2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Automatic Configuration of Multi-Agent Model Predictive Controllers based on Semantic Graph World Models

K. de Vos[1,*], E. Torta[1], H. Bruyninckx[1,2,3], C.A. López Martínez[1], M.J.G. van de Molengraft[1]

***Abstract*— We propose a shared semantic map architecture to construct and configure Model Predictive Controllers (MPC) dynamically, that solve navigation problems for multiple robotic agents sharing parts of the same environment. The navigation task is represented as a sequence of semantically labeled areas in the map, that must be traversed sequentially, i.e. a route. Each semantic label represents one or more constraints on the robots' motion behaviour in that area. The advantages of this approach are: (i) an MPC-based motion controller in each individual robot can be (re-)configured, at runtime, with the locally and temporally relevant parameters; (ii) the application can influence, also at runtime, the navigation behaviour of the robots, just by adapting the semantic labels; and (iii) the robots can reason about their need for coordination, through analyzing over which horizon in time and space their routes overlap. The paper provides simulations of various representative situations, showing that the approach of runtime configuration of the MPC drastically decreases computation time, while retaining task execution performance similar to an approach in which each robot always includes all other robots in its MPC computations.**

I. Introduction

Multi-robot deployment has become common in applications such as warehousing, manufacturing, and farming. This requires motion coordination approaches that guarantee safe navigation of all robotic agents that share the same, often dynamic, environment. These environments typically contain an inherent structure, such as the aisles and intersections in a warehouse setting. The environment dynamics only occur through obstacles that move through the environment.

In this work we present a method to dynamically configure Model Predictive Controllers to solve local navigation problems based on the information represented in the semantic graph world model. Model Predictive Control (MPC) is a motion control approach that solves a constrained optimisation problem at every sample time, and applies the control input in a receding horizon fashion. Since constraints and objectives of agents can be composed independently of each other, MPC is applicable for solving the navigation coordination challenges in the above-mentioned applications, in which agents and their task descriptions are expected to be heterogeneous, and environments dynamic.

We show that constraints and objectives of the MPC problem can be derived from a semantic graph world model that represents the environment's layout, geometry, and area semantics. We argue that by changing the area semantics, the layout or the geometry of the environment as represented in the graph world model, the robot will behave differently in different areas in the environment for several reasons, such as closeness to walls, presence of obstacles, constraints on maximal or minimal speed, etc. The navigation task of a robot (its route), is represented as a sequence of areas it has to drive through. While executing their task the MPC controller ensures that individual robots coordinate their motions such that collisions with elements of the environment and each other are avoided. As reported in e.g. [1], communication can significantly boost the coordination performance, especially in complex dynamic environments. However, communication is not always necessary, since the risk of collision varies as robots get closer together or further away from each other. More importantly, when the number of agents becomes large, the broadcasting of communication messages between all agents in an environment may become infeasible.

Our contribution is twofold. Firstly we propose a semantic graph world model that captures both the topology and the geometry of the environment in the form of a property graph. The topology of the graph is designed to allow agents to dynamically and automatically derive which constraints and objective functions are applicable to their MPC controller given the current state of the system as represented in the graph. Secondly, we propose an algorithm to dynamically determine which agents should coordinate their movements which results in a reduction of the number of constraints in the MPC controller of each agent with a significant reduction in communication and computation time of the MPC controller.

Related work. Different methods can be used to solve multi-agent navigation problems. Planning is often performed offline assuming static or predictable system's configurations (see [2]–[4] for examples). When deviations between the plan and the system's state are observed, a new plan can be computed and executed. Contrary to planning, reactive approaches (see [5]–[7] for examples) tend to adapt to rapid changes in the environment at the expense of disregarding global optimality, deadlock or livelock freeness. A combination of planning and reactivity is beneficial for reliable navigation in multi-robot systems. Model Predictive Control (MPC) is a method that combines planning and predictions over a defined horizon with reactivity to the current system's state therefore it is regarded as a viable method to address multi-robot navigation problems. In MPC-based approaches, trajectories are computed by solving a constrained optimization problem. The objective function is commonly a measure of the desired performance of the

[1] Department of Mechanical Engineering, Eindhoven University of Technology, The Netherlands
[2]Department of Mechanical Engineering, KU Leuven, Belgium
[3]Flanders Make, Leuven, Belgium
[*]E-Mail: k.d.vos at tue.nl

system. Constraints typically arise from the system dynamics, hardware limits, and (safety-) requirements. Within the context of mobile-robot navigation, different MPC formulations have been proposed, e.g., [8], [9]. Similarly to the scenarios addressed by this work, [8] consider robot navigation in environments with an inherent structure. They introduce the use of separating hyperplane constraints for collision avoidance which this paper adopts in the proposed MPC formulation as well. On the same line but for vessels navigation, [9], designed a distributed nonlinear MPC for the navigation of vessels at a canal intersection.

MPC approaches, however, often suffer from issues with scalability as the number of agents grows large [10]. To address scalability issues, many authors have proposed decentralizing the MPC problem [8], [9], [11]. This, however, comes at the cost of an increased need for communication due to the coupling between sub-problems resulting from agents sharing the same workspace. Several works have proposed reducing the complexity of the MPC problem itself. The proposed methods typically use the current context to reduce the number of optimisation variables and/or constraints, or to decouple problems which are unlikely to influence each other. The authors of [11] propose a framework in which a multi-agent MPC trajectory planner is deployed alongside a communication policy that, using a multi-agent reinforcement learning approach, has learned when and with whom the agent should communicate to avoid collisions. The authors of [8] combine a global path planner with a local trajectory generator. This approach allows them to divide the task of navigating a large and complex environment into smaller frames, reducing the scope and complexity of subsequent optimal control problems. This is in line with the method presented in our paper however, the division in smaller frames is here achieved by interpreting the system state as provided by a graph world model. Prior work has explored the use of graph-based semantic world models to automatically generate semantic maps for localization [12] or navigation [13], in this paper we propose a new method to automatically (re)configure the (MPC-based) controller itself by automatically determining the applicable constraints and objective function.

II. Method

A. MPC Formulation

We cast the solution of the local navigation problem as an MPC problem. Since, within the MPC framework, collisions between the intended motion trajectories of individual agents can be predicted and acted upon before they occur. We consider an environment in which a set of agents $\mathbb{A}$ is deployed with $\mathbb{F}_o \subseteq \mathbb{A}$ a subset of agents who should coordinate their movements. Given a set of Objectives $\mathbb{O}_i$ and a set of environment elements, such as walls and obstacles, $\mathbb{W}_i$ for each subset $\mathbb{F}_o$ of agents A_i the multi-agent MPC problem can be formulated as:

$$\min_{\mathbf{x},\mathbf{u}} \sum_{i=0}^{|\mathbb{F}_o|} J_i(\mathbf{x}_i, \mathbf{u}_i, m_i, t), \tag{1}$$

Subject to:

$$\mathbf{x}_i(t+k+1) = \mathbf{f}_i(\mathbf{x}_i(t+k), \mathbf{u}_i(t+k)), \tag{2}$$
$$\mathbf{x}_i(t) = \mathbf{x}_{i,init}, \tag{3}$$
$$\mathbf{x}_{i,min} \leq \mathbf{x}_i(t+k) \leq \mathbf{x}_{i,max}, \tag{4}$$
$$\mathbf{u}_{i,min} \leq \mathbf{u}_i(t+k) \leq \mathbf{u}_{i,max}, \tag{5}$$
$$\text{dist}\left(A_i\left(t+k\right), w\right) > 0, \quad \forall w \in \mathbb{W}_i, \tag{6}$$
$$\text{dist}\left(A_i\left(t+k\right), A_j\left(t+k\right)\right) > 0, \quad \forall A_j \in \mathbb{F}_j \wedge j \neq i, \tag{7}$$

in which $\mathbf{f}_i$ describes the dynamic model of agent i, which within the context of this paper is assumed to reflect a kinematic unicycle model. Furthermore, $\mathbf{x}_i \in \mathbb{X}_i$ and $\mathbf{u}_i \in \mathbb{U}_i$ represent the state and input of agent i respectively. The state of agent i describes its longitudinal velocity and its Cartesian position and orientation w.r.t a global reference frame. The control inputs of agent i are the acceleration in longitudinal direction and its angular velocity.

Objective: We employ a standard cost function structure, containing intermediate costs l_i and final costs n_i,

$$J_i(\mathbf{x}_i, \mathbf{u}_i, m_i, t) := \sum_{k=0}^{N_t} l_i(\mathbf{x}_i(t+k), \mathbf{u}_i(t+k)) \tag{8}$$
$$+ n_i(\mathbf{x}_i(t+N_t)), \tag{9}$$

with t the current time step, and $\mathbf{x}_i \in \mathbb{X}_i$ and $\mathbf{u}_i \in \mathbb{U}_i$ the predicted state-trajectory and optimised input-trajectory of agent i respectively computed over a horizon of length $N_t \in \mathbb{N}$. The intermediate cost function is defined as

$$l_i(\mathbf{x}_i, \mathbf{u}_i, t) := \mathbf{u}_i^T \mathbf{R}_i \mathbf{u}_i, \tag{10}$$

with $\mathbf{R}_i$ a cost matrix of appropriate dimension that weights the system input of agent i. The terminal cost function rewards progress made within the prediction horizon and is thus defined as

$$n_i(\mathbf{x}_i, t, m_i) := \sum_{j=0}^{|\mathbb{O}_i|} w_o \left[\ d_{i,j}^T\ \mathbf{Q}_{i,j}\ d_{i,j} + \mathbf{q}_{i,j}^T d_{i,j},\right] \tag{11}$$

with $d_{i,j}$ the (signed) distance of agent A_i from the j-th element of the set of objectives $\mathbb{O}_i$, with w_o weighing the objectives. The weights are determined through analysis of the solution in the previous iteration. All objectives corresponding to areas which were reached in the previous prediction, are weighted equally, the remaining objectives receive zero weight.

Constraints: For safety reasons, agents should not collide with any elements, e.g. walls, obstacles or virtually defined boundaries, in the environment, nor with each other. To represent the no-collision constraints (6) we utilize the separating hyperplane constraint [14]. This allows the specification of no-collision constraints between agents and obstacles described by convex shapes, however, puts no limitation on the convexity of the resulting drivable space. This opposed to, for example, formulating the non-collision constraints as a collection of linear constraints which only allows for describing a convex drivable space.

For each element $w \in \mathbb{W}_i$ with vertices $\mathbf{v}_w$, we formulate the no-collision constraints for a circular agent A_i with radius r_v as:

$$\mathbf{a}_{w,i}^T \mathbf{v}_{w,j} - b_{w,i} \geq 0, \qquad \forall \mathbf{v}_{w,j} \in \mathbf{v}_w, \tag{12}$$

$$\mathbf{a}_{w,i}^T \mathbf{x}(k) - b_{o,i} \leq -r_v, \qquad \forall k \in [0, N_t], \tag{13}$$

$$||\mathbf{a}_{w,i}||_2 \leq 1, \tag{14}$$

with $\mathbf{a}$ and b the normal vector to and offset of the hyperplane respectively. Since we approximate the robot's geometric footprint with a circle, the no-collision constraint (7) between any two agents i and j, with radii r_i and r_j respectively, can be formulated as

$$||\mathbf{x}_{k,i} - \mathbf{x}_{k,j}||_2 > (r_i + r_j), \qquad \forall k \in [0, N_t]. \tag{15}$$

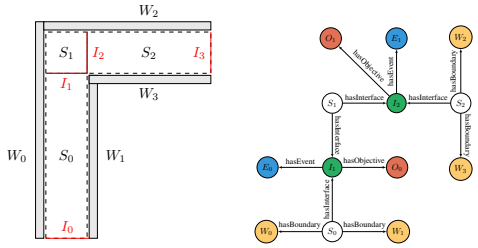
To encourage trajectories further away from obstacles and other agents, all no-collision constraints are augmented with soft constraints with an enlarged agent radius of r_{soft}.

B. Semantic Map

Similarly to [12], [13], the semantic map is a graph world model that describes the environment in which the agent is deployed at a topological level:

Definition 1 (Semantic Map): a graph $G = (V, E)$, with vertices V of type {Area, Boundary, Interface}, and edges E between vertices that show a spacial connection.

Knowledge of the environmental geometry is represented as a property of the vertices composing the graph. We assume each agent has access to an identical world model which contains correct information on the current state of the environment. We also assume that agents have the ability to recognize new or moved obstacles and are able to update the graph accordingly. A simple example environment, with corresponding graph world model, is depicted in Figure 1. The vertices of the graph represent semantic elements which we refer to as semantic map primitives. Three semantic primitives are identified: Areas, Interfaces, and Boundaries. **Areas** represent the drivable space for the agent, and are, therefore, the building blocks of the route of each agent. **Interfaces** model the connectivity between areas, i.e., areas are not directly connected through an edge in the graph but always through one interface node between them. Within the current area, interfaces are either active or inactive. Active interfaces are only those interfaces which connect the current area with the next one according to the route. Inactive interfaces are thus those interfaces that would lead the agent, if crossed, to areas that are not part of the current route. Crossing inactive interfaces should thus be prevented, through adding them as constraints in the local navigation problem. The navigation through an area is, furthermore, constrained through the addition of **Boundaries**. These can stem from physical elements such as walls or obstacles or can be defined virtually to prevent the agents from entering certain regions such as the space in front of emergency doors. For safety reasons, these are interpreted as hard constraints. We furthermore specify properties of semantic map primitives. These properties can stem from the geometry



(a) (b)

Fig. 1: (Left) Simplified structured environment consisting of three areas. (Right) Excerpt of a graph world model describing the environment on the left. The colors of the nodes signify their different types.

of those elements, which are encoded as lines and polygons, or can add additional information for the construction of the MPC problem. More specifically, interfaces provide information on the objective, events, and virtual boundaries. We define a mapping between the nodes, edges and properties of the graph and the configuration parameters of the MPC controller. In particular, the line geometry of nodes of type interface are mapped to the objective in (11), virtual boundaries are mapped to non-collision constraints to prevent the agent from entering areas outside of its route, events encode the transitions between areas, and are monitored to detect the agents entering the next area in the sequence, thus triggering a reconfiguration of the MPC controller.

C. Navigation Strategy

We assume agents are assigned a route, i.e., the sequence of areas to be traversed sequentially, which is planned independently of the plan other agents. The navigation task is thus decomposed into sub-tasks. Each sub-task requires the agent to reach the end of the area by reaching the corresponding interface, while avoiding all collisions with other agents or environmental elements. For example, consider again the environment depicted in Figure 1, and an agent located in Area S_0. Instead of formulating the navigation task as reaching a certain coordinate, the navigation task can be formulated by specifying a sequence of areas to be visited, in this case, the task specification is "*reach* I_3 *after crossing areas* S_0, S_1 *and* S_2". From their routes, the agents can dynamically configure their MPC controller.

Specifying the local navigation tasks as a set of subsequent sub-tasks, combined with the semantic map in Definition 1, allows the agent to reason about which elements in the world are relevant in the current area and for its current sub-task, e.g. walls or agents which are only connected to topologically far-away areas are not relevant at this time instance, which reduces the complexity of the navigation problem. This thus enables the simplification of the local navigation problems.

In other words, given the semantic map in Definition 1, the multi-agent navigation problem can be specified by considering a team of N_A agents that are deployed in an environment of which a graph world model that follows Definition 1 is available. Agent A_i is tasked with moving from its current

position $\mathbf{x}_{i,0}$ in area $S_{i,0}$ to its goal area $S_{i,g}$. A planned sequence of sub-tasks $\mathbf{P}_i = [S_{i,0} \dots S_{i,g}]$ to complete its task is provided to the agent, e.g. by a fleet-management system. It is assumed that, under nominal undisturbed conditions, execution of plan $\mathbf{P}_i$ leads to a successful completion of the task, i.e. that both task and plan are feasible. The task is deemed to be successfully completed once agent A_i is contained within area $S_{i,g}$ without having collided with any other agent A_j or with any element of the environment. Within our approach, we define that the index m_i of an agent A_i corresponds to the index of the area in the plan that is currently traversed by the agent. The objective of agent A_i is the objective encoded by the interface that connects its current area S_i with the next area of its route. The set of hard environmental constraints $\mathbb{W}_i$ to which the agent A_i is subjected contains only those (virtual-) boundaries which are linked via vertices to its current area S_i. The inter-agents constraints are formulated between couples of agents $c_i := (A_i, A_j)$ which are contained in the same area. Once the agent reaches the currently active interface, and its event is triggered, the agent has entered the area succeeding its current area, the objective and constraints are updated accordingly. To decrease the influence of the discrete transitions in objective and constraints between the current and the next area, a semantic horizon $\mathbf{h}$ is introduced, which acts as a look-ahead on the objectives and constraints of the next N_h areas in the pre-specified planning. It effectively means, that not only does the agent take into account the constraints and objectives in the current area S_{i,m_i}, but it, furthermore, takes into account the environmental elements and objectives which relate to the next N_h areas of the pre-specified route. The sets of coordinating agents, and thus the set of agents controlled by each MPC controller, are determined by checking the overlaps between their semantic horizon with length N_{hA}. Agents will cooperate if there is a common area in this semantic horizon. The problems from sets of agents which do not show overlap in their horizons can be solved separately. Figure 2 depicts the semantic horizon, and corresponding constraints, for an example scenario. The algorithm for retrieving relevant elements and the algorithm for retrieving the relevant agent configuration, given the graph world model K, are reported on in more detail in algorithms 1 and 2 respectively, these algorithms are called at startup, and thereafter every time one of the agents leaves its current area and enters the next.

III. Validation

In order to validate the method we define four distinct scenarios. In each scenario, a number of robotic agents is tasked with completing the route they receive at the start of the simulation. The difference between scenarios is the amount of agents in the environment as well as the expected amount of interaction required to complete the pre-specified plan, i.e. the amount of overlap in the plans. For each scenario we perform multiple simulations with the starting position of each agent in the first area of the semantic plan randomly initialized at every simulation. The scenarios

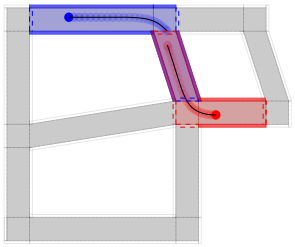


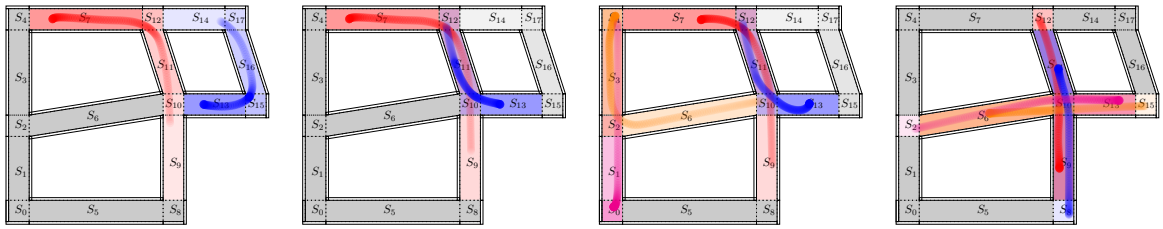
Fig. 2: Illustration of two agents with overlapping semantic horizons with $N_h = 2$. Red and blue polygons illustrate the environmental no-collision constraints of the red and blue agent respectively. Purple polygons show the overlap between the two. Dashed lines illustrate the virtual boundary constraints.

Algorithm 1 Relevant element retrieval for agent A_i

Input: K, $\mathbb{P}_i$, m_i, N_h
for $j \leftarrow m_i$ to $m_i + N_h$ **do**
  $\mathbb{B}_{i,j} \leftarrow$ Boundaries from K related to $\mathbb{P}_i[j]$
  $\mathbb{V}_{i,j} \leftarrow$ Interfaces from K related to $\mathbb{P}_i[j] \wedge \neg\mathbb{P}_i[j \pm 1]$
  $\mathbb{W}_{i,j} \leftarrow \mathbb{B}_{i,j} \cup \mathbb{V}_{i,j}$
  $\mathbb{I}_{i,j} \leftarrow$ Interfaces from K related to $\mathbb{P}_i[j] \wedge \mathbb{P}_i[j+1]$
  $\mathbb{O}_{i,j} \leftarrow$ Objectives from K related to $\mathbb{I}_{i,j}$
  $\mathbb{E}_{i,j} \leftarrow$ Events from K related to $\mathbb{I}_{i,j}$
Return: $\mathbb{W}_i$, $\mathbb{I}_i$, $\mathbb{O}_i$, $\mathbb{E}_i$

are illustrated in Figure 3. In scenario 1, two agents are deployed within the environment. Their plans do not overlap. In scenario 2, similar to scenario 1, two agents are deployed, but their plans show partial overlap. Scenario 3 involves four agents with partially overlapping plans, but agents do not have overlapping plans with all other agents at the same time. Finally, in scenario 4, four agents arrive at an intersection at approximately the same time.

The performance of the algorithm in the scenarios is evaluated in three different configurations, firstly a configuration in which agents do not cooperate (**N**ever), secondly a configuration in which agents always cooperate (**A**lways), and finally the proposed approach in which the sets of coordinating agents are formed dynamically (**D**ynamic), i.e. using Algorithm 2. Results from each scenario are averaged over all simulation runs of that scenario. The first metric, to evaluate the performance of the proposed approach, is the task completion time of each agent. This time is computed as the time it takes for the last agent in each run to reach their final mode. We report the minimum, maximum and average over all runs and configurations of each scenario. We, furthermore, report on the time it takes for the solver to compute a solution, and the correctness and feasibility of these solutions. In the case of computation time, we report on solver time, and configuration time separately, the latter including the time required to query the graph world model and set up the MPC problem. Note that at each time step we record the maximum time over all currently active solver instances, since each solver can be executed in parallel.



1 2 3 4

Fig. 3: The tested scenarios 1-4. Agents are initialized in the darkest shade of their respective color, and are tasked with navigating through the lighter shades. Overlaid are the resulting trajectories of one of the runs of the **D** configuration.

Algorithm 2 Relevant agent configuration retrieval

Input: K, $\mathbb{A}$, $\mathbb{P}$, $\mathbf{m}$, N_{sh}
$c \leftarrow \emptyset, \quad \mathbb{A}_f \leftarrow \emptyset$
Find couples of agents which have overlap
for $i \leftarrow 0$ to N_A **do**
  $\mathbf{h}_i \leftarrow \mathbb{P}_i[m_i : m_i + N_{hA}]$
  for $j \leftarrow i+1$ to N_A **do**
    $\mathbf{h}_j \leftarrow \mathbb{P}_j[m_j : m_j + N_{hA}]$
    if $\mathbf{h}_i \cap \mathbf{h}_j \neq \emptyset$ **then**
      $c \leftarrow c \cup \{(A_i, A_j)\}$
      $\mathbb{A}_f \leftarrow \mathbb{A}_f \cup \{A_i\} \cup \{A_j\}$
Expand sets of relevant agents
$\mathbb{F}^k \leftarrow c$ ▷ Sets at iteration 0
while $\mathbb{F}^k \neq \mathbb{F}^{k+1}$ **do**
  $\mathbb{F}^{k+1} \leftarrow \emptyset$
  for $\mathbb{F}_i$ in $\mathbb{F}^k$ **do**
    for $\mathbb{F}_j$ in $\mathbb{F}^{k+1}$ **do**
      if $\mathbb{F}_i \cap \mathbb{F}_j \neq \emptyset$ **then**
        $\mathbb{F}_i \leftarrow \mathbb{F}_i \cup \mathbb{F}_j$
    $\mathbb{F}^{k+1} \leftarrow \mathbb{F}^{k+1} \cup \{\mathbb{F}_i\}$
Add agents without overlap
$\mathbb{F} \leftarrow \mathbb{F} \cup (\mathbb{A} \setminus \mathbb{A}_f)$
Return: $\mathbb{F}$ ▷ Set of sets of relevant agents

We again report the minimum, maximum and average over all runs of each scenario. Collisions and infeasibilities are reported on through the number of runs in which they occur.

A. Implementation

The algorithm described in the paper, along with an example simulation environment is available upon request. The algorithm is implemented in Python, with the MPC controller implemented using the do-mpc package [15]. The resulting optimization problem is solved using CasADi [16] and the HSL_MA27 [17] solver in IPOPT[1] [18]. The graph querying is implemented in SPARQL using the RDFLib package [19]. Simulations are implemented using the built-in simulation in do-mpc, as well as in PyBullet [20]. Validation results[2] are obtained from the built-in simulation unless explicitly mentioned. The environment in which simulations were performed is depicted in Figure 3. The environment workspace is approximately 20 by 23 $[m]$ in size. The corridor width of each corridor is approximately 2.0 $[m]$. The configuration parameters of the MPC are summarized in Table I. Velocity and acceleration ranges represent what is expected in common commercial robots. During the experiments, it was assumed that agents are homogeneous, more specifically, that all agents can be contained in a cylinder with a base radius of 0.3 $[m]$. Dynamically, agents were described by a kinematic unicycle model.

[1]The max_iter parameter of IPOPT is set to 150 iterations.

[2]Results were obtained on an Intel Core i5 13500 processor running Ubuntu 20.04 under Windows Subsystem for Linux 2 (WSL2)

TABLE I: MPC configuration parameters in simulation experiments

Parameter	Symbol	value	
Longitudonal vel.	$[v_{min}, v_{max}]$	[0.0, 1.0]	$[m/s]$
Longitudonal acc.	$[a_{min}, a_{max}]$	[-0.5, 0.5]	$[m/s^2]$
Angular vel.	$[\omega_{min}, \omega_{max}]$	[-0.5, 0.5]	$[rad/s]$
Agent radius	r_v	0.40	[m]
Enlarged agent radius	r_{soft}	0.45	[m]
MPC time step	ΔT	0.50	[s]
MPC update frequency	f	4	[Hz]
Prediction horizon	N_t	25	[-]
Cost matrices	$\mathbf{R}$	diag(0.05, 0.5)	[-]
	$\mathbf{Q}$	1	[-]
	$\mathbf{q}$	10	[-]
Semantic Horizons	N_h, N_{hA}	2, 1	[-]

B. Results

For visualization purposes, an example realization of a run of the **D** configuration of each scenario has been overlaid in Figure 3. The remaining of this section presents the quantification of the performance metrics over 25 simulations per scenario and configuration.

1) Completion: For each scenario and configuration, Table II summarizes the number of non-successful task completions. It is observed that in scenario 1 all runs lead to successful task completion. In scenario 2 and 3, the majority of runs in the **N**ever configuration fails due to collisions. This is explained by the lack of collision constraints between agents when there is no cooperation. Two of the **A** and one of the **D** runs fail due to the solver concluding that the problem is infeasible. From Table II, it is apparent that scenario 4 is the most challenging, even in the **A** and **D** configurations. In both, at least 5 out of 25 runs fail due to minor collisions (overlapping footprints of < 0.01[m]) between sample points, which is a result of the time discretisation employed

TABLE II: Non-successful Completion of each scenario for the **A**lways, **D**ynamic and **N**ever configurations, due to infeasibilities/ collisions.

	N	**A** infeas. / coll.	**D** infeas. / coll.	**N** infeas. / coll.
1	25	0 / 0	0 / 0	0 / 0
2	25	2 / 0	1 / 0	1 / 17
3	25	0 / 0	0 / 0	0 / 25
4	25	0 / 5	0 / 6	0 / 25

TABLE III: Completion time, in time steps, of completed runs of each scenario for the **A**lways, **D**ynamic and **N**ever configurations.

T_{task}	**A** [min, avg, max]	**D** [min, avg, max]	**N** [min, avg, max]
1	[74, 95.2, 143]	[78, 96.0, 138]	[78, 96.0, 138]
2	[84, 121.1, 161]	[84, 114.0, 142]	[78, 102.8, 121]
3	[98, 133, 167]	[97, 128.5, 155]	[84, 124.7, 162]
4	[74, 101.7, 140]	[74, 101.4, 140]	[81, 93.6, 116]

within the MPC formulation. All runs in the **N** configuration fail due to collisions between agents, again due to the lack of no collision constraints.

2) Completion Time: The task completion time for all scenarios and configurations expressed in time steps is reported in Table III. It is apparent, from the average completion time, that the results obtained in the **A** and **D** configuration are largely comparable to each other. The **N**ever configuration does perform better, leading to shorter time to completion, however, this approach leads to the highest number of observed collisions, both explained by the lack of no-collision constraints in the MPC formulation. Which, in this case, allows for trajectories that do intersect each-other.

3) Computation Time: The MPC computation times are reported in Table IV. The average number of sets of coordinating agents, and thus MPC controllers, and the average number of agents within these sets, are reported in Table VI. From Table IV it is observed that the MPC computation times of the **D** configuration are, on average, lower than the **A** configuration, but higher than the **N** configuration. In scenario 1, no plan overlap occurs, which leads to the **D** configuration performing, on average, the same as the **N** configuration. In scenario 4, four agents arrive at the same intersection leading to similar performance between the D and N configuration, still better than the A configuration.

Table V summarizes the recorded configuration times. When reconfiguration is necessary, i.e. when one of the agents controlled by the MPC controller crosses over into the next area, it is apparent that significant time is spent on reconfiguration of the controller. Further observations are that, on average, the **N** and **D** configurations perform better than the **A** configuration.

IV. Conclusions

The paper presented an approach to enable a set of mobile robots to navigate in environments with an inherent structure with such knowledge encoded in a semantic graph world model. The proposed algorithms allow to dynamically

TABLE IV: MPC computation time, in milliseconds, of completed runs of each scenario for the **A**lways, **D**ynamic and **N**ever configurations.

T_{mpc} [ms]	**A** [min, avg, max]	**D** [min, avg, max]	**N** [min, avg, max]
1	[29, 66, 195]	[14, 29, 93]	[14, 29, 89]
2	[26, 54, 345]	[13, 34, 250]	[14, 24, 243]
3	[68, 157, 656]	[15, 50, 326]	[17, 30, 124]
4	[62, 145, 781]	[15, 82, 768]	[14, 25, 107]

TABLE V: Configuration time, in seconds, of completed runs of each scenario for the **A**lways, **D**ynamic and **N**ever configurations.

T_{conf} [s]	**A** [min, avg, max]	**D** [min, avg, max]	**N** [min, avg, max]
1	[0.18, 0.25, 0.35]	[0.10, 0.14, 0.28]	[0.10, 0.14, 0.37]
2	[0.18, 0.25, 0.37]	[0.10, 0.20, 0.37]	[0.10, 0.15, 0.35]
3	[0.36, 0.50, 0.76]	[0.10, 0.19, 0.60]	[0.10, 0.15, 0.91]
4	[0.35, 0.53, 0.80]	[0.10, 0.35, 0.76]	[0.10, 0.15, 0.48]

TABLE VI: Average number of sets of coordinating agents/average number of agents in these sets in each scenario for the **A**lways, **D**ynamic and **N**ever configurations.

$\mathbb{F}$	**A** avg nr. / avg size	**D** avg nr. / avg size	**N** avg nr. / avg size
1	1.0 / 2.0	2.0 / 1.0	2.0 / 1.0
2	1.0 / 2.0	1.6 / 1.4	2.0 / 1.0
3	1.0 / 4.0	3.2 / 1.3	4.0 / 1.0
4	1.0 / 4.0	3.0 / 1.9	4.0 / 1.0

(re)configure a Model Predictive Controller (MPC) for a set of agents. The derived constraints of the MPC problem relate to environmental elements that are relevant to each of the agents described in the MPC problem as well as the inter-agent no collision constraints. We show, in simulation, that the method is applicable, and that the proposed dynamic cooperation strategy results in decreased computation times when multiple agents are in a situation in which the algorithm identifies more than one set, i.e. in situations in which the plans of some of the agents do not overlap. Otherwise, the computational performance of the "*always cooperate*"-configuration and "*dynamic cooperation*"-configuration are similar. Furthermore, the "*dynamic cooperation*"-configuration exhibits similar performance (other than computation time) to the "*always cooperate*"-configuration across all tested scenarios. Future work will look into addressing a hybrid coordination approach where discrete approaches that could handle certain situation more efficiently or robustly, like scheduling at an intersection, can be combined with MPC controllers for continuous motion control. The challenges related to the automatic configuration and instatiation of such controllers based on graph world models remain, as well as their integration with continuous layers, here addressed by the MPC controller.

Acknowledgments

This work has been executed as part of the Semantic navigation for Teams of Open World Robots (TOWR) research project and is part of the ICAI EAISI FAST lab

References

[1] L. E. Parker, "Multiple mobile robot systems," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 921–941.

[2] T. Standley and R. Korf, "Complete algorithms for cooperative pathfinding problems," in *IJCAI*, Citeseer, 2011, pp. 668–673.

[3] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, 2015, ISSN: 0004-3702.

[4] M. Čáp, P. Novák, A. Kleiner, and M. Selecký, "Prioritized planning algorithms for trajectory coordination of multiple mobile robots," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 835–849, 2015.

[5] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[6] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The international journal of robotics research*, vol. 17, no. 7, pp. 760–772, 1998.

[7] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1928–1935.

[8] T. Mercy, E. Hostens, and G. Pipeleers, "Online motion planning for autonomous vehicles in vast environments," in *2018 IEEE 15TH International workshop on advanced motion control (AMC)*, IEEE, 2018, pp. 114–119.

[9] L. Ferranti, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, "Coordination of multiple vessels via distributed nonlinear model predictive control," in *2018 European Control Conference (ECC)*, IEEE, 2018, pp. 2523–2528.

[10] R. Firoozi, L. Ferranti, X. Zhang, S. Nejadnik, and F. Borrelli, "A distributed multi-robot coordination algorithm for navigation in tight environments," *arXiv preprint arXiv:2006.11492*, 2020.

[11] Á. Serra-Gómez, B. Brito, H. Zhu, J. J. Chung, and J. Alonso-Mora, "With whom to communicate: Learning efficient communication for multi-robot collision avoidance," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 11 770–11 776.

[12] R. W. M. Hendrikx, P. Pauwels, E. Torta, H. J. Bruyninckx, and M. J. G. van de Molengraft, "Connecting semantic building information models and robotics: An application to 2d lidar-based localization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 654–11 660.

[13] P. Pauwels, R. de Koning, B. Hendrikx, and E. Torta, "Live semantic data from building digital twins for robot navigation: Overview of data transfer methods," *Advanced Engineering Informatics*, vol. 56, p. 101 959, 2023, ISSN: 1474-0346.

[14] T. Mercy, W. Van Loock, and G. Pipeleers, "Real-time motion planning in the presence of moving obstacles," 2016, pp. 1586–1591.

[15] S. Lucia, A. Tatulea-Codrean, C. Schoppmeyer, and S. Engell, "Rapid development of modular and sustainable nonlinear model predictive control solutions," *Control Engineering Practice*, vol. 60, pp. 51–62, Mar. 2017.

[16] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[17] *The HSL mathematical software library*, UK Research and Innovation. Accessed: Jul. 28, 2023, 2023. [Online]. Available: https://www.hsl.rl.ac.uk/.

[18] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, 2006.

[19] *RDFLib a pure Python package for working with RDF*, RDFLib Team. Accessed: Jul. 28, 2023. [Online]. Available: https://rdflib.readthedocs.io/.

[20] E. Coumans and Y. Bai, *PyBullet, a Python module for physics simulation for games, robotics and machine learning*, http://pybullet.org, 2016–2021.