

Optimal Constrained Task Planning as Mixed Integer Programming

Alphonsus Adu-Bredu¹ Nikhil Devraj¹ Odest Chadwicke Jenkins¹

Abstract—For robots to successfully execute tasks assigned to them, they must be capable of planning the right sequence of actions. These actions must be both optimal with respect to a specified objective and satisfy whatever constraints exist in their world. We propose an approach for robot task planning that is capable of planning the optimal sequence of grounded actions to accomplish a task given a specific objective function while satisfying all specified numerical constraints. Our approach accomplishes this by encoding the entire task planning problem as a single mixed integer convex program, which it then solves using an off-the-shelf Mixed Integer Programming solver. We evaluate our approach on several mobile manipulation tasks in both simulation and on a physical humanoid robot. Our approach is able to consistently produce optimal plans while accounting for all specified numerical constraints in the mobile manipulation tasks. Open-source implementations of the components of our approach as well as videos of robots executing planned grounded actions in both simulation and the physical world can be found at this url: <https://adubredu.github.io/gtmip>

I. INTRODUCTION

The successful execution of manual tasks often requires the satisfaction of certain physical constraints. For instance, to retrieve a sugar canister from a seven-foot high shelf in a kitchen, the average person would have to stand *close enough* and stretch their hands *far enough* to not only reach the sugar, but to grasp it stably and lift it. Here, *close enough* and *far enough* are physical constraints that need to be satisfied to guarantee the success of their efforts to retrieve the sugar. Similarly, for a robot to successfully perform this sugar retrieval task, it would have to account for similar physical constraints when *deciding* the *right actions* to take. While deciding the right actions, this shrewd robot would also have to bias its decisions towards actions that optimize certain quantities like energy consumed or distance travelled. We call this problem the *Optimal Constrained Task Planning* problem.

The predominant way to solve a task planning problem is to formulate it as a symbolic AI planning problem, represent it in a graph structure and employ graph

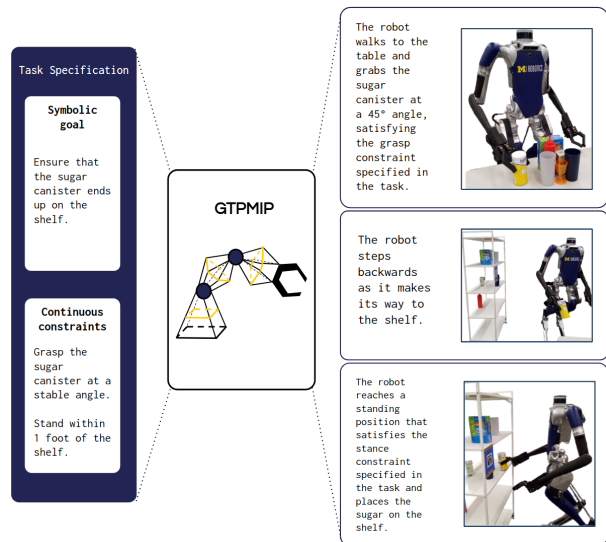


Fig. 1: Given a constrained task planning problem, our approach (GTPMIP) plans a sequence of coherent actions with optimal parameters needed to accomplish the task.

search algorithms to find paths from a start state to a goal state. However, this approach is purely symbolic and provides no avenues for incorporating numerical constraints in the planning process or to bias the search to choose actions that optimize numerical objective functions [4], [5], [12]. As such, these approaches only allow the specification of symbolic task goals and are unable to support the specification of continuous goals. It is often up to the human expert to introduce symbols that aptly represent desired continuous goals. A few works have proposed extensions to graph search algorithms to enable them to handle constraints and objective functions [7]. However, these approaches are only capable of handling simple additive objective functions with soft linear constraints.

In this work, we propose an approach for task planning that is capable of handling both linear and non-linear constraints and optimizes for convex objective functions to global optimality. We take a unique approach to the task planning problem by encoding the entire task planning problem as a single Mixed Integer Convex Program (MICP). By doing this, we gain the flexibility of subjecting the problem to arbitrary action

¹Alphonsus Adu-Bredu, Nikhil Devraj and Odest Chadwicke Jenkins are with the Robotics Institute and Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA. [adubredu|devrajn|ocj]@umich.edu

constraints that need to be satisfied in order for the resulting plan to be physically realizable by a robot. We also escape the restriction of having to specify planning goals symbolically as this encoding enables the specification of continuous planning goals. We then use an off-the-shelf Mixed Integer Programming solver to solve the MICP to optimality, extract the grounded plan and its optimal parameters and execute the plan with a robot. The unique contributions of this work are as follows:

- Firstly, we extend the Planning Domain Definition Language (PDDL) to allow for the specification of numerical action and task constraints, numerical initial values of continuous task variables, numerical objective functions, numerical action dynamics functions, numerical preconditions and numerical effects. We call this extension the Hybrid PDDL Description (HPD).
- Secondly, we propose a unique representation of continuous actions as Funnels and propose an approach for representing the continuous plan space, which we call the Hybrid Funnel Graph.
- Finally, we describe an approach for encoding the Hybrid Funnel Graph as a single Mixed Integer Convex Program which we solve using an off-the-shelf MIP solver.

We evaluate our approach in simulation on several 2-D object rearrangement task planning problems subject to unique geometric constraints. We also demonstrate our approach on real-world mobile manipulation tasks involving kinematic constraints using the Digit humanoid robot, as shown in Figure 1.

II. RELATED WORKS

A. Mixed Integer Programming

A Mixed Integer Program (MIP) is a mathematical optimization problem with both integer and real-valued variables [18]. The ability of MIPs to have both discrete and continuous variables makes them ideal for formulating sequential planning problems that involve taking discrete actions which are subject to continuous constraints [1], [6], [15], [24]. This work encodes the optimal constrained task planning problem as a Mixed Integer Convex Program (MICP).

B. Symbolic AI Planning

The state-of-the-art methods in STRIPS-style [8] Symbolic AI Planning first decompose the planning problem into a causal graph and employ graph search techniques like A* [11] to find plans from some initial node in the graph to a desired goal node [3], [12], [13]. Although these approaches thrive for purely symbolic domains, they do not naturally allow for the incorporation of numerical constraints and objective functions

[16]. The formulation of AI Planning problems as Integer Programs has been explored a few times in the planning and scheduling literature [25], [26].

In this work, we build up on Vossen et. al's [26] Integer Programming formulation by encoding the AI Planning problem as a MIP and solving it using off-the-shelf MIP solvers. This MIP encoding is convenient because it naturally allows for the incorporation of numerical constraints and objective functions into the planning problem. This ability is essential because real world problems often involve numerical constraints and objectives. Metric-FF [14] is a symbolic planning system that can account for numerical state variables. However, unlike our approach, Metric-FF is incapable of optimizing for convex objective functions, continuous action dynamics and continuous task constraints.

C. Integrated Task and Motion Planning

The class of approaches that interleave symbolic AI planning and continuous planning is called Task and Motion Planning [2], [9], [22], [23]. Among these, approaches like Garrett et. al. [9] and Srivastava et. al. [22] devise symbols to describe continuous constraints for actions. These symbols are used as action preconditions in the symbolic AI planning process and are evaluated on demand. In addition to the chore of having to devise symbolic abstractions for every continuous constraint, these approaches are hampered by their requirement of symbolic goal descriptions. They are incapable of planning for continuous goal descriptions that can only be evaluated by an objective function. By formulating the planning problem as a Mixed Integer Convex Program, our proposed approach is able to both reason on the symbolic level using integer-valued variables and integer inequalities and optimize for continuous objective functions using the real-valued variables, and convex equations and inequalities.

D. Combined symbolic and continuous planning as Mathematical Programs

Works like Toussaint [23] and Li and Williams [19] have sought to solve the combined symbolic and continuous planning problem by formulating them as Mathematical Programs. Toussaint [23] uses an iterative 3-level Nonlinear Constrained Optimization to optimize for continuous robot configurations over discrete action sequences it acquires from running Monte Carlo Tree Search. Our approach differs from Toussaint [23] in that, we formulate the entire planning problem as a single Mixed Integer Program, solving for both the discrete action sequences and the continuous robot configurations in a single run. Li and Williams [19] employ hybrid flow graphs to represent the entire plan space and formulate the planning problem as a Mixed Logic

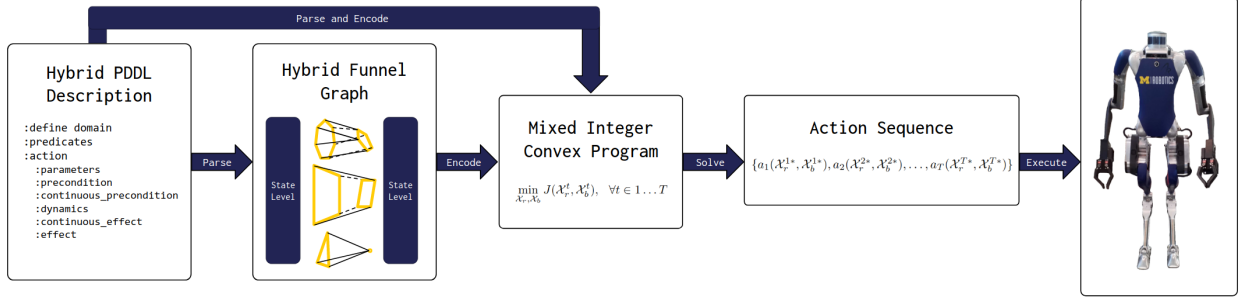


Fig. 2: An overview of GTPMIP. Given a task, our approach represents the plan space as a Hybrid Funnel Graph, encodes the task and the Hybrid Funnel Graph as a Mixed Integer Convex Program and solves it to produce an optimal action sequence which is executed by the robot.

Nonlinear Program in planning actions for autonomous underwater vehicles in ocean exploration tasks. Our representation of continuous actions as funnels and our formulation of Hybrid Funnel Graphs to represent the plan space are inspired by Li and Williams’s work.

III. PROBLEM FORMULATION

In this work, we tackle the problem of optimal task planning under numerical constraints. Our goal is to generate a grounded plan made up of a logically consistent sequence of actions, with each action associated with its corresponding optimal continuous parameters. We will use the task of package rearrangement within a warehouse environment (The Warehouseman’s Problem [21]) by a mobile manipulator robot as a running example for the remainder of this paper.

The inputs to our approach are:

- A set of initial symbolic propositions, \mathcal{I} , that describe the initial symbolic state of the world. For example, the set

$$\mathcal{I} = \{(\text{hand-empty}), (\text{not } (\text{packed boxA}))\}$$

represents a world where the robot is not holding any package and `boxA` has not been packed.

- A set of initial continuous variable values, \mathcal{X}_r^I and \mathcal{X}_b^I , where \mathcal{X}_r^I represents the robot’s initial configuration in SE(2) space and \mathcal{X}_b^I represents the configurations of the packages, also in SE(2) space.
- A set of action primitives, \mathcal{A} , that can be executed by a robot. An action primitive is comprised of:
 - Symbolic preconditions: A conjunction of symbols whose truth-value must be true in order for the action to be executable.
 - Continuous preconditions: Continuous constraints on the continuous variables (\mathcal{X}_r and \mathcal{X}_b) that must be satisfied in order for the action to be executable.
 - Action Dynamics: A dynamics function that computes the state of the continuous variables (\mathcal{X}_r and \mathcal{X}_b) after the action is executed.

- Symbolic effects: A conjunction of symbols that represent the state of the world after the action is executed.
- Continuous effects: The numerical values of continuous variables (\mathcal{X}_r and \mathcal{X}_b) after the action is executed.

- A set of task-specific numerical constraints \mathcal{H} on the continuous variables.
- A set of goal symbolic propositions, \mathcal{G} , that must hold true at the end of the plan execution. For example, the symbolic propositions

$$\mathcal{G} = \{(\text{hand-empty}), (\text{packed boxA})\}$$

would represent a world where the robot is not holding any package and `boxA` is packed.

- A set of goal continuous variable values, \mathcal{X}_r^G and \mathcal{X}_b^G .
- An objective function $J(\mathcal{X}_r, \mathcal{X}_b)$ to be optimized.

The output of our approach is a grounded plan π^* made up of a sequence of logically consistent actions $\{a_1(\mathcal{X}_r^{1*}, \mathcal{X}_b^{1*}), a_2(\mathcal{X}_r^{2*}, \mathcal{X}_b^{2*}), \dots, a_N(\mathcal{X}_r^{N*}, \mathcal{X}_b^{N*})\}$, with each action associated with its corresponding optimal continuous parameter values.

IV. METHODOLOGY

In this section, we describe each component of our approach, as illustrated in Figure 2. We name our approach *Grounded Task Planning as Mixed Integer Programming* (GTPMIP). As stated in the previous section, GTPMIP takes as input a description of the optimal constrained task planning problem including descriptions of action primitives the robot is capable of executing. GTPMIP then builds a Hybrid Funnel Graph from this description to represent the entire plan space. Finally, it encodes the Hybrid Funnel Graph and the planning problem as an MICP, which it then solves using an off-the-shelf MIP solver. Each of these components are described in the following subsections.

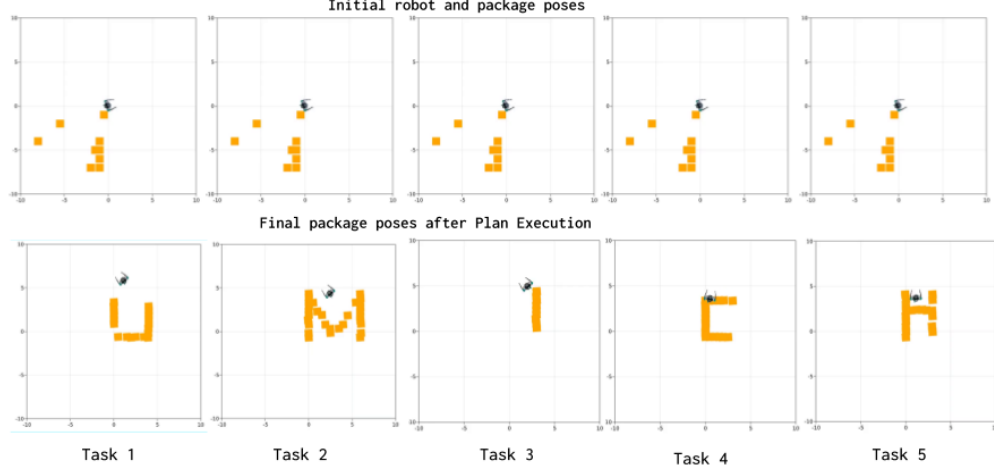


Fig. 3: Qualitative results from the execution of plans generated by solving the package rearrangement problems in Tasks 1-5 with GTPMIP

A. Hybrid PDDL Description

Hybrid PDDL Description (HPD) is an extension of PDDL that allows for the specification of task planning problems with numerical action and task constraints, numerical initial values of continuous task variables, numerical objective functions, numerical action dynamics functions, numerical preconditions and numerical effects. Similar to PDDL, an HPD description of a task planning problem is made up of two files; the `domain.hpd` file and the `problem.hpd` file.

The `domain.hpd` file describes the action primitives that the robot can execute. An action primitive has fields

- `:action` to specify the name of the action primitive
- `:parameters` to specify the symbolic and continuous parameters the action takes.
- `:precondition` to specify a conjunction of symbols whose truth-values must be true in order for the action to be executable.
- `:continuous_precondition` to specify continuous constraints on the continuous variables that must be satisfied in order for the action to be executable.
- `:dynamics` to specify dynamics functions that compute the state of the continuous variables after the action is executed.
- `:continuous_effect` to specify the numerical values of continuous variables after the action is executed.
- `:effect` to specify a conjunction of symbols that represent the state of the world after the action is executed.

The `problem.hpd` file describes the initial symbolic and continuous states as well as the goal symbolic and continuous states of the task. It also describes the task-specific constraints and the objective function to be

optimized.

PDDL+, which is also an extension to PDDL, allows for the specification of numerical action effects. However, unlike HPD, PDDL+ is incapable of specifying numerical task constraints, objective functions and action dynamics functions.

B. Funnels and Hybrid Funnel Graphs

We represent action primitives as *funnels*. A funnel is made up of three components; an input region, a dynamics function and an output region. The input region is the region of intersection of all the continuous constraints that need to be satisfied before the action can be executed (the continuous preconditions). The dynamics function computes the state of the continuous variables after the action is executed. We apply the dynamics function on the peripheries of the input region to result in a new region which we call the output region. The geometric representation of this abstraction takes the shape of a funnel as shown in Figure 4; hence its name. The representation of action primitives as funnels helps in determining which action primitives are *applicable* given the state of the robot. If the values of the continuous variables of the current state intersects with the input region of a funnel and the symbolic preconditions of the corresponding action hold true for the symbolic propositions of current state, then the action is *applicable*. The output region of the funnel also determines the continuous state after the corresponding action is executed. In addition to action funnels, *No-op* funnels are identity operations which represent actions that make no changes to the symbolic state of the world and whose set of symbolic preconditions are equal to their set of symbolic effects.

Given this representation of actions as funnels, we build up the Hybrid Funnel Graph by alternating be-

tween state levels and action levels. A state level is a set of all possible states (both symbolic and continuous) at a specific time instance. An action level is a set of all *applicable* funnels at a specific time instance. The first state level is a set of all the symbolic propositions \mathcal{I} and continuous variables \mathcal{X}_b^I where \mathcal{X}_r^I that make up the initial state. The continuous variables could take the form of either singular continuous values or intervals of continuous values that represent regions in the continuous space (SE(2) space in our package rearrangement problem formulation). We then compute all funnels that are applicable given the symbolic and continuous state variables in the first state level. These funnels constitute the first action level. As noted in the previous paragraph, a funnel is applicable to a state level if the continuous state variables in the state level intersect with the input region of the funnel and the symbolic preconditions of the action corresponding to the funnel hold true for the symbolic propositions of the state level. We also include to the first action level, *No-op* funnels for each symbolic proposition in the state level. The second state level is then computed as the set of all symbolic effects of actions and output regions of their corresponding funnels in the first action level. These output regions are computed by applying the funnel's dynamics function to the region of intersection of the continuous variables of the first state level and the funnel's input region. Likewise, the second action level is computed in the same manner as the first action level.

After the computation of each state level, we check if the goal symbolic propositions \mathcal{G} hold true in the state level and if the goal continuous variable values \mathcal{X}_b^G where \mathcal{X}_r^G intersect the continuous variable regions in the state level. If both of these conditions are true, we have a valid Hybrid Funnel Graph for the task and terminate the graph building process. If not, we keep building the graph by adding additional state and action levels. Each action level represents a single time step in the resulting plan. Hence the total number of action levels, T , represents the total period of the entire resulting plan. Note that, since the Hybrid Funnel Graph starts with the initial state level and ends with the terminal state level, the total number of state levels is greater than the total number of action levels, T , by 1. This process is similar to the process of building planning graphs in GraphPlan [5] except that planning graphs in GraphPlan are made up of only symbolic propositions and symbolic actions. Hybrid Funnel Graphs are made up of both symbolic propositions and continuous variables, hence its name.

Unlike with GraphPlan where the graph building process is guaranteed to terminate if the planning problem is valid, our approach to building Hybrid Funnel Graphs is not guaranteed to terminate due to our inclusion of continuous variables. However in this work we observe

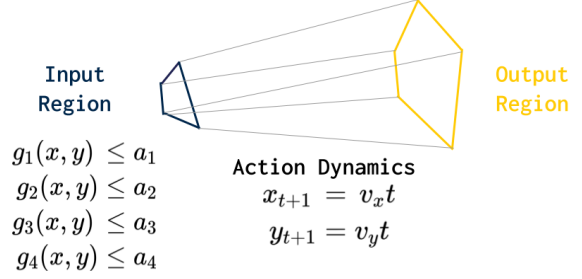


Fig. 4: A funnel representation for the move action. The input region is formed by the intersection of continuous precondition inequality constraints (g_1 to g_4) on the robot position. The action dynamics compute the next robot position, x_{t+1}, y_{t+1} after the action is applied to the poses in the input region. The output region represents the space of resulting poses.

GTPMIP successfully terminate graph building in every planning problem it is applied to.

C. Encoding as a Mixed Integer Convex Program

Before we encode the Hybrid Funnel Graph and the planning problem as an MICP, we first define a set of useful variables.

- Let T represent the total number of action levels in our Hybrid Funnel Graph, which is also the total planning period.
- Let \mathcal{F} represent the set of all instantiated symbolic propositions in our planning domain. Hence $\mathcal{I} \subseteq \mathcal{F}$ and $\mathcal{G} \subseteq \mathcal{F}$
- Let \mathcal{A} represent the set of all instantiated actions in our planning domain.
- Let pre_f represent the set of all actions that have symbolic proposition f as a symbolic precondition.
- Let add_f represent the set of all actions whose symbolic effects affirm symbolic proposition f . (the truth-value of f in the action's symbolic effect is **True**).
- Let del_f represent the set of all actions whose symbolic effects negate symbolic proposition f . (the truth-value of f in the action's symbolic effect is **False**)

Next, we define integer variables. For all $f \in \mathcal{F}$ and $t \in 1 \dots T$,

$$p_{f,t} = \begin{cases} 1, & \text{if proposition } f \text{ holds true at time } t \\ 0, & \text{otherwise} \end{cases}$$

$$q_{a,t} = \begin{cases} 1, & \text{if action } a \text{ is taken at time } t \\ 0, & \text{otherwise} \end{cases}$$

Given these variable definitions, we now build the constraints into our MICP.

The first set of constraints to be added are the initial and terminal constraints.

The initial constraint,

$$p_{f,1} = \begin{cases} 1, & \forall f \in \mathcal{I} \\ 0, & \forall f \notin \mathcal{I} \end{cases} \quad (1)$$

ensures that all initial symbolic propositions hold true in the first state level.

The terminal constraint,

$$p_{f,T+1} = 1, \quad \forall f \in \mathcal{G} \quad (2)$$

ensures that all goal symbolic propositions hold true in the last state level.

The next set of constraints are the precondition constraints

$$q_{a,t} \leq p_{f,t}, \quad \forall a \in \text{pre}_f, \forall t \in 1 \dots T, f \in \mathcal{F} \quad (3)$$

These inequality constraints encode the implication constraint that if action a , which has symbolic proposition f as its precondition, is taken in action level t , then f should also hold true in state level t . This constraint is called the precondition constraint because it ensures that all preconditions of an action hold true before the action can be taken.

The next set of constraints are the effect constraints

$$p_{f,t+1} \leq \sum_{a \in \text{add}_f} q_{a,t}, \quad \forall t \in 1 \dots T, f \in \mathcal{F} \quad (4)$$

These inequality constraints encode the implication constraint that if symbolic proposition f holds true in state level $t+1$, then at least one action a which has f as a positive effect should be taken in action level t .

The next set of constraints are the mutual exclusion constraints

$$q_{a,t} + q_{a',t} \leq 1 \quad (5)$$

for all $t \in 1 \dots T$ and all a, a' for which there exists an $f \in \mathcal{F}$ such that $a \in \text{del}_f$ and $a' \in \text{pre}_f \cup \text{add}_f$.

These inequality constraints ensure that two actions a and a' that cancel each other are not both taken in action level t .

The next set of constraints are the task-specific numerical constraints

$$q_{a,t} \leq h(\mathcal{X}_r^t, \mathcal{X}_b^t), \quad \forall h \in \mathcal{H}, t \in 1 \dots T \quad (6)$$

that ensure that if action a is taken in action level t , the continuous variable parameters of a satisfy all the task-specific numerical constraints \mathcal{H} .

The final set of constraints are the initial and terminal constraints

$$\mathcal{X}_r^1 = \mathcal{X}_r^I, \quad \mathcal{X}_b^1 = \mathcal{X}_b^I \quad (7)$$

and

$$\mathcal{X}_r^{T+1} = \mathcal{X}_r^G, \quad \mathcal{X}_b^{T+1} = \mathcal{X}_b^G \quad (8)$$

that ensure that the values of continuous variables at the first and final levels are equal to the problem-specified initial and goal continuous variable values respectively.

The objective function to be optimized

$$J(\mathcal{X}_r^t, \mathcal{X}_b^t) \quad \forall t \in 1 \dots T \quad (9)$$

is a convex function on all continuous variables for the entire planning period. For a warehouseman's problem, a suitable objective function would be to minimize the total Euclidean distance the robot travels while rearranging the packages.

Putting together Equations 1 - 9, our entire MICP can now be summarized as

$$\begin{aligned} & \min_{\mathcal{X}_r^t, \mathcal{X}_b^t, q_{a,t}} J(\mathcal{X}_r^t, \mathcal{X}_b^t), \quad \forall t \in 1 \dots T \\ & \text{subject to} \\ & p_{f,1} = \begin{cases} 1, & \forall f \in \mathcal{I} \\ 0, & \forall f \notin \mathcal{I} \end{cases} \\ & p_{f,T+1} = 1, \quad \forall f \in \mathcal{G} \\ & q_{a,t} \leq p_{f,t}, \quad \forall a \in \text{pre}_f, \forall t \in 1 \dots T, f \in \mathcal{F} \\ & p_{f,t+1} \leq \sum_{a \in \text{add}_f} q_{a,t}, \quad \forall t \in 1 \dots T, f \in \mathcal{F} \\ & q_{a,t} + q_{a',t} \leq 1 \\ & q_{a,t} \leq h(\mathcal{X}_r^t, \mathcal{X}_b^t), \quad \forall h \in \mathcal{H}, t \in 1 \dots T \\ & \mathcal{X}_r^1 = \mathcal{X}_r^I, \quad \mathcal{X}_b^1 = \mathcal{X}_b^I \\ & \mathcal{X}_r^{T+1} = \mathcal{X}_r^G, \quad \mathcal{X}_b^{T+1} = \mathcal{X}_b^G \\ & p \in \{0, 1\}, q \in \{0, 1\}, \mathcal{X} \in \text{SE}(2) \end{aligned} \quad (10)$$

We solve this MICP using an off-the-shelf MIP solver which returns the grounded plan π^* made up of a sequence of logically consistent actions $\{a_1(\mathcal{X}_r^{1*}, \mathcal{X}_b^{1*}), a_2(\mathcal{X}_r^{2*}, \mathcal{X}_b^{2*}), \dots, a_T(\mathcal{X}_r^{T*}, \mathcal{X}_b^{T*})\}$, with each action associated with its corresponding optimal continuous parameter values.

V. IMPLEMENTATION

A. Open-source software implementations

We provide open-source software implementations for each of the components of our approach. We provide

- **HPD.jl** as a software package for reading and parsing HPD files. It also contains example HPD files for the warehouse rearrangement problem. url: <https://github.com/adubredu/HPD.jl>
- **HybridFunnelGraphs.jl** as a software package for building complete Hybrid Funnel Graphs when given HPD files of an optimal constrained Task Planning Problem. url: <https://github.com/adubredu/HybridFunnelGraphs.jl>
- **gtpmip.jl** as a software package for encoding Hybrid Funnel Graphs and HPD files of an optimal constrained task planning problem as an MIP and solving the MIP to output the optimal plan. url: <https://github.com/adubredu/gtpmip.jl>

- `westbrick.jl` as a simulator for a 2D version of the Warehouse rearrangement problem. url: <https://github.com/adubredu/westbrick.jl>

B. Solving the Mixed Integer Convex Program

Throughout our experiments, we use the Gurobi Optimization software [10] to solve all MICPs. We use Gurobi because it was the fastest MIP solver amongst all solvers considered.

VI. EXPERIMENTS

We evaluate the capabilities of GTPMIP on a series of experiments both in simulation and on a physical robot.

A. Pure Symbolic Task Planning evaluation

First, we compare the symbolic task planning capabilities of GTPMIP to the state-of-the-art symbolic planning approaches Fast-Downward [12], Pyperplan [4] and Forward Search with A* [17]. We compare the planning times of these approaches on purely symbolic block stacking tasks with increasing problem size, with Problem 1 having 4 blocks and Problem 5 having 9 blocks. Table I shows the average planning times of each approach.

As can be seen from results in Table I, GTPMIP is slightly faster than Fast Downward and Pyperplan on the smaller Problems 1 - 4. GTPMIP however gets much slower than the other symbolic planning algorithms as the size of the problem increases in Problem 5. This significant reduction in planning speed can be attributed to the increase in number of variables and constraints in the resulting MIP that GTPMIP solves. However, the unique capabilities of GTPMIP that are lacking in the other symbolic planning approaches are its ability to account for numerical constraints and optimize for numerical objective functions. This is demonstrated in the next experiment.

B. Warehouse package rearrangement Problem

Next, we evaluate GTPMIP on a series of 5 tasks to evaluate its ability to perform optimal task planning under numerical constraints. Each task is setup with a virtual robot in a 2D warehouse simulator. For each Warehouse package rearrangement problem, the goal is to plan for the optimal action sequence with optimal continuous parameters that rearrange the packages by satisfying a specific set of linear geometric constraints on package placements in SE(2) space. The set of constraints for the five tasks are listed in Table II.

We evaluate the time to build the Hybrid Funnel Graph as well as the time to solve the resulting MICP for each of these tasks. Quantitative experimental results for each task are presented in Table III, with the corresponding qualitative results shown in Figure 3.

The experiments were run in `westbrick.jl`, a 2D package rearrangement simulator we developed.

Videos of the robot executing the plans generated for each task can be found on the project’s webpage at this url: <https://adubredu.github.io/gtpmip>

C. Mobile Manipulation tasks with Physical Robot

Finally, we employ GTPMIP in planning for optimal constrained tasks in the real world. We use the Digit [20] humanoid robot platform to execute output plans. We focus on 2 main tasks; the shelf-stocking task (pictured in Figure 1) and the table serving task.

The shelf-stocking task requires that the robot stock a shelf with a predefined set of grocery items at specific positions on the shelf. The table serving task requires that the robot collect a specified set of grocery items from a shelf and distributes them to a dinner table in predefined desired configurations.

The kinematic constraints we consider in these tasks are the robot stance pose constraint and the grasp angle constraint. The robot stance pose constraint constrains the robot’s standing pose to a desired region in SE(2) space from which the object to be grasped is kinematically reachable by the robot. The grasp angle constraint ensures that the angle of approach of the robot’s grippers results in a stable grasp.

Video demonstrations of the robot performing all these tasks can be found on the project’s website at this url: <https://adubredu.github.io/gtpmip>

VII. DISCUSSION

The primary assumption we made in this work is that the entire environment was fully-observable; that the robot had absolute knowledge about the poses of all objects of interest, the best constraints to satisfy and the right set of action primitives. However, robots operating in most interesting real world settings do not often have access to these capabilities. An exciting avenue for future work would be to ease the expense of predefining constraints. Could we learn to derive numerical constraints from natural language or from user demonstrations? Another potential avenue for future focus would be to enable robots to autonomously learn the right set of action primitives needed to complete a given task.

VIII. CONCLUSION

We tackled the problem of optimal constrained task planning by proposing an approach that encoded the entire task planning problem as a single MICP and solved it using an off-the-shelf MIP solver. We evaluated our approach on a set of optimal constrained task planning problems and demonstrated its ability to generate optimal plans including on a physical robot platform under kinematic constraints.

Algorithm	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5
Fast Downward [12]	0.102 ± 0.0003	0.104 ± 0.002	0.214 ± 0.001	0.218 ± 0.002	0.216 ± 0.008
Pyperplan [4]	0.167 ± 0.008	0.169 ± 0.012	0.169 ± 0.014	0.183 ± 0.012	0.190 ± 0.007
Forward Search [17]	0.0006 ± 0.004	0.0011 ± 0.001	0.0021 ± 0.0014	0.004 ± 0.002	0.008 ± 0.004
GTPMIP (ours)	0.039 ± 0.011	0.033 ± 0.0004	0.062 ± 0.009	0.172 ± 0.011	0.719 ± 0.014

TABLE I: Comparison of planning times (in seconds) of GTPMIP with those of state-of-the-art symbolic planners on purely symbolic block stacking problems with increasing number of blocks. Problem 1 and Problem 2 have 4 blocks, Problem 3 has 5 blocks, Problem 4 has 6 blocks and Problem 5 has 9 blocks.

Task	Constraints
Task 1	$x = 0.0 \cap 0.0 \leq y \leq 4.0$ $0.0 \leq x \leq 4.0 \cap y = 0.0$ $x = 4.0 \cap 0.0 \leq y \leq 4.0$
Task 2	$x = 0.0 \cap y = 5.0$ $x = 6.0 \cap y = 5.0$ $4.5 \leq y + 1.6x \leq 5.0 \cap \dots$ $0.0 \leq x \leq 3.0 \cap 0.0 \leq y \leq 4.0$ $-5.0 \leq y - 1.6x \leq -4.5 \cap \dots$ $3.0 \leq x \leq 6.0 \cap 0.0 \leq x \leq 4.0$
Task 3	$x = 3.0 \cap 0.0 \leq y \leq 6.0$
Task 4	$x = 0 \cap 0.0 \leq y \leq 4.0$ $0.0 \leq x \leq 3.0 \cap y = 4.0$ $0.0 \leq 3.0 \cap y = 0.0$
Task 5	$x = 0.0 \cap 0.0 \leq y \leq 5.0$ $0.0 \leq x \leq 3.0 \cap y = 4.0$ $x = 3.0 \cap 0.0 \leq y \leq 5.0$

TABLE II: The set of geometric constraints on package placements for each of the Warehouse package rearrangement tasks.

Task	HFG Building Time(s)	MICP Solving Time(s)
Task 1	21.12 ± 1.230	0.26 ± 0.100
Task 2	87.41 ± 10.160	0.60 ± 0.084
Task 3	5.20 ± 0.140	0.12 ± 0.004
Task 4	29.92 ± 0.450	0.31 ± 0.044
Task 5	32.06 ± 0.330	0.30 ± 0.046

TABLE III: Times (in seconds) for building the Hybrid Funnel Graphs(HFG) and for solving the resulting Mixed Integer Convex Program for each of the tasks.

REFERENCES

- [1] Bernardo Aceituno-Cabezas and Alberto Rodriguez. A global quasi-dynamic model for contact-trajectory optimization. In *Robotics: Science and Systems (RSS)*, 2020.
- [2] Alphonsus Adu-Bredu, Nikhil Devraj, Pin-Han Lin, Zhen Zeng, and Odest Chadwicke Jenkins. Probabilistic inference in planning for partially observable long horizon problems. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3154–3161. IEEE, 2021.
- [3] Alphonsus Adu-Bredu, Zhen Zeng, Neha Pusalkar, and Odest Chadwicke Jenkins. Elephants don’t pack groceries: Robot task planning for low entropy belief states. *IEEE Robotics and Automation Letters*, 7(1):25–32, 2022.
- [4] Yusra Alkhazraji, Matthias Frorath, Markus Grützner, Malte Helmert, Thomas Liebetraut, Robert Mattmüller, Manuela Ortlieb, Jendrik Seipp, Tobias Springenberg, Philip Stahl, and Jan Wülfing. Pyperplan. 2020.
- [5] Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. *ARTIFICIAL INTELLIGENCE*, 90:1636–1642, 1995.
- [6] Robin Deits and Russ Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *2014 IEEE-RAS international conference on humanoid robots*, pages 279–286. IEEE, 2014.
- [7] Stefan Edelkamp and Peter Kissmann. Optimal symbolic planning with action costs and preferences. In *Twenty-First International Joint Conference on Artificial Intelligence*. Citeseer, 2009.
- [8] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.
- [9] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Sampling-based methods for factored task and motion planning. *The International Journal of Robotics Research*, 37(13-14):1796–1825, 2018.
- [10] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022.
- [11] Peter Hart, Nils Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- [12] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.
- [13] Jörg Hoffmann. Ff: The fast-forward planning system. *AI magazine*, 22(3):57–57, 2001.
- [14] Jörg Hoffmann. The metric-ff planning system: Translating “ignoring delete lists” to numeric state variables. *Journal of artificial intelligence research*, 20:291–341, 2003.
- [15] François Robert Hogan and Alberto Rodríguez. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics. *arXiv preprint arXiv:1611.08268*, 2016.
- [16] Franc Ivankovic, Patrik Haslum, Sylvie Thiébaux, Vikas Shivashankar, and Dana Nau. Optimal planning with global numerical state constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 24, pages 145–153, 2014.
- [17] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.
- [18] Jon Lee and Sven Leyffer. *Mixed integer nonlinear programming*, volume 154. Springer Science & Business Media, 2011.
- [19] Hui X Li and Brian C Williams. Generative planning for hybrid systems based on flow tubes. In *ICAPS*, pages 206–213, 2008.
- [20] Agility Robotics. Digit robot.
- [21] R. Sharma and Y. Aloimonos. Coordinated motion planning: the warehouseman’s problem with constraints on free space. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(1):130–141, 1992.
- [22] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 639–646. IEEE, 2014.
- [23] Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *IJCAI*, pages 1930–1936, 2015.
- [24] Andrés Klee Valenzuela. *Mixed-integer convex optimization for planning aggressive motions of legged robots over rough terrain*. PhD thesis, Massachusetts Institute of Technology, 2016.
- [25] Menkes HL Van Den Briel and Subbarao Kambhampati. Optiplan: Unifying ip-based and graph-based planning. *Journal of Artificial Intelligence Research*, 24:919–931, 2005.
- [26] Thomas Vossen, Michael O Ball, Amnon Lotem, and Dana Nau. On the use of integer programming models in ai planning. Technical report, 1999.