



Faculté des sciences et de génie
Département de génie mécanique

Revue de littérature

Présenté à :
Professeur Claude-Guy Quimper

Par :
Alexis Fortin-Côté - 908 191 694
Vincent Babin - 910 227 872
André Gallant - 111 062 993

dans le cadre du cours :
IFT-7020
Optimisation Combinatoire

30 mars 2014

Graph search joint path planning for robot center of gravity positioning [1]

Cet article présente une méthode pour trouver des trajectoires articulaires à l'aide de la recherche dans les graphes, une approche similaire à celle qui sera explorée dans notre projet. L'algorithme de recherche dans un graphe présenté dans l'article est utilisé pour placer le centre de gravité d'un robot mobile en temps réel. Le déplacement ainsi occasionné permet au robot de se mouvoir en roulant. La position articulaire initiale et le centre de gravité du robot sont connus. Le problème est de trouver la trajectoire articulaire permettant de déplacer le centre de gravité à la position finale.

Un espace de configuration (C-space) valide doit d'abord être construit. Cet espace est de dimension trois avec les trois axes correspondants aux trois degrés de liberté du robot ($\theta_1, \theta_2, \theta_3$). Cet espace est ensuite discrétisé en voxels, chaque voxel représente une pose articulaire et le centre de gravité correspondant. On doit ensuite vérifier que chaque voxel ne représente pas un état où il y a interférence entre les membrures du robot. À cette fin, une fonction permettant de déterminer s'il y a des interférences est lancée à chaque voxel. S'il y a interférence, le voxel est retiré de l'espace de configuration. Chacun de ces voxels représente un nœud dans le graphe de recherche. Chaque nœud est relié par des arcs aux nœuds adjacents dans l'espace de configuration.

Deux méthodes de recherche sont explorées. La première a comme heuristique de recherche de choisir le nœud adjacent pour lequel le centre de gravité correspondant est le plus proche du centre de gravité final. Comme l'espace de configuration est discrétisé, la solution est trouvée lorsque le centre de gravité trouvé se trouve dans un intervalle de tolérance autour de la valeur visée. La deuxième méthode trouve d'abord une pose articulaire avec un centre de gravité près du centre de gravité final. La distance de Manhattan minimale entre cette pose et les poses des nœuds adjacents est alors utilisée pour choisir le nœud suivant jusqu'à ce que l'un des nœuds adjacents soit la pose articulaire finale. Les nœuds déjà visités sont exclus de la suite de la recherche dans les deux méthodes.

Cet article est un exemple de planification de trajectoire appliqué à la robotique mobile. La méthode est près de celle envisagée dans notre projet, à la différence où l'objectif final n'est pas la position du centre de gravité, mais directement l'état articulaire. Notre projet, par contre, doit aussi inclure dans l'espace de recherche les vitesses articulaires, ce qui n'est pas pris en compte dans l'algorithme de l'article. Aussi, leur heuristique de recherche n'est pas applicable directement dans notre problème. Une heuristique applicable à notre projet est présentée dans l'article de Wurll [2] suivant.

Point-to-point and multi-goal path planning for industrial robots [2]

Cet article présente des méthodes pour trouver des chemins sans collisions pour des robots industriels. L'article explore les problèmes points-à-points et multi-points. Le problème point-à-point est celui le plus intéressant en lien avec notre problématique. De plus, une section de l'article est dédiée aux méthodes de détection de collisions, ce qui est un sujet de moindre importance dans l'optique de notre projet.

Les auteurs de l'article se basent également sur la recherche dans l'espace de configuration par des algorithmes de type A* qui sont des améliorations des algorithmes de recherche tel que l'algorithme de Dijkstra [10]. Pour utiliser ces algorithmes, l'espace de configuration doit être discrétisé. Le principal problème lié à la discrétisation est le choix de la résolution. Une résolution trop fine fait croître l'espace de recherche, alors qu'une résolution trop grossière peut éliminer des solutions possibles. Les auteurs ont recours à des heuristiques de discrétisation qui sont présentées dans l'article. Ces heuristiques pourraient facilement être intégrées à notre méthode de recherche dans le graphe pour les problèmes à plusieurs degrés de liberté.

Un algorithme de recherche dans le graphe de l'espace de configuration est aussi présenté et utilise des ensembles pour garder les nœuds en exploration et ceux explorés en mémoire. Les nœuds en exploration sont classés selon une heuristique qui est présentée et qui détermine l'ordre d'exploration. Cet algorithme de recherche est amélioré de façon à pouvoir paralléliser les calculs. Leur méthode de parallélisation repose sur une décomposition de l'espace de configuration et la création d'ensembles ("nœuds en exploration", "nœuds explorés") locaux sur chaque processeur. L'algorithme est aussi bidirectionnel, c.-à-d. que la recherche peut commencer du point de départ et du point d'arrivée. L'avantage de cette recherche est que la recherche à

partir du point final peut être plus rapide que la recherche à partir du point initiale et vice-versa. De plus, si les deux chemins se rejoignent, le temps de calcul est réduit.

La recherche bidirectionnelle, l’heuristique de discrétisation et l’heuristique de recherche utilisées dans cet article seront probablement utilisées dans le cadre de notre projet. L’adaptation qui devra être faite est l’inclusion de la dynamique du robot qui devra être prise en compte dans la discrétisation de l’espace de configuration. Ceci implique que cet espace ne sera pas seulement l’espace articulaire, mais doit inclure d’autres états comme les vitesses articulaires. Les auteurs de l’article suggèrent une amélioration de la recherche en utilisant les graphes hiérarchiques. Sujet qui est abordé dans par Fernandez [3] dans l’article suivant.

Hierarchical graph search for mobile robot path planning [3]

Cet article présente une méthode de recherche de trajectoires pour les robots mobiles utilisant les graphes de type hiérarchiques (H-Graph). Les graphes hiérarchiques sont des graphes à plusieurs niveaux de détails, chaque nœud du graphe peut contenir des sous-graphes et chaque arc peut contenir des sous-arcs. Un exemple utilisé dans l’article est le plan d’un édifice pour la navigation d’un robot mobile. Les niveaux moins détaillés de la hiérarchie sont les salles et les corridors. Les niveaux plus détaillés représentent les obstacles sur le sol, les portes et les autres robots.

L’objectif de l’article est l’amélioration de l’efficacité de la recherche par l’utilisation des caractéristiques des H-Graph par rapport à la recherche dans des graphes ordinaires. La base de leur algorithme de recherche est de trouver un chemin dans le graphe de haute hiérarchie et d’ensuite augmenter les détails récursivement jusqu’à la plus basse hiérarchie. La méthode se sert d’intervalles des coûts pour les arcs. Ces intervalles représentent le coût minimum et le coût maximum pour traverser l’arc pour plusieurs essais. Ils définissent ensuite un certain nombre d’opérations valides sur ces intervalles. L’algorithme de base que les auteurs modifient est l’algorithme de Dijkstra [10], bien connu pour la recherche dans les graphes simples. L’algorithme est modifié et devient ς . Il retourne non plus un seul chemin faisable avec un coût et une longueur, mais un ensemble de plusieurs chemins ayant le même coût. L’application de cet algorithme directement au plus bas niveau de la hiérarchie serait potentiellement prohibitif en temps de calcul, mais retourne ce qu’ils appellent le chemin réel optimal \mathbf{P}^* . L’algorithme proposé ξ_ς utilise ς à des niveaux de hiérarchie plus élevés pour réduire le nombre de nœuds à chercher. Un ensemble d’expression, donc celle qu’ils appellent *sufficient condition for optimality*, ainsi que certaines hypothèses, non-restrictives dans le cadre de la robotique mobile, doivent être respectées pour que leur algorithme ξ_ς n’élimine pas la solution optimale à des hiérarchies plus élevées. L’article présente ensuite un exemple d’utilisation avec un H-Graph à deux niveaux représentant la recherche d’une trajectoire sur un campus.

Cet article nous permet d’avoir un exemple de la recherche dans un graphe appliqué à la robotique. Dans le cadre de notre problématique, cette méthode pourrait être envisageable pour les problèmes plus complexes, où une chaîne de mouvement donnée doit être accomplie en minimisant les temps de déplacement. La méthode présentée permettrait de raffiner notre recherche dans un graphe en hiérarchisant la discrétisation de l’espace d’états, c.-à-d. d’avoir une discrétisation de l’espace de plus en plus fine et d’en faire une hiérarchie.

Minimum-time control of robotic manipulators with geometric path constraints [4]

Cet article présente une méthode pour déterminer la trajectoire optimale pour le problème de suivi de trajet géométrique¹ avec le temps total d’exécution comme critère d’optimisation. La méthode présentée requiert que le trajet soit préétabli. Celle-ci est alors chargée de trouver le temps auquel chaque point du trajet est atteint pour créer une trajectoire. La trajectoire contient alors l’information sur la vitesse et l’accélération tout au long du trajet ce qui permet de déterminer les efforts articulaires nécessaires pour l’exécuter. Il est donc possible d’exploiter la dynamique complète du robot et d’imposer des contraintes sur

1. Séquence de positions du robot en ordre, mais sans information sur le temps auquel chaque point est atteint.

les efforts articulaires. Cela rend la méthode intéressante, car la véritable trajectoire optimale permise par les actionneurs du robot est trouvée pour le cas où le trajet est prédéterminé.

La méthode en question se résume ainsi. Premièrement, en partant du début du trajet, on intègre celui-ci en imposant une accélération la plus grande possible (la contrainte d'effort maximal d'au moins une articulation sera active et empêche le robot de suivre le trajet plus rapidement). Ensuite le même processus est effectué, mais dans le sens inverse à partir de la fin du trajet. Si ces deux courbes se rencontrent, la solution est trouvée et le point d'intersection marque alors le point où le système change d'une phase d'accélération maximale à la phase de décélération maximale. Cependant, il existe un espace dans le plan position-vitesse de suivi où aucune commande ne permet au robot de rester sur le trajet. Si l'intégration dans un sens ou dans l'autre mène à un tel point, le point le plus proche sur la frontière de cette contrainte qui est tangent à la décélération maximale est trouvé. De ce point, le trajet est intégré dans les deux sens jusqu'à ce que les deux courbes rencontrent leur courbe d'accélération et de décélération maximales respectives. La méthode est donc continuée jusqu'à ce que la trajectoire soit complètement définie.

Il a été démontré que cette méthode donne la solution optimale pour le suivi de trajet en temps minimal, mais elle ne peut pas être appliquée directement dans le cadre de notre projet pour deux raisons principales : elle requiert que le robot soit capable de maintenir toute posture statiquement, ce qui est contre les objectifs de notre projet, et elle requiert que le trajet soit préétabli ce qui sort encore des objectifs de notre projet.

A dynamic programming approach to trajectory planning of robotic manipulators [5]

Cet article présente une méthode qui peut être vue comme une extension de la méthode présentée dans [4] pour inclure des contraintes d'efforts articulaires plus complexes. Un exemple serait le cas où les contraintes d'efforts articulaires sont couplées ou bien le cas où limiter la dérivée des efforts articulaires est désiré. En outre, point plus intéressant, la fonction objectif peut être plus générale que la simple minimisation du temps d'exécution.

Pour ce faire, la méthode présentée dans cet article utilise la programmation dynamique pour trouver la trajectoire qui suit le trajet prescrit de façon optimale selon une fonction objectif quelconque. La méthode, comme celle dans [4], travaille dans l'espace position sur le trajet et vitesse de suivi. Cela est valide, car le trajet est prédéterminé et a l'avantage de réduire la dimensionnalité de l'optimisation de tout système possédant n degrés de liberté de $2n$ dimensions (position et vitesse de chaque articulation) à 2 dimensions (position et vitesse sur le trajet). Le fameux *fléau de la dimension* est alors évité. Cela étant dit, la programmation dynamique requiert une discrétisation de cet espace et cherche un chemin optimal du début du trajet vers sa fin tout en respectant les contraintes articulaires. La taille du problème d'optimisation est alors largement dépendante de la finesse de la discrétisation. La méthode consiste essentiellement à effectuer une fouille en profondeur dans l'espace discrétisé à partir de la fin de la trajectoire pour trouver le chemin avec le coût le moindre qui mène vers le début de la trajectoire.

Des parallèles importants peuvent être observés avec les approches des articles [Fern1998, wurll2001point, phipps2008] puisque l'espace des états du robot le long du trajet est discrétisé sous forme de graphe et un chemin est trouvé. La distinction la plus importante est le fait que, dans les articles [4, 5] l'espace de solution est de 2 dimensions et donc impose la condition que le trajet soit prédéterminé.

On point-to-point motion planning for underactuated space manipulator systems [6]

Cet article présente une différente méthode de génération de trajectoire appliquée aux manipulateurs sous-actionnés² utilisés en apesanteur. La méthode fonctionne en faisant une paramétrisation de la trajectoire articulaire et cherche à trouver les paramètres permettant d'effectuer la tâche prescrite en temps prédéterminé.

La méthode présentée débute en imposant une forme polynomiale à la trajectoire articulaire avec un certain nombre de paramètres superflus, c.-à-d. des paramètres qui ne sont pas explicitement fixés par les

2. Manipulateurs qui possèdent un nombre inférieur d'actionneurs que de degrés de liberté de mouvement.

conditions limites³ imposées par la tâche. Ces paramètres superflus sont nécessaires puisque les systèmes contrôlés sont sous-actionnés. Puisqu'il n'existe seulement qu'un petit sous-ensemble de combinaisons de ces paramètres superflus qui créent une trajectoire respectant les contraintes des efforts articulaires, une optimisation est faite sur ceux-ci pour trouver une solution en utilisant un algorithme de programmation quadratique séquentielle (SQP).

L'avantage principal de cette méthode par rapport aux méthodes des articles [4] et [5] est le fait qu'aucun trajet n'est à fournir. Cela augmente alors les possibilités de trajectoires. Cette méthode a cependant l'inconvénient de devoir fixer le temps total d'exécution qui limite la possibilité d'optimisation. Le fait que cette méthode fonctionne avec des manipulateurs sous-actionnés la rend idéale pour traiter le problème d'augmenter la charge utile des robots posé dans notre projet puisque le sous-actionnement peut être considéré comme l'ultime contrainte d'effort articulaire.

Particle swarm optimization : basic concepts, variants and applications in power systems [7]

Cet article présente les principes de base de l'optimisation par essaim particulaire (PSO) qui permet de résoudre des problèmes dont la fonction objectif est non-convexe tel le problème de la planification de la trajectoire. Celle-ci se résume ainsi :

Le vecteur des paramètres d'optimisation est défini par \mathbf{x} et la fonction objectif est $F(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x})$ où f est la fonction à minimiser et g est une fonction de pénalité si une contrainte est violée. L'algorithme initialise aléatoirement un nombre finit de solution appelées particules dans l'espace de recherche et par la suite évalue F pour chaque particule. On définit \mathbf{p}_g comme la particule ayant donné F la plus petite depuis le début de la résolution. Par la suite on définit \mathbf{p}_i comme étant la meilleure solution trouvée par une même particule depuis le début de l'optimisation. Cette variable est en quelque sorte la mémoire de la particule. L'algorithme itère comme suit. Suite à l'initialisation on calcule la vitesse des particules selon :

$$\mathbf{v}_i(k) = \mathbf{v}_i(k-1) + rand_1 \cdot (\mathbf{p}_i - \mathbf{x}_i(k-1)) + rand_2 \cdot (\mathbf{p}_g - \mathbf{x}_i(k-1)) \quad (1)$$

où k est la présente itération.

La première composante de l'équation représente la tendance de la particule à garder la même vitesse (inertie). La deuxième composante de la vitesse est une attraction vers la meilleure solution trouvée par cette même particule (mémoire). La dernière composante de l'équation est une attraction vers la meilleure solution trouvée parmi toutes les particules (coopération). Suite au calcul des vitesses on met à jour les particules selon :

$$\mathbf{x}_i(k) = \mathbf{x}_i(k-1) + \mathbf{v}_i(k) \quad (2)$$

Par la suite on itère jusqu'à ce que la valeur de $F(\mathbf{p}_g)$ ne varie pas plus d'une valeur appelée *tolérance* pendant plus d'un certain nombre d'itérations nommées *stall*.

PSO-based time-optimal trajectory planning for space robot with dynamic constraints [8]

Cet article utilise l'algorithme PSO pour optimiser la trajectoire d'un véhicule spatial. Un des points importants de cet article est la modélisation du problème. Ici, on modélise la trajectoire avec des polynômes définis par morceaux et points de dégagements. De façon analogue à une fonction spline, le modèle est construit de sorte à imposer la position initiale et finale et à chercher la position, la vitesse et le temps d'un certain nombre de points nommés points de dégagement. La fonction objectif cherche alors à optimiser le temps total d'exécution en rapprochant les points de dégagement les uns des autres.

Un des avantages de cette méthode, comparativement à une méthode où la trajectoire est discrétisée sans interpolation polynomiale, est que l'espacement de la discrétisation est aussi optimisé. Cette propriété est

3. position, vitesse et accélération au début et à la fin de la trajectoire.

intéressante, car elle nous assure que tous les points contribuent à augmenter la précision de la solution ce qui n'est pas vrai des modèles où la discrétisation temporelle est fixe. Dans ces modèles, les derniers points sont souvent redondants. Un autre bénéfice de cette méthode est qu'une fois la solution partielle obtenue, la solution peut être affinée plus facilement en utilisant une descente de gradient car $f(\mathbf{x})$ sera continue et différentiable.

Dynamical exploitation space reduction in particle swarm optimization for solving large scale problems [9]

Cet article montre l'utilisation des méthodes PSO combinant deux stratégies d'exploration qui visent à mieux couvrir l'espace de recherche et à prendre en compte des connaissances de la fonction objectif (Dynamic learning).

La première modification est basée sur le théorème du “No Free Lunch” (NFL) qui stipule qu'en général aucun algorithme de résolution n'est meilleur qu'un autre pour résoudre tous les problèmes. Sachant cela, il est possible de rendre l'algorithme plus performant en apprenant les caractéristiques de la fonction objectif. La méthode compte le nombre de particules présentes dans différentes régions de l'espace puis, lorsqu'un certain nombre d'itérations est atteint, une région de recherche de l'espace est retirée. Cette méthode est appliquée en même temps que la ré-initialisation partielle. La ré-initialisation est la deuxième modification et consiste à réinitialiser une partie des particules de manière aléatoire lorsqu'un certain critère est atteint. Un exemple de critère est lorsque la densité des particules atteint un certain seuil. Les valeurs sont réinitialisées sur le nouvel espace de recherche permis par la réduction de l'espace de recherche de la première méthode.

Les auteurs ont effectué plusieurs tests sur des fonctions ayant plus de 200 dimensions et les résultats suggèrent que l'application de cette méthode à notre projet, ayant plus de 50 dimensions, permettra de faciliter la convergence de la solution optimale. Les auteurs font aussi une comparaison entre la topologie en étoile et en anneau dans les PSO. Dans la topologie en étoile, toutes les particules connaissent \mathbf{p}_g parmi toutes les particules tandis que dans la topologie en anneau, le \mathbf{p}_g de chaque particule est la meilleure particule parmi ses deux plus proches voisins. Cette topologie mène à de meilleurs résultats parmi les fonctions testées parce qu'elle garde une plus petite densité de particules que la formation en étoile.

Références

- [1] C. C. Phipps, D. Johnson, and M. A. Minor, “Graph search joint path planning for robot center of gravity positioning,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 3878–3883, IEEE, 2008.
- [2] C. Wurll and D. Henrich, “Point-to-point and multi-goal path planning for industrial robots,” *Journal of Robotic Systems*, vol. 18, no. 8, pp. 445–461, 2001.
- [3] J. Fernandez and J. Gonzalez, “Hierarchical graph search for mobile robot path planning,” in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 1, pp. 656–661 vol.1, May 1998.
- [4] K. Shin and N. McKay, “Minimum-time control of robotic manipulators with geometric path constraints,” *Automatic Control, IEEE Transactions on*, vol. 30, no. 6, pp. 531–541, 1985.
- [5] K. G. Shin and N. D. McKay, “A dynamic programming approach to trajectory planning of robotic manipulators,” *Automatic Control, IEEE Transactions on*, vol. 31, no. 6, pp. 491–500, 1986.
- [6] I. Tortopidis and E. Papadopoulos, “On point-to-point motion planning for underactuated space manipulator systems,” *Robotics and Autonomous Systems*, vol. 55, no. 2, pp. 122–131, 2007.
- [7] Y. Del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. G. Harley, “Particle swarm optimization : basic concepts, variants and applications in power systems,” *Evolutionary Computation, IEEE Transactions on*, vol. 12, no. 2, pp. 171–195, 2008.

- [8] P. Huang and Y. Xu, “Pso-based time-optimal trajectory planning for space robot with dynamic constraints,” in *Robotics and Biomimetics, 2006. ROBIO’06. IEEE International Conference on*, pp. 1402–1407, IEEE, 2006.
- [9] S. Cheng, Y. Shi, and Q. Qin, “Dynamical exploitation space reduction in particle swarm optimization for solving large scale problems,” in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pp. 1–8, IEEE, 2012.
- [10] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization : algorithms and complexity*. Courier Dover Publications, 1998.