Towards Reliable Benchmarking for Multi-Robot Planning in Realistic, Cluttered and Complex Environments

Simon Schaefer[1], Luigi Palmieri[2], Lukas Heuer[2], Niels van Duijkeren[2], Ruediger Dillmann[1], Sven Koenig[3], Alexander Kleiner[2]

***Abstract*— Multi-robot planning and coordination is a hard task to solve particularly in cluttered and complex environments. Several methods exist for solving such task. Due to the lack of adequate benchmarking tools, comparing these approaches and judging their suitability for use in realistic scenarios is hardly possible. To this end, in this work we propose a novel benchmark toolchain that aims to close this gap. Differently from the related works, our benchmark uses full-stack multi-robot navigation systems in realistic 3D simulated intralogistic and household environments. The usage of open-source frameworks ROS2, Gazebo and RMF gives the user the possibility to easily add novel robot platforms. The framework provides easy-to-use and to-extend abstractions, common metrics and interfaces to several well-known planning libraries for multi-robot systems. With all these features our framework successfully aids practitioners and researchers in comparing multi-robot planning and coordination algorithms against the state of the art.**

I. Introduction

Planning and controlling a fleet of autonomous robots is a challenging task, extremely hard in cluttered and dynamic environments. Those systems are controlled by a complex pipeline of components ranging from centralized path finders to local distributed controllers, each suffering of their own limitations. For instance, Multi-Agent Path Finding (MAPF) and generalized task assignment problems are typically NP-hard [8] [21], even for static environments. Many suboptimal but faster algorithmic solutions have been proposed, choosing the best solution for a certain application and a given set selection criteria is difficult. This is particularly true for fleets of robotic systems navigating in uncertain and dynamic environments.

Several works on benchmarking have been presented recently [4], [14], [18], [20]. However, they only consider a sub-part of the control pipeline for fleets of robots, most of them focus only on the path planning problem. With the goal of enabling practitioners and researchers to select the algorithms best suited for the needs of their robotic fleets, we propose a multi-robot planning and coordination benchmark toolchain that considers several planning and control layers (i.e., centralized and decentralized MAPF algorithms, single robot navigation systems composed of global and local planners). The open-source framework includes a set of relevant service-robot-oriented simulated environments, metrics, and interfaces to available state-of-the-art planners and navigation systems. Its modular structure and versatile interfaces facilitate its extension with further scenarios, algorithms, or additional functionalities.

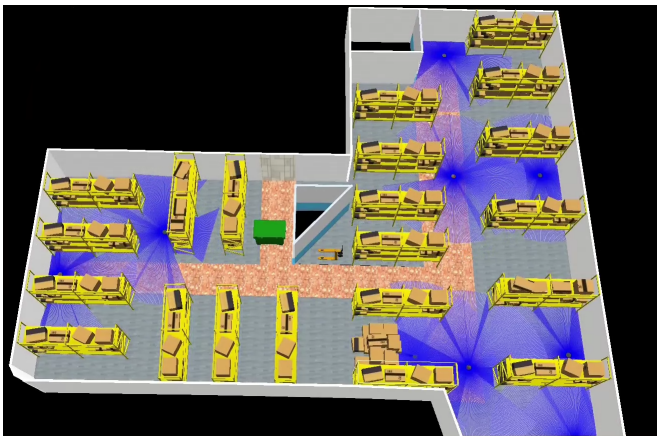

Fig. 1: Several robots moving in the warehouse environment. The individual laser scanners are visualized with blue rays.

II. Related Work

Benchmarking planning algorithms have received a lot of attention in the last years [3], [4], [6], [9], [14], [16], [18], [20]. Numerous benchmarks have been presented for multi-robot planning and coordination [4], [14], [18], [20]. Stern et al. [18] discuss a benchmark called "Grid-based MAPF" from MovingAI [19], [20]. As the name suggests, different grid-based maps are supplied. Maps are always in 2D and each cell in a map is either blocked or not blocked for an agent. For each of these maps, various scenarios are provided. A scenario is a list of tasks, with a task being a tuple of start and target cells for which a path has to be found. The number of agents can be varied by selecting the desired amount of tasks from the scenario, up to several hundreds. For each task, an optimal path length is provided. On some of these maps, the authors in [19], [20] performed further analysis, allowing some estimation of their respective difficulty. This benchmark assumes mainly a perfect knowledge about the world, meanwhile our approach considers not only planning but also execution in realistic simulated environments (thus considering possible

[1]S. Schaefer and R. Dillmann are with the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany {mrp@simon-schaefer.net, ruediger.dillmann@kit.edu}

[2]L. Palmieri, L. Heuer, N. van Duijkeren, A. Kleiner are with Robert Bosch GmbH, Corporate Research, Stuttgart, Germany {luigi.palmieri, lukas.heuer, niels.vanduijkeren@de, alexandre.kleiner}@de.bosch.com.

[3]S. Koenig is with the Computer Science Department of the University of Southern California {skoenig@usc.edu}

This work was partly supported by the EU Horizon 2020 research and innovation program under grant agreement No. 101017274 (DARKO).

uncertainty). As opposed to the MovingAI maps, Asprilo [4] offers a full simulation environment. This framework is aimed specifically at intralogistic scenarios in warehouses. The world is represented by a 2D grid, similar to the MovingAI maps. Specifics of intralogistics are also modelled: these include shelves containing specific amounts of some product, which need to be brought to picking stations. Therefore agents can perform additional actions such as picking up and place down a shelf, instead of just moving from location to location. While the constraints can be defined using answer set programming (ASP), the robot motion model is rather simple and the focus lies on abstract representations of agents. Differently from our approach higher level control of robots is neglected. Flatland [1], [14] is a tool for benchmarking vehicle rescheduling problem (VRSP), but it is not well suited for broader robotics domains we have in mind. The environments they consider are 2D grid with some restrictions on transitions between cells: e.g., there is no type of cells that allows entering and exiting a cell from all directions, as one would expect for most houshold or intralogistics robots. Contrarily to Flatland, our approach considers more realistic robotic scenarios, in terms of environment representation and modelling of the systems to control.

Moreover in our approach, differently from all the others, to reduce the gap between simulation and real-world performances we make use of state-of-the-art-robot navigation frameworks, namely: ROS2 [13] with Gazebo [10], Navigation 2 (Nav2) [12], and the Robotics Middleware Framework (RMF) [15].

III. Benchmark implementation

In this section, we explain key decision in designing the benchmark suite and outline its architecture.

A. Software Architecture

Figure 2 provides an overview of `MRP-Bench`.

1) Starting Up: The workflow starts at the **RMF Traffic Editor** which can be used to generate a **Gazebo** world file with the intermediate step of a *building.yaml* description. Together with the *config.yaml*, this provides the necessary information for the **Bench Manager Node** to start the benchmark: e.g. number of robots, random seed, start and goals. Using the world file, Gazebo launches the simulated 3D environment. From this simulation, a binary costmap is obtained using raytracing. The *building.yaml* also contains a representation of the navigation graph, which denotes the lanes that robots may move in. From this navigation graph, another simpler occupancy grid is created. This occupancy grid serves as an input for the path planning algorithm. Some algorithms can also directly use the navigation graph.

2) Planning and Fleet Management: If the planning library has managed to create a schedule, the Bench Manager computes and saves performance metrics from the planning, converts the schedule into separate path requests for each agent, and continues with sending the path requests to the fleet server, which delivers them to the individual fleet clients. The path is formed by several waypoints that will be then given to the local navigation units.

3) Local Navigation: Together with a state publisher and the fleet client, a full Nav2 stack is spawned for each robot. We use the standard global and local planning algorithms provided by the main repository. The benchmark user is free to choose the most interesting planners for their scenarios. The Nav2 stack interpolates a local path between the waypoints of the provided high level path, controls the robots and in case of conflicts performs collision avoidance and local recovery. Ground truth position from the simulator can be used or the user can decide to run a SLAM algorithm. Currently, the benchmark operates under the assumption that robots progress from cell to cell with the same average speed.

4) Data Visualization and Collection: Robot poses are displayed on a map using the **RMF schedule visualizer**. While the agents are following their schedule, their states (e.g. poses and velocities) are recorded and can be analyzed later to gather additional metrics. Additionally the users can record more data in rosbag format.

All custom, self-written nodes are implemented in Python3. The architecture is heavily based on the ROS2 launch system.

IV. Evaluation Sub-System

In this section we detail the scenarios and the metrics included in the benchmarking suite. Those can be further extended by the user.

A. Scenarios

We provide three main environments, the *office*, the *airport* and *warehouse*, see Figures 4 and 3. Their main properties are shown in Table I. The warehouse and airport environments are the most complicated for planning algorithms, both due to their size and their layout with high traffic *main roads*.

Property	**Office**	**Warehouse**	**Airport Terminal**
Width	21.53 m	22.16 m	282.22 m
Height	12.05 m	27.07 m	64.35 m
Nav. graph			
Vertices	29	54	210
Edges	32	59	211
Occupancy grid			
Cells total	1025	3009	105700
Cells passable	333	788	7645
Cells impassable	692	2221	98055

TABLE I: Environments key facts. Grid statistics at 0.4 m grid resolution and two-way roads.

B. Metrics

There are two sets of metrics. The first one are related to planning performance and quality: *success rate*, *planning time*, *makespan* and *cost*. The second set is calculated offline by analyzing recorded data of the execution of the scenarios:

i) Execution time: The execution time is defined as the time it took for all agents to reach their goals. It is bounded by a pre-configured timeout value; once passed, the simulation will be interupted.

ii) Number of goals reached: This counts the number of agents that managed to reach the goal before the timeout. In

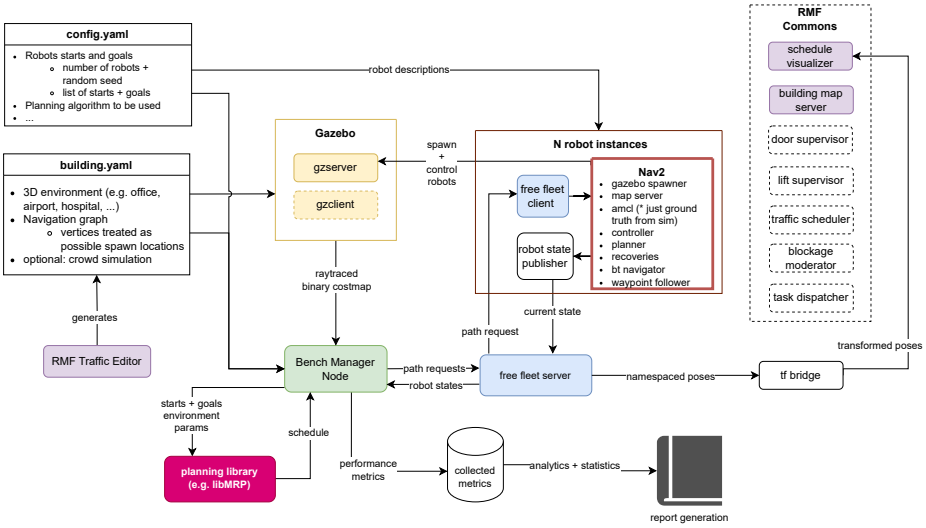

Fig. 2: A flow-chart of the proposed architecture. On the left, components that generate the scenario and set the configurations (e.g. starts, goals, planners to be used). In the middle the simulation and navigation frameworks, and the components (Bench Manager Node and free fleet server) orchestrating the benchmark. On the right the RMF components that are used for controlling the fleet and visualization.

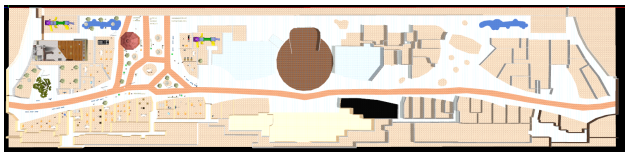

Fig. 3: The airport scenario is the largest and it offers the possibility to test the algorithms considering large automated ground vehicles.

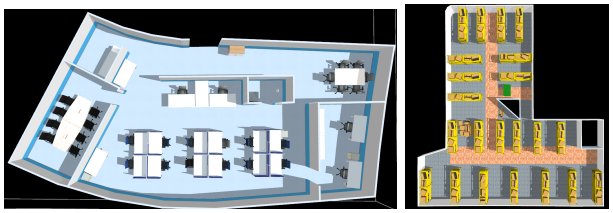

Fig. 4: **Left:** The office environment provides different homotopy classes and cluttered spaces. **Right:** The warehouse environment has been designed considering classical situations for robots working in intralogistic settings.

case the execution time is less than the timeout, this number will match the number of agents in the scenario.

iii) Minimum distance between two agents: At any point in time, we check the respective distances between all controlled agents. Accordingly, we can see whether some agents ran into each other, and if not, how close they got.

iv) Time blocked per agent and total: An agent is considered *blocked* if it has not progressed by at least one cell width within a certain time. For each agent, this metric calculates the amount of time spent blocked using a floating window approach.

We use the selected metrics for analysing the algorithms' performance in the following section.

V. Experiments and Results

To demonstrate the usefulness of the benchmark suite for gaining insights from different algorithms and scenarios, we performed experiments that compare several algorithms namely: distributed A* [5], CBS [17], ECBS [2] from *libMultiRobotPlanning* [7] and EECBS [11]. For all the experiments we use a differential drive robot model, and standard planners and parameters in Nav2. All the experiments run on a computer with Intel(R) Xeon(R) W-1270 CPU, 3.40GHz and 16Gb of memory.

A. Comparison of Algorithms Across Multiple Scenarios

For the main experiment, we modified three parameters: the random seed determining the starting positions of robots and tasks, the map used (office and warehouse) and the number of agents (5 and 9). In total, 864 experiments were performed, split evenly across the possible permutations of parameters. For algorithms supporting suboptimality, a factor of 1.2 was used.

Looking at the rate of success in planning a schedule within a timeout of 60 s, we obtain the results listed in Table II. We choose to group these results by the map being used, as there are significant differences in the planning success rate depending on the map being used.

Algorithm	Office	Warehouse
A*	100%	100%
CBS	99%	83%
ECBS	100%	97%
EECBS	100%	100%

TABLE II: Algorithms' success rate of finding a schedule within 60 s, on a basis of 108 experiments for each combination of map and algorithm.

CBS gives us an indication of the difficulty of maps: with still 99% success for office, this goes down to 83% for the warehouse. The time limit of 60 s was selected as a high, but still reasonable number for real-life applications. In fact, lower times may be desired and some algorithms like EECBS can easily provide these, while others like CBS take significantly longer.

In the next step, we compare how well the plans generated by the different algorithms actually perform during execution in the simulation. The normalized success rate describes all scenarios where all algorithms could calculate a schedule. For the overall success rate, cases where no schedule was found count as unsuccessful. Whether the execution would have been successful, had the schedule creation finished, is not known, but the practical result is the failure.

Table III shows the differences between algorithms. On the office map, the success rate is high for all algorithms and the differences are relatively small. On the more difficult warehouse map, differences become more distinct. The highest success rate, both relatively and overall, is obtained by ECBS. CBS, on the other hand, scores decently in the normalized column, but obtains the lowest success rate in the overall examination. This is due to the fact that CBS is the computationally heaviest of the four algorithms. In 17 of 108 scenarios, CBS does not find a schedule within the timeout of 60 s. For ECBS, this only occurs twice and for A* and EECBS, it is never the case. While the decentralized A* has a slightly lower success rate on the normalized warehouse column, it is not far below the other algorithms.

Algorithm	**Success Rate Normalized**		**Overall**	
	Office	**Wareh.**	**Office**	**Wareh.**
A*	95%	81%	95%	77%
CBS	93%	84%	92%	70%
ECBS	95%	89%	95%	85%
EECBS	95%	84%	95%	78%

TABLE III: Algorithms' success rate of completing a scenario (planning and execution) within the timeout of 5 min. Data based on 108 experiments, except for *warehouse, normalized*, which is based on 91 experiments.

B. Summary

In summary, our experiments suggest that using suboptimal algorithms is a viable approach for coordinating multiple robots in those settings. ECBS turned out to be faster than CBS and also delivered a higher success rate. EECBS is even faster and never failed to find a solution in our scenarios, at a small cost in the success rate. Overall, the local recovery feature offered by Nav2 may be sufficient in some cases to even use decentralized approaches such as A*. This applies especially if the environment is less static than in the scenarios we set up. If robots have to rely even more on local observations due to a rapidly changing environment, an approach using central planning is further disadvantaged.

VI. Conclusions

In this paper, we introduce MRP-Bench: a novel benchmark for multi-agent task assignment and path planning problems in realistic environments. The benchmark offers a set of scenarios, metrics ready to be used together with state-of-the-art algorithms. Its architecture has been designed such that more scenarios or additional robot models can be integrated with little effort. We provide interfaces to the most common frameworks for robot simulation, navigation and multi-robot planning algorithms. In the proposed prelimary experiments, we were able to demonstrate that data gathered using this benchmark allows us to judge the suitability of multi-agent algorithms for different scenarios. By going open source, i.e. `https://github.com/boschresearch/mrp_bench`, we hope that the research community will use this benchmark suite to evaluate novel algorithms and scenarios in the field of multi-robot planning. We warmly welcome contributions to this project.

References

[1] Welcome to flatland. `https://flatland.aicrowd.com/intro.html`. Accessed: 2022-3-31.

[2] M. Barer, G. Sharon, R. Stern, and A. Felner. Suboptimal variants of the Conflict-Based search algorithm for the Multi-Agent pathfinding problem. In *Seventh Annual Symposium on Combinatorial Search*, July 2014.

[3] C. Chamzas, C. Quintero-Pena, Z. Kingston, A. Orthey, D. Rakita, M. Gleicher, M. Toussaint, and L.E. Kavraki. Motionbenchmaker: A tool to generate and benchmark motion planning datasets. *IEEE Robotics and Automation Letters*, 7(2):882–889, 2021.

[4] M. Gebser, P. Obermeier, T. Otto, T. Schaub, O. Sabuncu, V. Nguyen, and T.C. Son. Experimenting with robotic intra-logistics domains. *arXiv:1804.10247 [cs]*, April 2018.

[5] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968.

[6] E. Heiden, L. Palmieri, L. Bruns, K.O. Arras, G.S. Sukhatme, and S. Koenig. Bench-mr: A motion planning benchmark for wheeled mobile robots. *IEEE Robotics and Automation Letters*, 6(3):4536–4543, 2021.

[7] W. Hönig. libMultiRobotPlanning. `https://github.com/whoenig/libMultiRobotPlanning`. Accessed: 2022-5-30.

[8] O. Kaduri, E. Boyarski, and R. Stern. Algorithm selection for optimal Multi-Agent pathfinding. *ICAPS*, 30:161–165, June 2020.

[9] L. Kästner, T. Bhuiyan, T.A. Le, E. Treis, J. Cox, B. Meinardus, J. Kmiecik, R. Carstens, D. Pichel, B. Fatloun, et al. Arena-bench: A benchmarking suite for obstacle avoidance approaches in highly dynamic environments. *IEEE Robotics and Automation Letters*, 7(4):9477–9484, 2022.

[10] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[11] J. Li, W. Ruml, and S. Koenig. Eecbs: A bounded-suboptimal search for multi-agent path finding. In *AAAI*, 2021.

[12] S. Macenski, F. Martín, R. White, and J.G. Clavero. The marathon 2: A navigation system. March 2020.

[13] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.

[14] S. Mohanty, E. Nygren, F. Laurent, M. Schneider, C. Scheller, N. Bhattacharya, J. Watson, A. Egli, C. Eichenberger, C. Baumberger, G. Vienken, I. Sturm, G. Sartoretti, and G. Spigler. Flatland-rl : Multi-agent reinforcement learning on trains, 2020.

[15] open-rmf. RMF demos. `https://github.com/open-rmf/rmf_demos`. Accessed: 2022-5-19.

[16] L. Rocha and K. Vivaldini. Plannie: A benchmark framework for autonomous robots path planning algorithms integrated to simulated and real environments. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 402–411. IEEE, 2022.

[17] G. Sharon, R. Stern, A. Felner, and N.R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artif. Intell.*, 219:40–66, February 2015.

[18] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T.K. Satish Kumar, E. Boyarski, and R. Bartak. Multi-Agent pathfinding: Definitions, variants, and benchmarks. June 2019.

[19] N.R. Sturtevant. MAPF benchmarks. `https://movingai.com/benchmarks/mapf.html`. Accessed: 2022-3-30.

[20] N.R. Sturtevant. Benchmarks for Grid-Based pathfinding. *IEEE Trans. Comput. Intell. AI Games*, 4(2):144–148, June 2012.

[21] M. Yagiura and T. Ibaraki. The generalized assignment problem and its generalizations. *St. Marys College of Maryland, St. Marys City, MD, USA, Tech. Rep.*, 1989.