 99(1) :21–71, 1998. 56

[219] J. Tožička, J. Jakubův, and A. Komenda. Generating Multi-Agent Plans by Distributed Intersection of Finite State Machines. In *21th European Conf. on Artificial Intelligence, Computer Games Workshop (CGW-2014).* 89

[220] M. Travers. Animal construction kits. In *Int. Conf. on Artificial Life (ALIFE), 421-442, 1987.* 50

[221] T. Ueda, T. Hashimoto, J. Hashimoto, and H. Iida. Weak proof-number search. In *6th Int. Conf. Computers and Games (CG), 2008.* 74

[222] I. Ulrich and J. Borenstein. VFH+ : Reliable Obstacle Avoidance for Fast Mobile Robots. In *IEEE Int. Conf. on Robotics and Automation (ICRA'98)*, pages 1572–1577, Leuven, Belgium, 1998. 59

[223] D. Vallejo, I. Remmler, and N. M. Amato. An adaptive framework for 'single shot' motion planning : A self-tuning system for rigid and articulated robots. In *IEEE Int. Conf. on Robotics and Automation, pp. 21–26, Seoul, Korea, 2001.* 10

[224] R.T. Vaughan, K. Stoy, G.S. Sukhatme, and M.J. Matarić. LOST : Localization-Space Trails for Robot Teams. *IEEE Trans. on Robotics and Automation*, 18(5) :796–812, 2002. 51

[225] M. Vukobratovic and B. Borovac. Zero-moment point - thirty five years of its life. In *Int. J. of Humanoid Robotics, Vol. 1(1), pp. 157–173, 2004.* 33

[226] Z.D. Wang, Y. Hirata, and K. Kosuge. Control a Rigid Caging Formation for Cooperative Object Transportation by Multiple Mobile Robots. In *Int. Conf. on Robotics and Automation (ICRA), 2004.* 47

[227] J. Wawerla and R.T. Vaughan. A fast and frugal method for team-task allocation in a multi-robot transportation system. In *Int. Conf. on Robotics and Automation (ICRA), 2010.* 46

[228] S.A. Wilmarth, N.M. Amatoy, and P.F. Stiller. MAPRM : A Probabilistic Roadmap Planner with Sampling on the Medial Axis of the Free Space. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1024–1031, Detroit, MI, USA, 1999. 61

[229] M.H.M. Winands, J.W.H.M. Uiterwijk, and H.J. van den Herik. PDS-PN : A New Proof-Number Search Algorithm. In *3rd Int. Conf. Computers and Games (CG), Edmonton, 2002.* 73

[230] A. Winfield. Foraging robots. In *R. Meyers, ed. Encyclopedia of Complexity and Systems Science. Springer, New York, pp. 3682-3700, 2009.* 49

[231] I.C. Wu, H.H. Lin, P.H. Lin, D.J. Sun, Y.C. Chan, and B.T. Chen. Job-level proof-number search for Connect6. In *7th Int. Conf. Computers and Games (CG), 2010.* 74

[232] K.M. Wurm, C. Stachniss, and W. Burgard. Coordinated Multi-Robot Exploration using a Segmentation of the Environment. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'08)*, pages 1160–1165, Nice, France, 2008. 56

[233] P.G. Xavier. Fast swept-volume distance for robust collision detection. In *Int. Conf. Robotics and Automation (ICRA), 1997.* 12

[234] F. Xue, X. Chen, J. Liu, and D. Nardi. Real time biped walking gait pattern generator for a real robot. In *15th annual RoboCup Int. Symp. (RCUP), 2011.* 32

[235] Z. Yan. *Contributions à la coordination de tâches et de mouvements pour un système multi-robots*. PhD thesis, Université Paris 8, 2012. 3

[236] Z. Yan and N. Jouandeau. ACS-PRM : Adaptive Cross Sampling Based Probabilistic Roadmap for Multi-robot Motion Planning. In *12th Int. Conf. on Intelligent Autonomous Systems (IAS), 2012.* 3, 56

[237] Z. Yan, N. Jouandeau, and A. AliCherif. Multi-robot Heuristic Goods Transportation. In *6th IEEE Int. Conf. on Intelligent Systems (IS), 2012.* 55

[238] Z. Yan, N. Jouandeau, and A. Ali Cherif. Multi-Robot Decentralized Exploration Using a Trade-Based Approach. In *8th Int. Conf. on Informatics in Control, Automation and Robotics (ICINCO), 2011.* 3, 56, 82

[239] Z. Yan, N. Jouandeau, and A. Ali Cherif. On the problem of task planning in multi-robot systems. In *9th Int. Conf. on Informatics in Control, Automation and Robotics (ICINCO), 2012.* 3, 56

[240] Z. Yan, N. Jouandeau, and A. Ali Cherif. Sampling-Based Multi-Robot Exploration. In *Int. Conf. ISR/ROBOTIK (ISR), 2010.* 56

[241] Z. Yan, N. Jouandeau, and A. Ali Cherif. Sampling-based Multi-robot Motion Planning. In *IVC-ITS 2013 (Special Session on Intelligent Vehicle Controls - Intelligent Transportation Systems), 10th Int. Conf. on Informatics in Control, Automation and Robotics (ICINCO), 2013.* 3, 56

[242] Z. Yan, N. Jouandeau, and A. Ali Cherif. Towards a probabilistic roadmap for multi-robot coordination. In *Int. Conf. on Artificial Intelligence (ICAI), 2012.* 3, 56

[243] Z. Yan, N. Jouandeau, and V. Hugel. L3M-SIM Team Description Paper. In *Robocup 3D Soccer Simulation League (RCUP), 2011.* 34

[244] A. Yershova and S.M. LaValle. Improving Motion Planning Algorithms by Efficient Nearest-Neighbor Searching. In *IEEE Trans. on Robotics, 23(1) :151-157, 2007.* 15

[245] A. Yershova, S.M. LaValle, and J.C. Mitchell. Generating Uniform Incremental Grids on SO(3) Using the Hopf Fibration. In *8th Int. Workshop on the Algorithmic Foundations of Robotics (WAFR), 2008.* 15

[246] O. Zedadra, N. Jouandeau, and H. Seridi. Collaborative Foraging Using a new Pheromone and Behavioral model. In *Int. Symp. on Informatics and its Applications (ISIA), 2014.* 50

[247] O. Zedadra, N. Jouandeau, H. Seridi, and G. Fortino. Stigmergic MASA : A Stigmergy Based Algorithm for Multi-Target Search. In *1st workshop on Multi-Agent Systems and Simulation, 4th Joint Agent-oriented Workshops in Synergy, Federated Conference on Computer Science and Information Systems, pp. 1515–1523, 2014.* 83, 88

[248] O. Zedadra, H. Seridi, and N. Jouandeau. Cooperative c-Marking Agents for the Foraging Problem. In *4th Int. Conf. on Advances in System Simulation (SIMUL), 2012.* 50

[249] E.H. Østergaard, G.S. Sukhatme, and M.J. Mataric. Emergent bucket brigading : A simple mechanism for improving performance in multi robot constrained-space foraging tasks. In *Int. Conf. On Autonomous Agents, Montreal, Canada, 2001.* 49

Titre : Contributions de la robotique mobile, humanoïde, multi-robots aux jeux à information incomplète

Résumé : Nous présentons les résultats de 10 années de travaux et d'expérimentation sur les algorithmes probabilistes, sur l'exploration et le transport multi-robots, sur les mouvements en robotique humanoïde, sur l'informatique médicale et sur les jeux. Sont présentés des principes de décomposition de l'espace des configurations permettant d'accélérer la résolution de problèmes de planification en robotique, la formalisation de la notion de situation d'attente utile aux problèmes de transport et d'exploration en robotique mobile, des solutions heuristiques minimisant temps et énergie consommés, des solutions probabilistes par échantillonnage, par estimation des congestions hors-ligne et par adaptation en-ligne à l'encombrement de l'espace de travail pour la robotique, des solutions de génération de mouvements, de vision temps-réel embarquée et de génération de stratégies collectives pour la robotique humanoïde, des travaux en optimisation morphologique humanoïde, des solutions de parallélisation des algorithmes Monte Carlo pour le domaine des jeux, des solutions combinant analyse retrograde, résolution parallèle et simplification par symétries pour des jeux à 1 et 2 joueurs. Les nombreuses participations dans les compétitions internationales de robotique avec la plateforme humanoïde *NAO* et dans les compétitions internationales de jeux telles que les *Computer Olympiad*, nous ont permis de valider en pratique les idées présentées dans ses travaux de recherche.

Title: From robotics to incomplete information games

Abstract: We present the results of 10 years of research and experiments on probabilistic motion planning, on transportation and exploration, on humanoids, medical computer science and computer games. This report presents configuration space solutions to accelerate motion planning solvers, a usefull waiting situation formalization for intelligent transport systems, heuristic solutions to minimize time and energy consumptions, off-line probabilistic estimated congestion and online adaptation in cluttered configuration spaces, motion generators, real-time embedded computer vision, collective strategies and morphological optimization for humanoids, parallelization of Monte Carlo algorithms in computer game, combination of retrograde analysis, parallelization and symmetries exploitation for 1 and 2 player games. The numerous participations in international competitions with *NAO* humanoid such as *RCUP-SPL* and *RCUP-3DSSL* and in international game competitions such as *Computer Olympiad* and *TCGA Competitions*, allowed us to validate ideas presented in this research.