2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Contextual Multi-Objective Path Planning

Anna Nickelson[1], Kagan Tumer[1], and William D. Smart[1]

Abstract— **Many critical robot environments, such as healthcare and security, require robots to account for context-dependent criteria when performing their functions (e.g., navigation). Such domains require decisions that balance multiple factors, making it difficult for robots to make contextually appropriate decisions. Multi-Objective Optimization (MOO) methods offer a potential solution by trading off between objectives; however concepts like Pareto fronts are not only expensive to compute but struggle with differentiating among solutions on the Pareto front. This work introduces the Contextual Multi-Objective Path Planning (CMOPP) algorithm, which enables the robot to trade off different complex costs dependent on context. The key insight of this work is to separate the path planning and path cost estimation into two independent steps, thus significantly reducing computation cost without impacting the quality of the resulting path. As a result, CMOPP is able to accurately model path costs, which provide meaningful trade-offs when choosing a path that best fits the context. We show the benefits of CMOPP on case studies that demonstrate its contextual path planning capabilities. CMOPP finds contextually appropriate paths by first reducing the search space up to 99.9% to a near-optimal set of paths. This reduction enables the generation of accurate path cost models, using up to 90% less computation than similar methods.**

I. Introduction

Robots have moved into critical sectors, such as healthcare and security, that require complex decisions that balance multiple context-dependent costs. A robot responding to an emergency call needs to choose a path that is likely to get there fastest with a low chance of failure. However, the same robot carrying delicate lab samples will instead need to consider paths with minimal potential disruptions, such as crowded hallways or bumps in the floor. Each of these contexts requires that the robot reason about different trade-offs to make an appropriate decision for the given context.

Multi-Objective Optimization (MOO) looks for optimal solutions on the Pareto front and offers ways to balance and trade off different objectives. However, finding the Pareto front is often computationally expensive, especially when complex costs are difficult to model. MOO methods typically calculate the cost of each solution during the search, effectively multiplying the complexity of each function. Surrogate methods aim to reduce computational complexity by using learned proxy functions to reduce the search space [1]. This allows them to perform the expensive cost computations on only a small subset of the search space. However, these require an extensive learning setup and training time.

*This work was funded by the Christina & Harold Merryman Fellowship, the National Institutes of Health (NIH) award, R01-EB024330, and the National Science Foundation grant No. IIS-2112633

[1]All authors are with the Collaborative Robotics & Intelligent Systems Institute, Oregon State University, Corvallis, OR 97331 {nickelsa, kagan.tumer, bill.smart}@oregonstate.edu

This work introduces the CMOPP algorithm, a path planner that leverages and balances complex cost measures to choose a contextually appropriate path. CMOPP accomplishes this by utilizing traditional graph search techniques to narrow the search space and focus on a promising set of paths. This allows it to separately perform the expensive path cost calculations, which are estimated from complex edge cost models, such as Gaussian Mixture Models (GMMs). This is a similar structure to surrogate multiobjective methods, but leverages graph search algorithms to eliminate the need for learned proxy functions. Finally, CMOPP employs MOO tools to trade-off between path costs and choose a final path based on contextual costs. In this work, we define context to be the set of objectives most appropriate to the situation, which is similar to setting dynamic weights – though that is not the mechanism used.

The key insight of this work is recognizing that paths are easy to find, but path costs can be expensive to compute. By separating the path and cost computations into two steps, we are able to significantly reduce overall computation without reducing the quality of paths found. As a consequence, CMOPP is able to accurately model path costs on a siginficantly reduced number of paths, which provide meaningful trade-offs when deciding which path best fits the context. Note that this approach does not guarantee the preservation of the optimal path for each objective. However, because additional computation will delay the determination of the optimal path, it is of little value in environments with uncertainty and non-deterministic execution. Instead, our approach focuses on the vicinity of the optimal solution for all costs concerned and provides paths that address all contextual requirements.

We show the utility of CMOPP on case studies that demonstrate the contextual path planning capabilities. CMOPP finds contextually appropriate paths by first reducing the search space up to 99.9% to a near-optimal set of paths. Though we do not guarantee coverage of the entire Pareto front, our method is able to find multiple solutions across the Pareto front in all cases tested. This reduction enables the generation of accurate path cost models using up to 90% less computation than other similar methods.

II. Related Works

Global path planning in a static environment with a single objective is considered solved. A* [2] and Dijkstra [3], including their many variants, provide an optimal path under these constraints. However, adding the complexity of a dynamic environment or multiple competing objectives leads to an abundance of open research questions. Our method

interweaves three areas of research: path planning that relies on (1) multiple objectives, (2) context, and (3) uncertainty.

A. Multi-Objective Path Planning

There are many strategies for autonomous decision making with multiple objectives. The strategies that are most closely related to this work are Pareto methods. Pareto methods aim to find the set of non-dominated paths. Non-dominated Sorting Genetic Algorithm II (NSGA-II) [4] is an evolutionary method commonly used for this purpose. NSGA-II ranks each point by how many points it dominates and how many points dominate it, then uses this to evolve the search space. Other methods find a Pareto front with more traditional path planning techniques [5], [6]. However, these methods do not include path costs that are difficult to compute, nor do they choose a path to adapt to the context.

Pareto methods can be computationally expensive to compute. Some works have looked at how to speed up the graph search component [7], [8]. Surrogate methods are utilized in cases where the environment is computationally cheap to search, but costs are expensive to compute [1], [9]. They rely on a set of learned proxy functions to reduce the search space. Once learned, these proxy functions are significantly cheaper to compute than the actual cost. This allows for the expensive cost computation to be performed on a small subset of the overall search space.

B. Contextual Path Planning

One interesting area of research that includes contextual information is Tourist Route Planning (TRP). Methods often incorporate the information into either a cost map or as a weight function [10], [11]. These methods integrate context by choosing and balancing a subset of costs to best fit the context. They rely on a linear combination of costs and sum across edges to estimate the total path cost. In order to make a final decision, some present the Pareto front as a set of options to a user [12], [13] or learn human preferences to align the decision-making process with human intent [14].

C. Path Planning Under Uncertainty

Global planning under uncertainty presents unique challenges, as the robot must learn and utilize long-term trends to plan a path. Many methods utilize non-scalar edge costs to model uncertainty, including normal distributions [15]. These methods rely on the ability to easily add values across edges to come up with a representative path cost. Therefore, they can rely on traditional graph search algorithms by incorporating these costs into the weight function.

Mixture models present a unique challenge, as they cannot be easily used in traditional graph search methods that add attributes across edges to get a path cost. Yang, et al. [16] model uncertain edge costs as a Gaussian Mixture Model (GMM), then find the Pareto front of paths. They use a graph search technique and estimate the path cost with every expanded edge by convoluting the individual distributions across edges. Using this method, if each distribution contains two modes, the result is 2^n distributions, where n is the number of edges. For example, combining the bi-modal distributions of six edges, resulting in 64 distributions. This quickly becomes an unwieldy and uninformative model. They handle this by sampling and refitting a new GMM as the number of distributions grows. Given that they are performing this method for each edge expanded in the graph search, this method is extremely computationally heavy and infeasible in large search spaces.

III. Contextual Multi-Objective Path Planning (CMOPP) Algorithm

In this work, we study how to alter a robot's path to best fit the operating context, where the context mandates trading off multiple costs that are difficult or expensive to compute. This paper introduces the Contextual Multi-Objective Path Planning (CMOPP) algorithm, a path planner that leverages complex uncertainty models to choose a contextually appropriate path. CMOPP models uncertainty in the environment to enable the generation of multiple meaningful path costs.

CMOPP accomplishes this in three steps, shown in Algorithm 1. First, it reduces the search space in order to focus on a set of promising paths (Alg. 2). This allows it to spend more computation power to estimate the path costs from complex edge cost models (Alg. 3). Finally, it chooses amongst the paths by balancing the costs most relevant to the given context (Alg. 4).

Variables used in these algorithms are as follows:

- Start and goal nodes ($\mathbf{start}$, $\mathbf{goal}$)
- Graph ($\mathbf{G}$) with observed edge costs ($\mathbf{EC}$)
- The set estimated path costs ($\mathbf{PC}$)
- The set of contextual costs ($\mathbf{CC}$), the subset of $\mathbf{PC}$ that is most relevant to the current context. Ordered with most important first
- The set of paths ($\mathbf{P}$) found in Algorithm 2
- The set of domination numbers ($\mathbf{D}$) for each path.
- An individual cost ($\mathbf{c}$), path ($\mathbf{p}$), weight function ($\mathbf{w}$), edge ($\mathbf{e}$), sample ($\mathbf{s}$), or domination number ($\mathbf{d_p}$).
- The final path ($\mathbf{final}$) chosen

Algorithm 1: Top-level algorithm

```
1 Function CMOPP(G, start, goal)
2     PC = {} // Initialize path costs
3     P ← Reduce(G, start, goal) // Alg. 2
4     PC ← Estimate(G, P) // Alg. 3
5     final ← Choose(CC, P, PC) // Alg. 4
6     return final;
```

A. Algorithm 2: Reduce the search space

CMOPP relies on complex edge costs to inform contextually appropriate decisions. This makes it computationally expensive to get an accurate path cost model from edge costs. CMOPP first reduces the search space in order to focus on a set of promising paths to compare. The key is to utilize a diverse set of edge costs to find a breadth of partially optimized paths. The aim here is not to form a Pareto front,

Algorithm 2: Reduce the search space

```
1  Function Reduce(G, start, goal)
2      P = {};                    // Resulting paths
3      foreach c_i, c_j ∈ G.EC × G.EC do
4          w = c_i × c_j;         // Weight function
5          p ← Dijkstra(G, start, goal, w);
6          if p ∉ P then
7              add p to P;
8      return P
```

but instead focus the search in order to provide meaningful trade-offs that may be appropriate in different contexts.

CMOPP creates a set of diverse weight functions by taking the product of each pair of possible edge costs (e.g. distance and mean time to traverse an edge). CMOPP runs Dijkstra with each weight function and keeps the unique paths, as is shown in Algorithm 2. Note that the exact functional form of these weight functions does not have a significant impact on the results; the critical factor is their ability to cover a breadth of different costs in order to find a diverse set of partially optimized paths.

B. Algorithm 3: Estimate timing distributions

By reducing the search space to a set of partially optimized paths, CMOPP focuses additional computation on creating accurate path cost models. Algorithm 3 demonstrates how CMOPP estimates the path costs. Many relevant costs, such as distance, can be accurately represented by a scalar. Other costs, however, require a more complex representation. The distribution of times it takes for the robot to traverse a sporadically busy hallway may have multiple peaks; as such, the time cannot be represented by a scalar or normal distribution, but can be modeled as a GMM. We consider two models for path costs: scalars and GMMs.

For this work, we model the time it takes to traverse each edge as a GMM. As demonstrated by Yang, et. al. [16], combining edge costs represented by GMM distributions requires convolution and/or sampling in order to find an accurate path cost model. Therefore, CMOPP estimates the path timing costs by sampling from each edge cost distribution; this method is inspired by Markov Chain Monte Carlo (MCMC) techniques [17]. As shown in lines 4 to 13 in Algorithm 3, for each iteration and each path, CMOPP takes a sample time from each edge GMM, then sums the sampled times together to get a realistic sample path time. After a fixed number of iterations, it fits a GMM to the sampled times for each path. Since the environment is stochastic, this step creates meaningful cost models for each path.

This differs from, and is more efficient than, the method developed by Yang, et. al [16]. They convolve, sample, and refit the distributions multiple times for a single path, which is done iteratively throughout the graph search each time an edge is expanded. Our method does not use convolution; we only sample and fit a new GMM once per path, as our path cost calculations are separate from the graph search.

For all scalar edge costs, we calculate the associated path cost by summing values across edges. For example, the distance of the path can be directly calculated by adding the distance of each individual edge. This is shown in lines 14 to 17 in Algorithm 3.

Algorithm 3: Estimate path costs

```
1  Function Estimate(G, P)
2      PC = {};
       // For each path and edge cost
3      foreach p ∈ P, c ∈ G.EC do
4          if c is GMM then
               // Take samples from edge GMM
                  and fit GMM for path
5              samples = [];
6              repeat 1000 times
7                  s = 0;
8                  foreach e ∈ p do
9                      s += sample from c_e;
10                 samples.append(s);
11             PC_{p,GMM} ← fit a GMM to samples
12         else // c is a scalar
13             cost = 0;
14             foreach e ∈ p do
15                 cost += c_e;
16             PC_{p,c} ← cost ;
17     return PC
```

C. Algorithm 4: Choose a path

Finally, CMOPP uses the path cost models to choose a path that best fits the context. CMOPP calculates the set of non-dominated paths of the reduced search space under the given subset of contextual costs, $\mathbf{CC}$. This method was inspired by the domination criterion in NSGA-II [4].

We define dominance under the contextual costs, $\mathbf{CC}$; path A dominates path B if changing from A to B will cause deterioration in the performance over one or more costs, with no improvement in other costs. For the given subset of costs, CMOPP calculates the domination number for each path by counting the number of paths it is dominated by and the number of paths it dominates, then subtracting the former from the latter. Utilizing both values allows CMOPP to find the Pareto front of the reduced space and contextual costs, then determine most dominant point within the front. If this step does not result in a single path, a runoff occurs. The runoff criteria is the first cost in the given contextual costs, $\mathbf{CC}$, which is ordered by importance. This is the only case where the order of costs comes into play; all other steps assume they are equally important.

Figure 1 shows the domination criterion and runoff situation. Note that the x and y values are purely demonstrative, as this works for any set of costs. Three of the four Pareto front points have the same domination number, as they all dominate the same number of points and are not dominated by any others. This is a case where the runoff criterion are

needed. The blue triangle is chosen, as it is the most optimal of the three points under the primary cost (x-axis).

Algorithm 4: Choose a path based on context

Function `Choose`$(\mathbf{CC}, \mathbf{P}, \mathbf{PC})$
  $\mathbf{D} = \{\}$;
  foreach $\mathbf{p_i} \in \mathbf{P}$ **do**
    $\mathbf{d_{p_i}} = 0$;
    foreach $\mathbf{p_j} \in \mathbf{P}, \mathbf{p_i} \neq \mathbf{p_j}$ **do**
      if $\mathbf{p_i}$ *dominates* $\mathbf{p_j}, \forall \mathbf{c} \in \mathbf{CC}$ **then**
        $\mathbf{d_{p_i}} \mathrel{+}= 1$;
      else if $\mathbf{p_j}$ *dominates* $\mathbf{p_i}, \forall \mathbf{c} \in \mathbf{CC}$ **then**
        $\mathbf{d_{p_i}} \mathrel{-}= 1$;
    add $\mathbf{d_{p_i}}$ *to* $\mathbf{D}$;
  final $\leftarrow \mathbf{p_k}$, *s.t.* $\mathbf{d_{p_k}} = max(\mathbf{D})$;
  return final

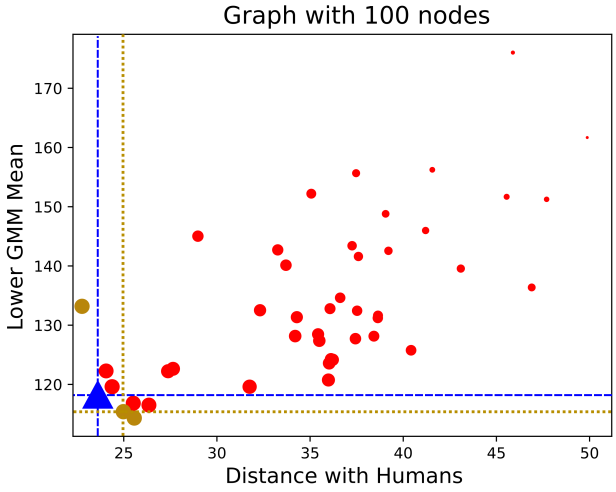

Fig. 1. Demonstration of Algorithm 4. Red points are dominated; gold are the Pareto front; the blue triangle is the final path choice. This method assigns a dominance value to each point (points are scaled by dominance value) by counting the number of points it dominates, and number of points it is dominated by, then subtracting the latter from the former.

IV. Experimental Evaluation

Below we outline the details of how we implemented CMOPP. The implementation utilized the data described below to demonstrate a useful application of CMOPP.

A. Experimental Setup

This section covers how we collected the data used in this paper. First, we model a building environment with humans present. The simulation environment is set up in Gazebo and an undirected NetworkX graph; each node is a region grounded in 2D space and edges are the areas between them, as shown in Figure 2. On each edge, we represent how often humans are present (frequency) and how "busy" the space is when there are humans present (presence). In the simulation

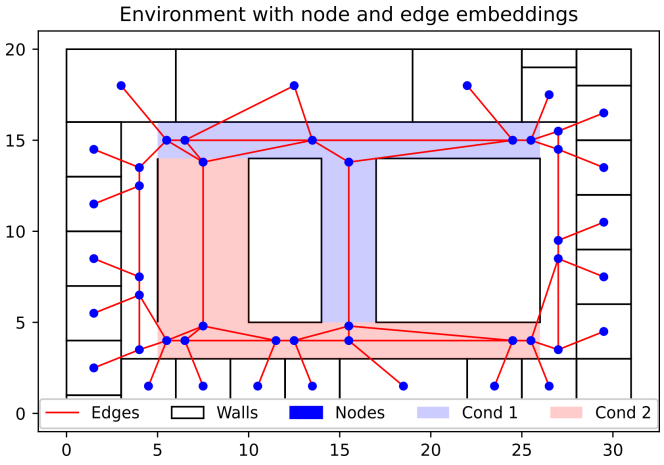

Fig. 2. Environment with node and edge embedding. Cond 1 and 2 show the human conditions in each area of the map.

environment, we reduced the robot's top speed as a proxy for presence. We used three conditions:

- Condition 0: humans never present
- Condition 1: frequency 40%; top speed reduced to 75%
- Condition 2: frequency 80%; top speed reduced to 50%

Areas where no presence is specified are in condition 0. In any given moment, the presence of humans along an edge is independent of all other edges.

We collected data of the robot continuously navigating around the space using the standard Robot Operating System (ROS) navigation stack. We recorded the amount of time it took to traverse each edge and fit a GMM to the data. We remove any timings where there were localization or navigation failures that result in a failure of all recovery behaviors (more than four standard deviations above the mean). Navigation failures are important to note, but we do not want these extreme outliers to affect statistical measures of timing, as it could provide an unrealistic picture. Therefore, in a separate measure, we tally of how many times navigation completely failed on a given edge.

Based on the data collected from the ROS test environment, we also created a set of NetworkX graphs to test the speed and accuracy of our method in different search spaces. We modeled the edge costs on those collected from ROS and created 20 worlds, ranging from 16 nodes to 150 nodes.

B. Contextual costs for Algorithm 4

This section outlines the contextual costs used. In this implementation, we use the robot's task as a type of context. The motivation behind these examples is enabling an assistive robot in a hospital setting to find the best course of action in different contexts.

Scenario 1: The urgent context is for when the robot needs to get somewhere with relative urgency. In this scenario, the robot will consider the following costs to meet the goal:

- Cost 1: 95% upper quartile cost of the largest path GMM, as these will be the slowest timings on that path

- Cost 2: Mean of all path time data, as it will need a relatively fast path

Scenario 2: The delicate context is for when the robot requires minimal disturbance, such as when it is carrying something delicate or that cannot be jostled. In this scenario, the robot will need to consider the least disruptive path; time and distance will be less of a concern. In order to do this, it minimizes the following costs:

- Cost 1: Total distance over which it encounters humans
- Cost 2: Number of intersections - this is used as a proxy for how many times it might have to stop along the path due to cross-traffic, a cost that would be gathered from real data.

Scenario 3: The after hours context is for times when there are few humans around, such as after visiting hours.

- Cost 1: 95% quartile of the smallest path time GMM, as these would likely be times when there are minimal humans around
- Cost 2: Number of navigation failures in order to avoid areas that may cause it to go into recovery behavior

V. Results

In this section, we show the efficiency and accuracy of Algorithms 2, 3, and 4 individually, then provide case studies to demonstrate the contextual decision making capabilities of CMOPP. To our knowledge, no other method has implemented contextual multi-objective path planning with GMMs as edge costs. Therefore, we provide a piecewise comparison to the other leading methods that cover relevant subsets of this work. We show our method provides similar and accurate results in significantly less time. We utilize the ROS and graph world data described in section IV-A.

A. Algorithm 2: Reduce the search space

We tested this method in the fabricated grid worlds to show the efficiency of search space reduction in different size graphs. This method reduces the search space by 88% for smaller graphs (15 nodes) and more than 99.9% in larger graphs (>25 nodes) compared against all simple paths.

Simply reducing the space is insufficient, as it does not guarantee a good set of paths. Figure 3 shows the quality of the resulting set of paths (both costs are minimized) against all possible paths, which is the entire search space. This graph shows that our method focuses the search on the area that balances between the costs. These results were consistent across all graph sizes and costs tested. Note that the subset of paths do not aim to form a Pareto front, but instead provide meaningful tradeoffs that may be appropriate in different contexts. In spite of having no theoretical coverage guarantees, in practice, this approach provides good coverage of the Pareto front because CMOPP utilizes a diverse set of cost functions to account for different potential tradeoffs. This shows that this method is extremely effective in not only reducing the search space but finding a near optimal set of paths from which to choose.

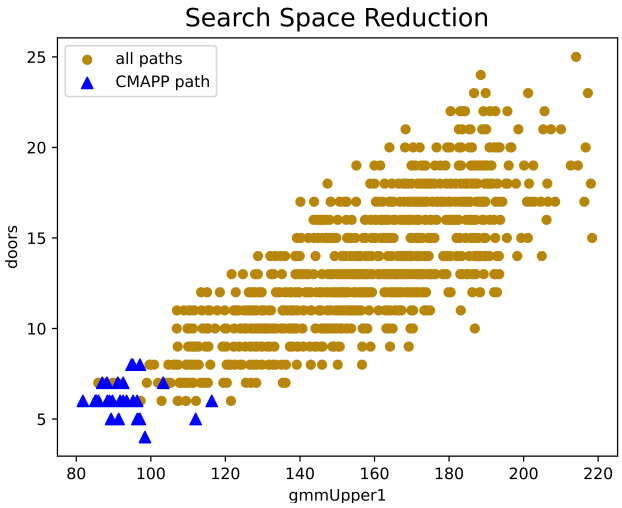

Fig. 3. This data came from the grid world with 20 nodes; our method reduced the search space from 976 to 29 paths. Blue triangles show paths chosen by our method. Yellow markers show the entire search space. Results were consistent across all graphs and costs tested.

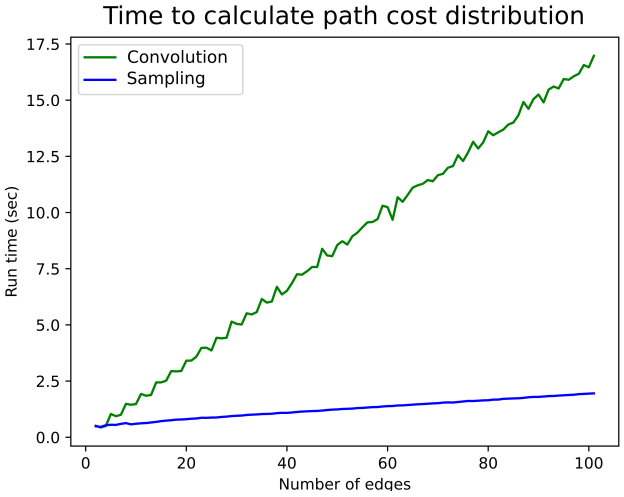

Fig. 4. This figure shows the time taken to calculate one path distribution versus the number of total edges in the graph. This compares our method (sampling) to the convolution method used by Yang, et. al. [16]. Run time is calculated as the average time over ten trials with randomly generated edge cost mixture models.

B. Algorithm 3: Estimate timing distributions

Next, we demonstrate the efficiency and accuracy of estimating the path costs in Algorithm 3. For simplicity, we refer to this method as the sampling method or Algorithm 3. In this instance the edge cost we are modeling is the time it takes to traverse each edge, recorded in ROS and modeled as a GMM. The sampling method estimates path cost distributions by sampling from the edge costs.

To demonstrate efficiency, we compare the sampling method against the convolution method presented by Yang, et al [16]; we show that our method reduces computation time by up to 90% compared to convoluting, resampling,

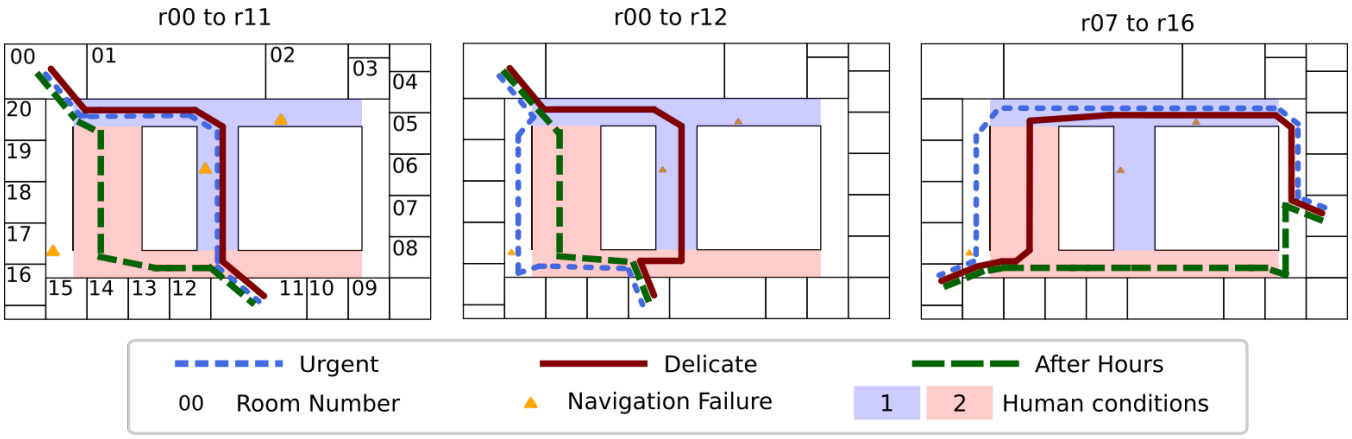

Fig. 5. Comparison of paths chosen for three different contexts. The human conditions are described in section IV-A. Where there is no condition specified, there are no humans. In condition 1, there are humans 40% of the time and the robot's top speed is reduced to 75%; condition 2 has humans around 80% of the time and the robot's speed is reduced to 50%. r## represents the room endpoints of the path (associated with the room numbers in the first figure).

and refitting. The sampling method also scales much better in larger search spaces, as can be seen in Figure 4.

However, reducing the computation time is not sufficient if it does not provide accurate results. We demonstrate the accuracy of this method by comparing against direct measurements of path costs in ROS, the ground truth. Table I shows the result of fitting GMMs to each of these distributions. These results show that our method provides accurate costs that can be used to make a path choice.

TABLE I: Comparison of GMMs fit to the distribution of data by the sampling method and ROS experiments.

	Mean	Stdev	Wt	Mean	Stdev	Wt
Alg. 3	197.5	12.2	0.20	239.1	12.9	0.80
ROS	198.2	11.0	0.18	240.3	10.9	0.82

C. Algorithm 4: Choose a path

The final step is to choose a path for the robot to execute. As demonstrated in Figure 1, CMOPP finds the most dominant points within the reduced search space, utilizing runoff criterion if there is a tie. This allows the robot to make a final decision by balancing multiple competing costs.

With this last step, we present case studies in Figure 5 showcasing CMOPP in its entirety. This figure shows the capabilities of CMOPP, based on the urgent, delicate, and after hours contexts. Note that condition 1 has fewer people around less often (40% of the time); condition 2 is crowded often (80% of the time). Any areas not under these conditions do not have humans present at any point in time. Areas where navigation failures occurred are noted with a yellow triangle; this cost is used in the after hours context.

We show CMOPP planning a path between two rooms on opposite sides of the building. Rooms that are close together do not provide interesting cases, as there is often only one reasonable path. Each sub-figure shows CMOPP planning a path for each of the three contexts.

The urgent context is used when the robot needs to get somewhere with relative urgency. From Figure 5, we see that in the urgent context, CMOPP occasionally takes a slightly longer route but finds paths that avoid heavy foot traffic and slow areas. The delicate scenario is used when the robot needs minimal physical disruption, such as when it is handling something delicate like a lab sample. In the delicate scenario, CMOPP avoids crowded hallways or paths with many potential interruptions (using intersections as a proxy). Finally, the after hours context is for times when there are likely few humans around. It is able to ignore most traffic, as it is operating during non-peak hours. CMOPP finds quick paths that also avoid areas the robot may get stuck (represented by navigation failures).

VI. Conclusion

This work introduces the Contextual Multi-Objective Path Planning (CMOPP) algorithm, a path planner that leverages complex cost measures to choose a contextually appropriate path. By first reducing the search space, we focus the expensive path cost estimations on a partially optimal set of paths. As a result, CMOPP accurately models path costs, which provides meaningful trade-offs when deciding which path best fits the context.

We demonstrate the contextual reasoning provided by this method through case studies. CMOPP finds a contextually appropriate path by effectively reducing the search space up to 99.9% to a near-optimal set of paths. CMOPP also provides accurate path costs using up to 90% less computation than other similar methods. For this work, the contextual objectives are hand-selected and tuned to individual scenarios. Future work would seek to learn which objectives are important for these situations and expand to novel contexts that have not been anticipated.

References

[1] Q. Liu, X. Li, H. Liu, and Z. Guo, "Multi-objective metaheuristics for discrete optimization problems: A review of the state-of-

the-art," *Applied Soft Computing*, vol. 93, p. 106382, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494620303227

[2] 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*, vol. 4, no. 2, pp. 100–107, 1968.

[3] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[5] F. Ahmed and K. Deb, "Multi-objective path planning using spline representation," in *2011 IEEE International Conference on Robotics and Biomimetics*, Dec 2011, pp. 1047–1052.

[6] Z. Ren, S. Rathinam, M. Likhachev, and H. Choset, "Multi-objective safe-interval path planning with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8154–8161, July 2022.

[7] C. Hernández Ulloa, W. Yeoh, J. A. Baier, H. Zhang, L. Suazo, and S. Koenig, "A simple and fast bi-objective search algorithm," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, no. 1, pp. 143–151, Jun. 2020. [Online]. Available: https://ojs.aaai.org/index.php/ICAPS/article/view/6655

[8] Z. Ren, R. Zhan, S. Rathinam, M. Likhachev, and H. Choset, "Enhanced multi-objective a* using balanced binary search trees," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 15, no. 1, 2022, pp. 162–170.

[9] T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen, "A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms," *Soft Computing*, vol. 23, no. 9, pp. 3137–3166, 2019.

[10] T. Hirakawa, T. Yamashita, and H. Fujiyoshi, "Scene context-aware rapidly-exploring random trees for global path planning," in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2019, pp. 608–613.

[11] L. Piyathilaka and S. Kodagoda, "Affordance-map : A map for context-aware path planning," in *Australasian Conference on Robotics and Automation 2014*, Melbourne, Australia, 12 2014. [Online]. Available: https://www.araa.asn.au/acra/acra2014/papers/pap148.pdf

[12] H.-P. Kriegel, M. Renz, and M. Schubert, "Route skyline queries: A multi-preference path planning approach," in *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. IEEE, 2010, pp. 261–272.

[13] A. M. Hendawi, A. Rustum, A. A. Ahmadain, D. Hazel, A. Teredesai, D. Oliver, M. Ali, and J. A. Stankovic, "Smart personalized routing for smart cities," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, 2017, pp. 1295–1306.

[14] M. T. Shaikh and M. A. Goodrich, "A measure to match robot plans to human intent: A case study in multi-objective human-robot path-planning*," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2020, pp. 1033–1040. [Online]. Available: https://faculty.cs.byu.edu/~mike/mikeg/papers/ShaikhGoodrich2020.pdf

[15] J. J. Chung, A. J. Smith, R. Skeele, and G. A. Hollinger, "Risk-aware graph search with dynamic edge cost discovery," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 182–195, 2019.

[16] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang, "Stochastic skyline route planning under time-varying uncertainty," in *2014 IEEE International Conference on Data Engineering*, 2014, pp. 136–147.

[17] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov Chain Monte Carlo in practice*. London: Chapman & Hall, 1996.