2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

MPC-Graph: Feedback Motion Planning Using Sparse Sampling Based Neighborhood Graph

O. Kaan Karagoz, Simay Atasoy and M. Mert Ankarali

***Abstract*— Robust and safe feedback motion planning and navigation is a critical task for autonomous mobile robotic systems considering the highly dynamic and uncertain nature scenarios of modern applications. For these reasons motion planning and navigation algorithms that have deep roots in feedback control theory has been at the center stage of this domain recently. However, the vast majority of such policies still rely on the idea that a motion planner first generates a set of open-loop possibly time-dependent trajectories, and then a set of feedback control policies track these trajectories in closed-loop while providing some error bounds and guarantees around these trajectories. In contrast to trajectory-based approaches, some researchers developed feedback motion planning strategies based on connected obstacle-free regions, where the task of the local control policies is to drive the robot(s) in between these particular connected regions. In this paper, we propose a feedback motion planning algorithm based on sparse random neighborhood graphs and constrained nonlinear Model Predictive Control (MPC). The algorithm first generates a sparse neighborhood graph as a set of connected simple rectangular regions. After that, during navigation, an MPC based online feedback control policy funnels the robot with nonlinear dynamics from one rectangle to the other in the network, ensuring no constraint violation on state and input variables occurs with guaranteed stability. In this framework, we can drive the robot to any goal location provided that the connected region network covers both the initial condition and the goal position. We demonstrate the effectiveness and validity of the algorithm on simulation studies.**

I. Introduction

Obstacle avoidance motion planning is a fundamental problem in autonomous mobile robot applications. With the implementation of motion planning algorithms human factor is eliminated from the process, and the application areas of autonomous robots spread from undersea applications to space explorations. The core aim of robotic motion planning is to obtain a series of actions and configurations to navigate the robot from an initial state to a goal state while obeying the rules of the environment and taking into account the limitations of the sensors and actuators.

Generally, motion planning algorithms first generate a set of open loop piecewise-smooth trajectories/paths and then follow these trajectories with feedback control policies. Several studies utilized the model predictive control (MPC) framework to carry out the second phase. MPC powered motion planning and control methods are used on autonomous ground vehicles [1]–[3], aerial vehicles [4], [5], underwater vehicles [6] and spacecrafts [7]. However, the majority of these applications use time-dependent reference trajectories, which may cause deficiencies in the accuracy, stability, optimality, and computational efficiency. To overcome some of these issues, several researchers proposed a variety of solutions [2], [3], [8], [9].

The authors are with the Department of Electrical and Electronics Engineering, Middle East Technical University, 06800 Ankara, Turkey {karagoz.osman, simay.atasoy, mertan}@metu.edu.tr

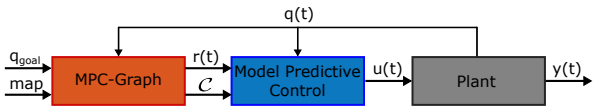

Fig. 1. Block diagram of the algorithm. MPC-Graph generates a reference signal, r, to reach from current state, $q(t)$, to the goal location, q_{goal} considering the obstacles, goal and map limits. Model Predictive Control computes the optimal input, $u(t)$, to get $r(t)$ in accordance with the constraints on $q(t)$, $\mathcal{C}$.

Instead of trajectory-based motion planning approaches, some researchers developed a trajectory free feedback motion planning concept that relies on dividing the workspace (or configuration space) into a set of connected sparse regions. Inside each region, the task of the motion planner is to navigate the robot to a different neighboring area. In that manner, each primary task is sequentially executed to fulfill the main task [10], i.e., convergence to the goal configuration.

On the other hand, feedback control of dynamical systems subject to some hard and soft constraints on state-space variables and inputs are emerging problems in control system applications. Yet few of the feedback motion planning algorithms directly enforce and address these constraints. In the context of robotic motion planning and control, some constraints need special attention, such as collision-free navigation, speed-limits, and actuator saturation. One of the most popular feedback control strategy that can enforce such constraints while also ensuring other critical system properties such as stability is the constrained MPC. It is a powerful tool in the sense that it estimates the future behavior of the system and generates the optimal input at each time step.

In this paper, we propose a new trajectory-free, sampling-based feedback motion planning algorithm that can handle arbitrary obstacle configurations for autonomous robots (in 2D environments). MPC makes it possible to use both linear and nonlinear system models, which highly increases the application areas of the proposed MPC-Graph algorithm. The algorithm consists of two stages. First stage generates a sparse graph structure that covers the collision-free configuration space with rectangular areas/volumes. Then, the second stage uses MPC to funnel the robot from a starting point to a goal location. Fig. 1 illustrates the block diagram

of the proposed algorithm.

We organize the paper as follows. Section II provides background on sampling-based motion planning algorithms and summarizes the MPC concept. Section III proposes the MPC-Graph algorithm in detail. Section IV reports simulation environment and the results. Section V outlines our work and discusses the future directions of the proposed study.

II. Background and Related Work

A. Sampling Based Motion Planning

Probabilistic Roadmaps (PRM) and Rapidly Exploring Random Trees (RRT) provide a foundation for the sampling-based motion planning algorithms. PRM algorithm first generates a graph (roadmap) that represents the collision-free configurations, and then one can find the optimal path through sampled nodes between the start and goal configurations [11]. On the other hand, the RRT algorithm generates nodes by sampling the collision-free space and connects the closest nodes and grows tree structure [12]. Both probabilistic methods can handle the difficulties that come from complex environments and high dimensional configuration spaces, making them powerful alternatives in many motion planning applications.

Sampling-based methods have plentiful modifications and extensions. For example, instead of using points, some of these studies focus on using sequentially connected regions. Each region navigates the robot next region to reach goal configuration, [10].

One of the first studies on this subject, Sampling Based Neighborhood Graph (SNG), was presented by Yang and LaValle, [13]. In this study, they partition the obstacle-free space with spherical regions (nodes) and create a neighborhood graph by analyzing the intersection of the nodes (Fig. 3). They define a global navigation function in a way that robot passes through one region to another until it reaches the node that contains the goal point. Later, Yang and LaValle [14] introduced an enhancement for this method to increase the size of the regions (at each sampling iteration), eventually obtaining a more sparse graph representation.

Several researchers developed methods based on sparse neighborhood trees (as opposed to the graph structure) to navigate to the robot using a different type of feedback controllers with varying shapes of node structures (ellipse, circle, square) [15]–[18].

B. Model Predictive Control

Model Predictive Control (MPC) is a feedback control algorithm that generates an input sequence at each sampling instant for the indicated horizon to minimize the related cost function while respecting the specified state and input constraints. After the computation of the input sequence, the controller drives the plant with the first element of the computed input array, and the horizon proceeds towards the future [19]. MPC aims to find an optimal reactive input signal that will derive the system to the desired configuration by minimizing the cost function.

In our work, we use the quasi-infinite horizon model predictive control, which guarantees asymptotic closed-loop stability [20]. In order to assure stability, Chen and Allgöwer propose a procedure for obtaining the terminal region and terminal cost matrix, which in turn generates an upper bound for the infinite horizon cost for the nonlinear problem. The optimization problem is formalized as follows:

$$J(\mathbf{x}(t),\mathbf{u}(\cdot)) = \int_t^{t+T_p} \Big(||\mathbf{x}(\tau)||_Q^2 + ||\bar{\mathbf{u}}(\tau)||_R^2 \Big) d\tau + ||\mathbf{x}(t+T_p)||_P^2 \tag{1}$$

subject to

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u}) \tag{2a}$$

$$\mathbf{u}(\tau) \in U, \qquad \tau \in [t, t+T_p] \tag{2b}$$

$$\mathbf{x}(t+T_p) \in \mathbf{\Omega}. \tag{2c}$$

$\mathbf{x}(\cdot)$ is the system state vector, $\mathbf{u}(\cdot)$ is the input to the system defined by a set of nonlinear differential equations, $f(\mathbf{x}(\cdot),\mathbf{u}(\cdot))$ and T_p is a finite prediction horizon. Q and R are positive-definite and symmetric matrices for states and inputs, respectively. We obtained terminal state penalty matrix P and terminal region $\mathbf{\Omega}$ by applying the procedure presented in [20] and obtained the following bound on the infinite horizon cost which guarantees stability of the closed-loop system,

$$||\mathbf{x}(t+T_p)||_P^2 \geq \int_{t+T_p}^{\infty} \Big(||\mathbf{x}(\tau)||_Q^2 + ||\mathbf{u}(\tau)||_R^2 \Big) d\tau \quad \forall \mathbf{x}(t+T_p) \in \mathbf{\Omega}. \tag{3}$$

III. MPC-Graph

MPC-Graph algorithm consists of three consecutive stages: graph generation, Dijkstra's search algorithm, and feedback motion control. In the graph generation stage, the algorithm aims to cover the obstacle of free space on the map with sampled regions. The sampling process continues by generating overlapping rectangular areas until the termination condition is satisfied. After the sampling process, we run Dijkstra's search algorithm on the graph to find all possible "optimal" paths starting from any arbitrary node to the goal node.

In the feedback motion control stage, MPC computes optimal (finite horizon) inputs that drive the dynamic robot towards the next node while satisfying the constraints, i.e., staying inside the region and not exceeding velocity and acceleration limits with guaranteed stability. In this algorithm, rather than constructing (open-loop) connected trajectories, we generate a sparse collision-free neighborhood graph.

A. Graph Generation

In the graph generation stage, we randomly sample a point in obstacle-free space, generate a node by creating a "large" region around the random sample, and finally connect the generated node to the graph by analyzing the intersections with the already existing zones. The process continues until the termination condition is satisfied. In a

Algorithm 1 MPC-Graph Generation

for $k = 1$ to K **do**
  do
    $q_{rand} \leftarrow$ UniformRandConf()
  while $q_{rand} \in WO$ and $q_{rand} \in \mathcal{B}$
  $Node_k \leftarrow$ GenerateRectRegion(q_{rand})
  $Node_k \leftarrow$ Expand($Node_k$)
  G.InsertNode($Node_k$)
  $\mathcal{B} \leftarrow \mathcal{B} \cup Node_k$
  if $TerminationSatisfied(G, \alpha, P_c)$ **then**
    Po = DijkstraAlgorithm(G)
    return Po
  end if
end for

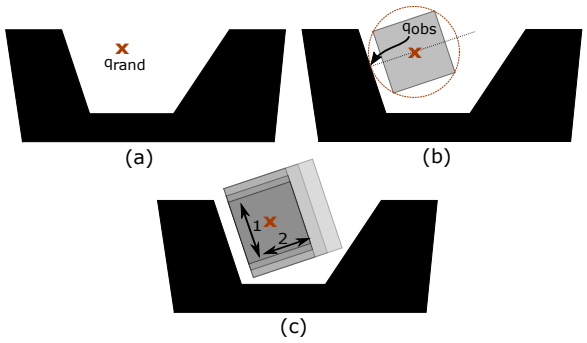

Fig. 2. Generation of a node: (a) initial map of the arena, (b) a square node is generated, (c) square node is expanded in discrete steps along directions indicated as 1 and 2

similar approach, Golbol et al. [17] cover the free space with rectangular regions connected in a tree-like manner for controller reusability in their reference governor based control approach. Since MPC performs the control policy computations in real-time for all possible convex constraint sets, in our work, we construct the graph using rectangular nodes that have arbitrary aspect ratios. Thus, we can obtain a more sparse graph.

The algorithm starts with randomly sampling the obstacle free space and generating nodes around these sampled points. At any time, the set of points covered by a node is denoted as $\mathcal{B}$:

$$\mathcal{B} = \bigcup_k Node_k \tag{4}$$

The algorithm continues to sample the free and non-covered space randomly and generates and expands a new rectangular node around the sample point. The new node is inserted in the graph, and an edge is generated between this node and every node overlapping with it. Edge weights are set to be the distances between the centers of the new node and the overlapping nodes.

In the algorithm, we implemented the termination condition presented in [14] which mainly estimates the quality of the coverage of the sampled space. If the algorithm satisfies the following condition graph generation stage terminates,

$$m \geq \frac{\ln(1 - P_c)}{\ln\alpha} - 1 \tag{5}$$

where m is the number of successive failures followed by the first success, P_c and α are user determined parameters that effect the density of the coverage of the map. If the termination condition in (5) is satisfied, we run a search algorithm on the graph to compute an optimal policy, and this policy is returned. Algorithm 1 details the whole process.

1) Node Generation: In order to generate a node around the point q_{rand}, first, the shortest distance between q_{rand} and obstacles is calculated. Formally, let WO_i be the i^{th} obstacle in the map, WO be the obstacle set, and q_{obs} be the point on the obstacle which is closest to q_{rand},

$$WO = \bigcup_i WO_i \tag{6a}$$

$$q_{obs} = \underset{q \in WO}{\arg\min} \|q - q_{rand}\|. \tag{6b}$$

Let also d_{min} be the shortest distance between q_{rand} and the closest point on the obstacle, q_{obs}, $d_{min} = \|q_{rand} - q_{obs}\|$. Then the hypothetical circle centered at q_{rand} with radius d_{min} is drawn as in Fig. 2b. By definition of d_{min}, this circle is guaranteed to be obstacle-free. After obtaining the circle, the largest square inside the circle with one edge perpendicular to the line segment connecting q_{rand} and q_{obs} is constructed. This procedure guarantees that the region covered by the square is obstacle-free.

2) Node Expansion: After generating a square region, we expand it to increase the sparsity and the coverage of the free space. Moreover, larger nodes enable the robot to move *faster* in the MPC-graph framework. Inside each node, MPC based control policy ensures that the robot strictly stays inside the boundaries of the active node. In order to enforce this constraint, the MPC algorithm fundamentally slows down the robot as it approaches a region boundary. In this context, with larger nodes, the robot stays longer in the same region; hence it moves for a longer time at high speed.

The procedure for expansion is as follows. First, the square node expands laterally to form a rectangle, as in Fig. 2(c), in discrete steps. At each discrete step, the current edge length of the rectangle parallel to direction 1 is multiplied by a constant factor ($\gamma = 1.2$ in this paper) until the node collides with an obstacle or limits of the arena. If a collision occurs, the algorithm reverts to the last expansion. The same process takes place in the perpendicular direction, as in Fig 2c, and finally, a larger rectangular node is obtained.

B. Graph Search

After the termination condition in (5) is satisfied, we run Dijkstra's search algorithm to find the optimal discrete planner for the graph. For each node, the policy returns the next node that the robot should go.

In the graph we constructed, G, there is an edge between two nodes if they overlap. Edge weight is the distance between the centers of the nodes. Thus, our approach optimizes the algorithm for the shortest average path length.

C. Motion Control

In the second stage of our MPC-Graph methodology, we calculate the "optimal" planning policy Po, which routes the robot from an arbitrary node to the $GoalNode$ in discrete steps. The last phase of the method is the motion control stage. Here, our goal is to smoothly and safely navigate the robot from its current node to the next node determined by the policy Po while satisfying both state and input constraints, which also involves strictly staying inside the current node for all time. Since the regions are drawn totally inside the obstacle-free space, this guarantees that no collision occurs (assuming the objects are static). The robot should also respect the imposed constraints such as speed (state) and acceleration (input) limits, which are very important for experimental robotics. In this work, we adopt a MPC based feedback policy for the execution of the motion control.

When the robot is in $CurrentNode$, and the target node is $NextNode$, reference point is chosen to be the centroid of their intersection for that region, $[x_r \;\; y_r]^T = \text{Centroid}(CurrentNode \cap NextNode)$. MPC stabilizes the system at the origin. Thus, we define a local reference frame $\mathcal{L}$ for $CurrentNode$ as

$$\begin{aligned} \bar{x} &= x - x_r \\ \bar{y} &= y - y_r. \end{aligned} \tag{7}$$

We rewrite the constraints in these coordinates. Using MPC, we calculate the optimal finite-horizon input sequence that satisfies the constraints and navigates the robot towards origin. We apply the first element of this sequence to the robot.

When robot enters the intersection area, $NextNode$ becomes the new $CurrentNode$, and the node determined by the policy Po for that node is the new $NextNode$. The same process recursively applies until the robot reaches the goal region. In the goal region there is no $NextNode$, and reference point is at q_{goal}. This procedure is given in Algorithm 2.

Algorithm 2 MPC-Graph Execution

$CurrentNode \leftarrow StartNode()$
$NextNode \leftarrow Po.\text{Next}(CurrentNode)$
$r \leftarrow \text{Centroid}(CurrentNode \cap NextNode)$
while q_{goal} **not** reached **do**
  if $q_t \in GoalNode$ **then**
    $r \leftarrow q_{goal}$
  else if $q_t \in NextNode$ **then**
    $CurrentNode \leftarrow NextNode$
    $NextNode \leftarrow Po.\text{Next}(CurrentNode)$
    $r \leftarrow \text{Centroid}(CurrentNode \cap NextNode)$
  end if
  $u_t \leftarrow \text{MPC}(q_t, r, CurrentNode)$
end while

IV. Implementation, Results and Conclusions

This section introduces the results obtained from the implementation of the MPC-Graph algorithm.

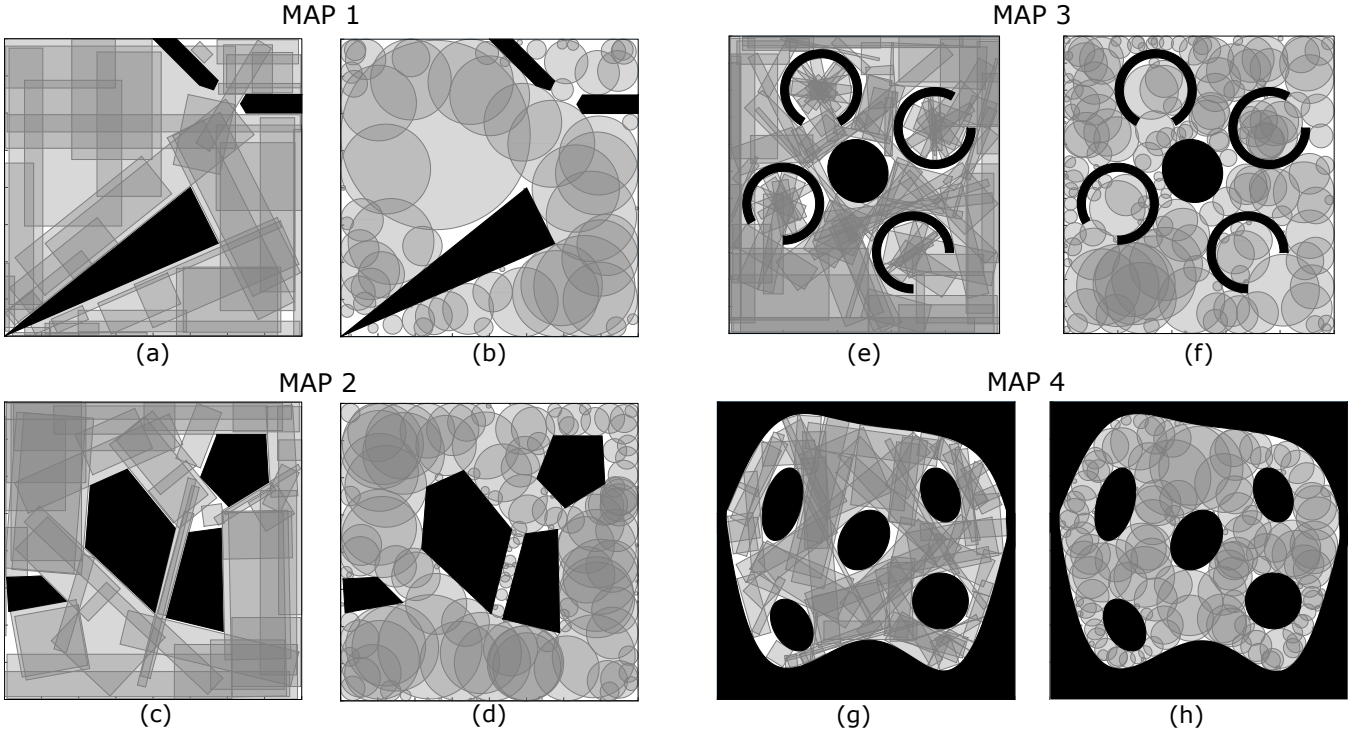

Fig. 3. On the different maps two different sampling based algorithms are tested. The proposed MPC-Graph algorithm generates 49, 51, 129 and 102 rectangular nodes in (a), (c), (e) and (g) respectively. SNG algorithm resulted in generating 78, 155, 192 and 148 circular nodes in (b), (d), (f) and (h) respectively.

TABLE I
COMPARATIVE RESULTS OF NODE GENERATION

Map	Algorithm	# Nodes	CPU Time
Map 1	SNG	90.85	0.14
	MPC-Graph	41.70	0.19
Map 2	SNG	125.60	0.17
	MPC-Graph	46.56	0.19
Map 3	SNG	199.51	0.82
	MPC-Graph	133.53	1.05
Map 4	SNG	138.63	1.12
	MPC-Graph	117.35	1.35

A. Robot Motion Model

In simulations, we model the robot as a fully actuated acceleration driven holonomic dynamic model in the presence of non-linear quadratic friction force acting on each axis. The state vector and the input vector of the model takes the form $q = [x\ y\ v_x\ v_y]^T$, and $u = [u_x\ u_y]^T$ respectively. In this context, the equations of motion for the robot model is

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} -\lambda\, v_x^2 + u_x \\ -\lambda\, v_y^2 + u_y \end{bmatrix} \tag{8}$$

where $\lambda = 0.7$. We assume that full state measurements are available in real-time. We discretize the continuous nonlinear dynamics of the system and uniform synchronous sampling of measurements with a sampling frequency of $f_s = 10Hz$ (or $T_s = 0.1s$).

In our control policy, we used quasi-infinite horizon MPC. The finite horizon length is $T_p = 1.5s$, which gives us enough degrees of freedom in enforcing the state and input constraints. We choose the weighting matrices for the objective cost function in (1) as an identity matrix, $Q = I_4$ and $R = I_2$. Then, by using the procedure presented in [20], we obtain the following state feedback gain K and terminal penalty matrix P,

$$K = \begin{bmatrix} 1.00 & 0 & 1.73 & 0 \\ 0 & 1.00 & 0 & 1.73 \end{bmatrix} \tag{9}$$

$$P = \begin{bmatrix} 185.28 & 0 & -152.19 & 0 \\ 0 & 185.28 & 0 & -152.19 \\ -152.19 & 0 & 168.32 & 0 \\ 0 & -152.19 & 0 & 168.32 \end{bmatrix}. \tag{10}$$

$$f(x, y) \leq c \tag{11a}$$
$$-2 \leq v_x, v_y \leq 2 \tag{11b}$$
$$-3 \leq u_x, u_y \leq 3 \tag{11c}$$

The constraints are given in equation (11). Note that equation (11a) is calculated for each node on the path.

B. Simulation Results

We implemented the methods in MATLAB and performed simulations on a desktop computer with Intel i7 3.6 GHz processor running Windows OS. To check the performance of the node generation stage of MPC-Graph, we implemented the SNG algorithm with the enhancement presented in [14], performed comparative analysis. We performed Monte-Carlo experiments on 4 different maps (# simulations = 1000, for each map), as illustrated in Fig. 3. We compared the algorithms in terms of sparsity and computational efficiency. Table I presents the Monte-Carlo simulation results.

In our simulations we choose $\alpha = 0.95$ and $P_c = 0.95$ since these values give satisfactory results in terms of map coverage and computational time. We adopted Map 1 from Yand and LaValle [14], which is a simple map composed of three polygonal obstacles. In this map MPC-Graph algorithm generates a more sparse graph compared to SNG algorithm ($\sim \%55$ reduction in # nodes), whereas the SNG creates its random map faster than the MPC-Graph algorithm ($\sim \%25$ faster). In addition, we tested both approaches on a more complicated map, Map 2 Fig. 3, composed of 4 polygonal obstacles with a relatively narrow path. In this scenario, while computation times are comparable to each other (SNG is $\%10$ faster than the MPC-Grapha), MPC-Graph illustrates a remarkable sparsity performance ($\sim \%63$ reduction in # nodes).

In order to provide a fair comparison, we also tested two other maps, Map 3 and Map 4, that are composed of curved (circular, elliptic etc.) obstacles and boundary (in Map 4). Qualitatively, the results are similar to the Maps 1 & 2, in the sense that the MPC-Graph algorithm generates more sparse graphs with the added cost of computational efficiency. We believe that offline computational performances of both methods are comparable, and both techniques provide possible times for real applications. Thus, due to the sparsity performance of our approach, the MPC-Graph method could be beneficial and powerful in different robotic applications.

Fig. 4 shows the execution of Algorithm 1 and Algorithm 2. Fig. 4a illustrates the graph structure and optimal policy generated by the graph search stage, and Fig. 4b shows the trajectory followed by the robot. The supplementary video visualizes velocity and acceleration plots of the robot while following the blue trajectory in Fig. 4b. The video shows that MPC successfully enforces the constraints on q_t and u_t while directing the robot to the goal configuration. Also, we measured the online computation time of MPC. CPU time of MPC for each iteration is at average $t_{CPU} = 0.08s$, which is less than our sampling time.

V. DISCUSSION AND FUTURE WORK

In this paper, we have introduced a new sampling-based motion planning method, MPC-Graph. The proposed method generates a sparse graph structure by randomly sampling the obstacle-free space and creating nodes that have rectangular shapes with different respect ratio to increase the coverage and sparsity. Then, MPC guides the robot from start location to goal location, ensuring that it strictly stays in the obstacle-free rectangular regions and velocity and input limits with guaranteed stability. We have tested our algorithm in different 2D environments and compared it to an existing sampling-based algorithm, SNG. Simulations show promising satisfactory results that our algorithm generates fewer nodes in a comparable amount of time.

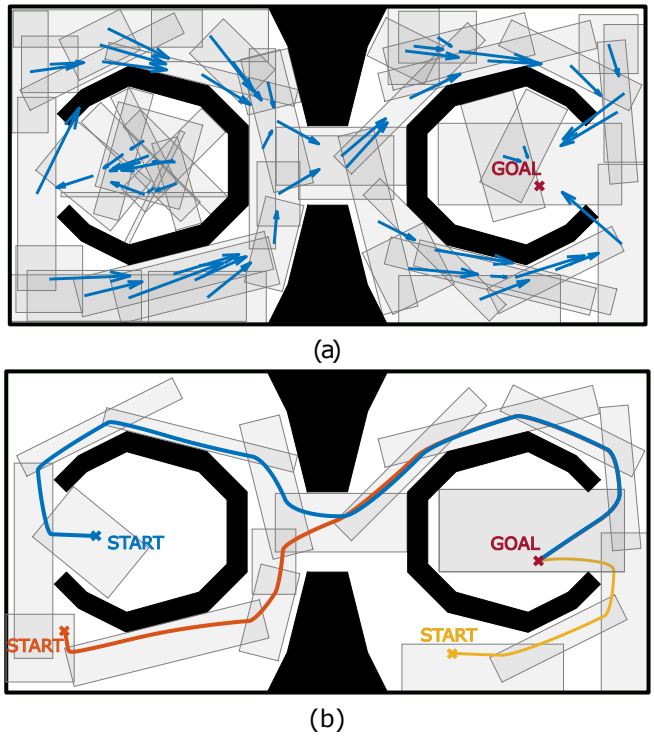

Fig. 4. (a) Representation of how rectangular regions are connected. Blue arrow indicates the next node in the policy. (b) Execution of the Algorithm 2. Blue, red and yellow lines show the trajectory that followed by the dynamic robot with three different starting points. Note that the nodes that are in the path are shown only.

We tested our algorithm on a holonomic non-linear robot model in the presence of quadratic friction force. In the future, we are planning to extend our approach to handle non-holonomic and under-actuated robot models. MPC-Graph can also be generalized to include higher dimensional spaces. For example, in 3D environments, prismatic zones can be generated to cover the obstacle-free space.

CPU time of MPC at each iteration suggests that we should further decrease the CPU time for the MPC-Graph algorithm to apply the algorithm in less powerful embedded platforms. Several researchers [21], [22] proposed different solvers that can reduce the computation time for MPC based control policies. We are planning to adopt a similar approach to reduce the CPU time of our algorithm, and hence improve the applicability of the algorithm in a wide range of robotic applications.

ACKNOWLEDGMENT

This work was supported by The Scientific and Technological Research Council of Turkey (TUBITAK) through Project 118E195, O. K. Karagoz's and S. Atasoy's scholarships. We also thank Klaus Werner Schmidt and Ferhat Golbol for their helpful insights and discussion.

REFERENCES

[1] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 952–964, 2016.

[2] E. Kayacan, E. Kayacan, H. Ramon, and W. Saeys, "Learning in centralized nonlinear model predictive control: Application to an autonomous tractor-trailer system," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 1, pp. 197–205, 2014.

[3] N. D. Wallace, H. Kong, A. J. Hill, and S. Sukkarieh, "Receding horizon estimation and control with structured noise blocking for mobile robot slip compensation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1169–1175.

[4] M. Saska, Z. Kasl, and L. Přeucil, "Motion planning and control of formations of micro aerial vehicles," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 1228–1233, 2014.

[5] D. H. Shim, H. J. Kim, and S. Sastry, "Decentralized nonlinear model predictive control of multiple flying robots," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, vol. 4. IEEE, 2003, pp. 3621–3626.

[6] D. D. Dunlap, C. V. Caldwell, and E. G. Collins, "Nonlinear model predictive control using sampling and goal-directed optimization," in *2010 IEEE International Conference on Control Applications*. IEEE, 2010, pp. 1349–1356.

[7] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, "Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 4, pp. 755–764, 2002.

[8] T. Kraus, H. J. Ferreau, E. Kayacan, H. Ramon, J. De Baerdemaeker, M. Diehl, and W. Saeys, "Moving horizon estimation and nonlinear model predictive control for autonomous agricultural vehicles," *Computers and electronics in agriculture*, vol. 98, pp. 25–33, 2013.

[9] Y. Zhang and J. Yi, "Velocity field-based maneuver regulation of autonomous motorcycles," *IFAC Proceedings Volumes*, vol. 43, no. 18, pp. 385–392, 2010.

[10] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *The International Journal of Robotics Research*, vol. 18, no. 6, pp. 534–555, 1999.

[11] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[12] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[13] L. Yang and S. M. LaValle, "A framework for planning feedback motion strategies based on a random neighborhood graph," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 544–549.

[14] L. Yang and S. M. Lavalle, "The sampling-based neighborhood graph: An approach to computing and executing feedback motion strategies," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 419–432, 2004.

[15] R. Tedrake, "Lqr-trees: Feedback motion planning on sparse randomized trees," 2009.

[16] E. Ege and M. M. Ankarali, "Feedback motion planning of unmanned surface vehicles via random sequential composition," *Transactions of the Institute of Measurement and Control*, p. 0142331218822698, 2019.

[17] F. Golbol, M. M. Ankarali, and A. Saranli, "Rg-trees: Trajectory-free feedback motion planning using sparse random reference governor trees," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6506–6511.

[18] M. Ozcan and M. M. Ankarali, "Feedback motion planning for a dynamic car model via random sequential composition," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Oct 2019, pp. 4239–4244.

[19] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[20] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205–1217, 1998.

[21] B. Stellato, V. V. Naik, A. Bemporad, P. Goulart, and S. Boyd, "Embedded mixed-integer quadratic optimization using the osqp solver," in *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 1536–1541.

[22] S. Richter, *Computational complexity certification of gradient methods for real-time model predictive control*. ETH Zurich, 2012.