Task and Motion Informed Trees (TMIT*): Almost-Surely Asymptotically Optimal Integrated Task and Motion Planning

Wil Thomason[1], Marlin P. Strub[2], and Jonathan D. Gammell[3]

***Abstract*—High-level autonomy requires discrete and continuous reasoning to decide both what actions to take and how to execute them. Integrated Task and Motion Planning (TMP) algorithms solve these hybrid problems jointly to consider constraints between the discrete symbolic actions (i.e., the *task plan*) and their continuous geometric realization (i.e., *motion plans*). This joint approach solves more difficult problems than approaches that address the task and motion subproblems independently.**

TMP algorithms combine and extend results from both task and motion planning. TMP has mainly focused on computational performance and completeness and less on solution optimality. Optimal TMP is difficult because the independent optima of the subproblems may not be the optimal *integrated* solution, which can only be found by jointly optimizing both plans.

This paper presents Task and Motion Informed Trees (TMIT*), an optimal TMP algorithm that combines results from makespan-optimal task planning and almost-surely asymptotically optimal motion planning. TMIT* interleaves asymmetric forward and reverse searches to delay computationally expensive operations until necessary and perform an efficient informed search directly in the problem's hybrid state space. This allows it to solve problems quickly and then converge towards the optimal solution with additional computational time, as demonstrated on the evaluated robotic-manipulation benchmark problems.

***Index Terms*—Task and Motion Planning, Motion and Path Planning, Manipulation Planning, AI-Based Methods**

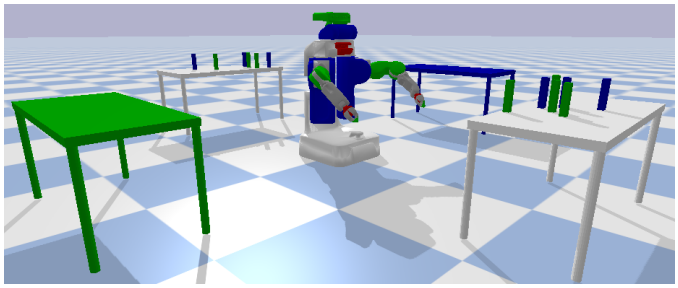


Fig. 1: A clutter clearing example. The robot must move the green and blue sticks from their initial positions scattered on two tables to ending positions on the corresponding colored tables. Sticks may occlude others, forcing the robot to reason about a geometrically feasible order of manipulation.

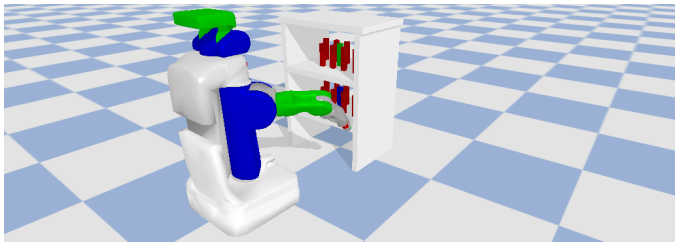


Fig. 2: A shelf rearrangement example. The robot must move the green stick from the upper shelf to the lower shelf and the blue stick from the lower shelf to the upper shelf. The red sticks block many grasps of the two target sticks and must be maneuvered around or moved to solve the problem.

I. Introduction

PLANNING solutions to problems described by high-level specifications requires autonomously deciding both *what* to do (i.e., the sequence of high-level actions) and *how* to do it (i.e., the associated motions). This is difficult since both of these decisions can affect later stages of the planning problem by altering the valid and reachable subsets of the search space. Integrated Task and Motion Planning (TMP) is a holistic approach to solve these high-level planning problems by jointly considering the symbolic (i.e., actions) and geometric (i.e., motion) constraints on the solution.

Solving TMP problems is computationally expensive. Evaluating a candidate sequence of actions (i.e., a *task* or *symbolic* plan) requires the computationally expensive operations of finding compatible action parameters and associated valid motion plans. TMP algorithms typically consider multiple symbolic plans to solve a problem [1] and must be efficient because the set of possible symbolic plans is combinatorially large for most real-world scenarios and the sets of possible action parameters and motion plans are uncountably infinite.

The majority of TMP work is focused on improving the computational performance of algorithms instead of finding optimal solutions [2]. Optimal TMP requires joint optimization of candidate symbolic plans and the corresponding action parameters and motion plans, which is more difficult than the independent optimal symbolic and optimal motion planning problems. Exist-

Manuscript received: February 24, 2022; Revised: July 19, 2022; Accepted: July 23, 2022.

This paper was recommended for publication by Editor Hanna Kurniawati upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported by the National Defense Science & Engineering Graduate Fellowship and UK Research and Innovation/EPSRC through ACE-OPS: From Autonomy to Cognitive assistance in Emergency OPerationS [EP/S030832/1]. We are grateful for this support. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this submission.

[1]Department of Computer Science, Rice University, Houston, Texas, USA. wbthomason@rice.edu

[2]Jet Propulsion Laboratory, California Institute of Technology. Work done at University of Oxford. marlin.p.strub@jpl.nasa.gov

[3]Estimation, Search, and Planning (ESP) Group, Oxford Robotics Institute, University of Oxford, UK. gammell@robots.ox.ac.uk

Digital Object Identifier (DOI): see top of this page.

ing solution-optimal TMP algorithms [3–5] have poor initial solution performance, require problem specific samplers and local motion planners, and/or are specific to manipulation planning.

This paper presents Task and Motion Informed Trees (TMIT*), an optimal sampling-based TMP algorithm that extends results from makespan-optimal symbolic planning and almost-surely asymptotically optimal motion planning. TMIT* efficiently searches the high-level problem's hybrid state space, which consists of both discrete and continuously valued dimensions. This direct search allows TMIT* to be informed by both the symbolic and geometric constraints and to reuse motion planning effort as symbolic plans change.

TMIT* interleaves asymmetric forward and reverse searches [6] to identify geometrically infeasible plans and avoid computationally expensive action-parameter sampling and motion-planning operations. When combined with novel SMT-based (Satisfiability Modulo Theories [7]) symbolic planning and a differentiable distance-based predicate representation [8], this allows TMIT* to quickly find initial solutions to high-level problems and almost-surely converge asymptotically to the optimum with additional computational time. We demonstrate the benefits of this approach on robotic-manipulation benchmark problems (Figs. 1 and 2), where TMIT* significantly outperforms earlier nonoptimal TMP work in initial solution times and is able to decrease solution costs given additional computational time.

This work contributes methods for (1) almost-surely asymptotically optimal TMP that integrates relaxed SMT-based symbolic planning and sampling-based motion planning, (2) anytime approximation of action-precondition-satisfying state manifolds to avoid state discretization, (3) motion guided deferral of expensive precondition satisfying state sampling, and (4) adaptive prioritization to order the search of a multimodal space.

II. Background and Related Work

TMP is an active area of research drawing from advancements in the motion planning and task planning literatures. TMIT* frames TMP as multimodal motion planning (Sec. II-A). It offers almost-surely asymptotically optimal TMP (Sec. II-B) with practical performance by extending work on batch-sampling-based motion planning (Sec. II-C) and incremental symbolic planning (Sec. II-D). TMIT* uses deferred sampling of continuous action parameters to improve its computational efficiency (Sec. II-E).

A. Multimodal Motion Planning and TMP

Multimodal motion planning finds valid motions through sequences of *modes*, or discrete variations of a continuous configuration space, each imposing different constraints on the valid configurations [9, 10]. Manipulation planning is often modeled as multimodal motion planning [10–14], where modes correspond to choices of object grasps and placements. TMP problems also have multimodal structure [8, 13, 15] with modes corresponding to different symbolic states that constrain the continuous (i.e., geometric) state components. Mode transitions are defined by symbolic high-level actions and link the task planning and motion planning subproblems. Framing TMP as multimodal motion planning allows approaches to avoid the backtracking necessary in most other formulations.

B. Almost-Surely Asymptotically Optimal TMP

Most TMP work focuses on efficiently finding initial solutions. Cost-aware or optimal TMP has only recently become a topic of interest to the field [4]. Earlier optimal TMP work [3] frames TMP as a nonlinear optimization problem and globally optimizes choices of action parameters and motions. This optimization-based approach does not scale well with plan length, action space size, and geometric complexity. Schmitt et al. [16] offer almost-surely asymptotically optimal manipulation planning based on a precomputed roadmap and domain-specific samplers for mode transitions. Garrett et al. [17] produce TMP solutions that are optimal in symbolic action cost, but assume domain-specific samplers and do not jointly optimize their motions and action parameter choices. Recent results [5] have shown that almost-sure asymptotic optimality results from pure motion planning are preserved for multimodal TMP under realistic assumptions on the measure of mode transition sets.

TMIT* uses almost-surely asymptotically optimal sampling based motion planning for performant planning in high dimensional state spaces. It does not require any precomputation or domain-specific samplers, and solves both manipulation and more general TMP problems.

C. Batch-Sampling-Based Motion Planning

BIT* [18] approaches sampling-based motion planning by viewing batches of valid samples as vertices in a series of edge-implicit *random geometric graphs* (RGGs). This perspective allows planners to order their search in a principled manner and incorporate problem-specific information, such as cost heuristics. AIT* [6] builds on this idea with an asymmetric bidirectional search. The reverse search computes an accurate cost heuristic without edge validation to guide the forward search for a solution, focusing the forward search and reducing the number of fully evaluated edges.

TMIT* extends AIT* to plan in multimodal spaces. AIT*'s lazy reverse search allows TMIT* to check the geometric feasibility of actions in a candidate symbolic plan before committing to computing its full motion plan. Toussaint [3] uses a similar hierarchy of feasibility checks to filter the space of symbolic plans.

D. Symbolic Planning for TMP

TMP solvers adapt advances in standalone symbolic planning to the TMP context [1, 17, 19, 20]. Symbolic planning for TMP is uniquely challenging since valid symbolic plans may not correspond to valid motion plans. TMP symbolic planners must be able to efficiently incorporate geometric feasibility constraints to generate alternative plans [1]. TMP solvers have addressed this need with custom action formulations [17, 19], top-k symbolic planning with Monte Carlo tree search [21], and incremental constraint construction [1].

TMIT* extends the incremental Boolean-satisfiability-based (SAT-based) symbolic planner proposed by Kautz and Selman

[22] and adapted for TMP by Dantam et al. [1]. Compared to this original adaptation, we neither include symbolic representations of continuous scenegraphs nor prediscretize the continuous problem state. We also solve a relaxed symbolic planning problem, do not require a translation step between discrete and continuous state, and use a new feature of the Z3 SMT solver [23] to improve planner performance and extensibility. Other work has encoded the full Planning Domain Definition Language+ (PDDL+) symbolic language into SMT [24]. TMIT* uses a subset of the simpler, more common PDDL 2.1.

E. Deferred Action Parameter Sampling

Finding geometrically valid values for continuous action parameters (e.g., grasps and placement poses) is one of the core challenges of TMP [2]. Recent work [17, 25] attempts to reduce the computational burden of finding valid parameters by validating actions incrementally.

TMIT* adopts the differentiable distance function predicate representation of [8]. This representation allows us to directly sample continuous states satisfying symbolic action preconditions and guide the motion planner toward these states. This guidance lets TMIT* defer sampling action parameters until it has evidence from the motion planner that an action is both geometrically feasible and likely to be part of a valid solution.

III. Problem Formulation

A robot, R, comprises a kinematic tree of joint-connected links, $l \in \mathcal{L}$, and a mobile base with pose $P(R) \in \mathrm{SE}(2)$[1]. An object, $o \in \mathcal{O}$, is a physical entity in the robot's environment with an associated 3D geometry and pose, $P(o) \in \mathrm{SE}(3)$.

A predicate defines a property of the combined robot and object configuration (Def. 1) as a relation. A symbol is a predicate applied to specific arguments (Def. 2).

Definition 1: *Predicate*
A predicate, $p \in \mathcal{P}$, is a function from a set of objects and/or robot links, $p_{\mathcal{O}\cup\mathcal{L}} \subseteq \mathcal{O} \cup \mathcal{L}$, to a Boolean domain, defining a problem-dependent property. Formally, $p : \prod_{n_p} p_{\mathcal{O}\cup\mathcal{L}} \to \mathbb{B}$, where n_p is the arity of p, and $\prod$ is the Cartesian product. Predicates may define properties with *geometric* or *discrete* interpretations (Sec. IV-A).

Definition 2: *Symbol*
A symbol, $s \in \mathcal{S}$, is the application of a predicate, p, to specific arguments (also known as a proposition or ground predicate). A symbol is `true` in a state if the resulting values of its arguments satisfy the associated predicate's property, and is `false` otherwise[2]. Some symbols correspond to differences in the free configuration space, e.g., a symbol representing the robot's manipulator holding a specific object.

The state space for our problem is the combination of the discrete and continuously valued dimensions (Def. 3). A mode is a subset of the state space with a unique, fixed setting of the discrete state and poses of ungrasped movable objects, and a *mode family* refers to an infinite set of modes which share their discrete state setting (Def. 4) [9]. A motion is a continuous path through the state space (Def. 5).

Definition 3: *State Space*
The state space of the planning problem is the Cartesian product of the continuous robot configuration and object poses and the discrete symbolic state,

$$\mathcal{Q} = \mathcal{Q}_R \times \mathcal{Q}_{\mathcal{O}} \times \mathbb{B}^{|S|}, \tag{1}$$

where $\mathcal{Q}_R$ is the robot configuration space [26], $\mathcal{Q}_{\mathcal{O}}$ is the space of all the objects' poses, and $\mathbb{B}^{|S|}$ is a Boolean domain representing the value of the discrete symbols.

Definition 4: *Modes and Mode Families*
A mode family, $\mathbb{M} \subseteq \mathcal{Q}$, of a specific setting of the discrete state, $\xi \in \mathbb{B}^{|S|}$, is defined as:

$$\mathbb{M} = \{(\boldsymbol{q}_R \in \mathcal{Q}_R, \boldsymbol{q}_{\mathcal{O}} \in \mathcal{Q}_{\mathcal{O}}, \xi)\} \tag{2}$$

A mode, $\mathcal{M} \subseteq \mathbb{M}$, for a specific set of movable object poses, $\rho \in \mathcal{Q}_{\mathcal{O}}$, is defined as:

$$\mathcal{M} = \{(\boldsymbol{q}_R \in \mathcal{Q}_R, \rho, \xi)\} \tag{3}$$

Definition 5: *Motion*
A motion between two states, $\boldsymbol{q}_i, \boldsymbol{q}_f \in \mathcal{Q}$, is a sequence of states described by a function, $\sigma : [0, 1] \to \mathcal{Q}$, that starts at the initial state, $\sigma(0) = \boldsymbol{q}_i$, ends at the final state, $\sigma(1) = \boldsymbol{q}_f$, and is continuous at all points, $\forall t, \lim_{\tau \to t} \sigma(\tau) = \sigma(t)$. By definition, a motion cannot change the discrete symbolic state.

Discrete actions are defined by their necessary preconditions and their resulting effects (Defs. 6 to 8).

Definition 6: *Precondition*
A precondition or constraint, $\phi : \mathcal{Q} \to \mathbb{B}$, is a function that evaluates a Boolean propositional logic formula at a state.

Definition 7: *Effect*
An effect, $\psi : \mathcal{Q} \to \mathcal{Q}$, is a deterministic function that possibly modifies a state in a discontinuous manner. An effect may alter the continuous and discrete parts of a state, but many TMP action effects only modify the discrete part.

Definition 8: *Action*
An action, $\alpha \in \mathcal{A}$, $\alpha = (\phi_\alpha, \psi_\alpha)$, comprises the preconditions, ϕ_α, necessary to execute the action and the effects, ψ_α, of the action on the state when successfully executed, as in symbolic planning [27]. The set of actions, $\mathcal{A}$, includes the *null action*, $\perp$, defined such that, $\forall \boldsymbol{q} \in \mathcal{Q}$, $\phi_\perp(\boldsymbol{q}) = \texttt{true}$, and, $\forall \boldsymbol{q} \in \mathcal{Q}$, $\psi_\perp(\boldsymbol{q}) = \boldsymbol{q}$. An action can be considered an abstraction of a specific high-level robot skill (e.g., grasping, pushing, pouring, etc.).

The TMP problem is then formally defined as the search for a plan of actions and motions (Def. 9).

Definition 9: *Task and Motion Planning (TMP) Problem*
Let $\boldsymbol{q}_0 \in \mathcal{Q}$ be an initial state, ϕ_g be a goal specified as a constraint, and $\mathcal{A}$ be a set of discrete actions executable by a robot. The TMP problem is then formally defined as the search for motions, $\sigma_i \in \Sigma$, and symbolic actions, $\alpha_i \in \mathcal{A}$, that can be interleaved into a task-and-motion plan, $(\sigma_1, \alpha_1, \sigma_2, \alpha_2, \ldots, \sigma_n, \alpha_n)$, such that:

[1]TMIT* is not limited to planar mobile robots.
[2]TMIT* extends to other SMT-encodable discrete symbolic domains.

(1) The plan begins at the initial state, $\sigma_1(0) = \boldsymbol{q}_0$.
(2) Robot and object motions are valid (e.g., collision free),

$$\forall i = 1, 2, \ldots, n,\ \forall t \in [0, 1],\ \texttt{is_valid}\left(\sigma_i(t)\right) = \texttt{true}.$$

(3) The final state of each motion, $\sigma_i(1)$, satisfies the requirements to execute the following action, α_i,

$$\forall i = 1, 2, \ldots, n,\ \phi_{\alpha_i}(\sigma_i(1)) = \texttt{true}.$$

(4) The effect of each intermediate action, α_i, results in the initial state of the following motion, $\sigma_{i+1}(0)$, and

$$\forall i = 1, 2, \ldots, n-1,\ \psi_{\alpha_i}(\sigma_i(1)) = \sigma_{i+1}(0).$$

(5) The effect of the final action, ψ_{α_n}, meets the specified goal constraint, $\phi_{\text{g}}(\psi_{\alpha_n}(\sigma_n(1))) = \texttt{true}$.

Optimal TMP finds TMP solutions which optimize a given cost function (Def. 10).

Definition 10: *Optimal TMP*
Let Θ be the set of all valid solutions to a TMP problem (Def. 9) and $\gamma : \Theta \to \mathbb{R}^{\geq 0}$ be a cost function. The optimal TMP problem is the search for a lowest cost solution, $\theta^* \in \Theta$, such that $\theta^* = \arg\min_{\theta \in \Theta} \gamma(\theta)$

Solving TMP as independent symbolic and motion planning problems is commonly infeasible [2]; TMIT* presents a holistic approach that solves the integrated problem.

IV. Task and Motion Informed Trees (TMIT*)

TMIT* (Fig. 3) plans in a multimodal hybrid state space [8] in which any valid path, annotated with symbolic actions at certain states, is a valid task-and-motion plan. It uses an incremental SMT-based symbolic planner inspired by Dantam et al. [1] on a relaxed problem to find a candidate symbolic plan, defining a sequence of hybrid state space modes. It samples batches of states along this mode sequence and uses a distance-based geometric predicate representation [8] to detect when samples are within the connection radius [28] of precondition-satisfying states for an action in the plan. Sampling these precondition-satisfying states corresponds to choosing continuous action parameters (e.g., grasps) and allows TMIT* to connect to the next reachable modes.

If this marched sampling procedure does not reach the goal mode then the symbolic plan is infeasible at the current resolution and TMIT* uses knowledge of the modes in which it failed to inform the search for a new symbolic plan. Sampling resumes without discarding old symbolic plans or samples to reuse motion planning effort and allow increased sample resolution to prove symbolic plan feasibility. If the sampling procedure does reach the goal mode then the symbolic plan is validated further via an asymmetric bidirectional search [6]. This search process continues until a solution is found and almost-surely converges asymptotically towards the jointly optimal task-and-motion solution.

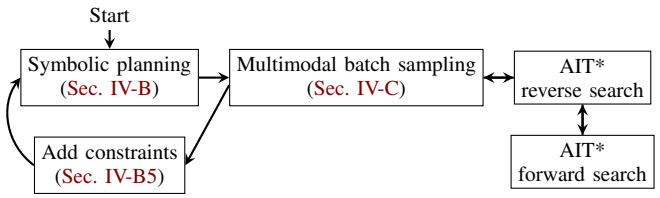


Fig. 3: The TMIT* planning loop. TMIT* cycles between generating candidate symbolic plans, (Sec. IV-B), sampling batches of states in the modes traversed by the candidate plans (Sec. IV-C), and the reverse and forward search stages of AIT* [6]. If it fails to reach the goal mode, it adds constraints (Sec. IV-B5) and returns to the symbolic planner for a new candidate. This process continues until it finds a valid path to the goal.

A. Predicate representation

TMIT* uses a split predicate representation to simplify the symbolic planning subproblem and guide the motion plan search toward action-precondition-satisfying states.

We partition the set of predicates into those with *geometric* properties, $\mathcal{P}_{\text{G}} \subseteq \mathcal{P}$, and those with *discrete* properties, $\mathcal{P}_{\text{D}} = \mathcal{P} \setminus \mathcal{P}_{\text{G}}$. Some predicates have natural categorizations, e.g., a discrete predicate representing a light's state or a geometric predicate describing distance to an object. For others, the correct interpretation may be problem specific, e.g., a predicate for an object being on a specific surface. This distinction aids in representing and solving TMP problems.

Discrete predicates comprise the discrete part of the state space (Def. 3). Geometric predicates have associated differentiable functions defining the distance from a given state to a nearest predicate-satisfying state. The distance for a geometric predicate, $p \in \mathcal{P}_{\text{G}}$, is zero at a given state, $\boldsymbol{q} \in \mathcal{Q}$, if and only if the state, $\boldsymbol{q}$, satisfies the predicate, p. We project uniform-randomly sampled states onto the zero-level set of this function via gradient-based optimization [29].

Predicate distance functions can be defined in different ways [8, 30, 31]. We automatically derive distance functions for symbolic formulae with "unsatisfaction semantics" [8].

B. Symbolic planning

TMIT* solves the symbolic planning subproblem with an incremental SMT-based symbolic planning algorithm inspired by Dantam et al. [1]. SMT solvers generalize SAT by extending the variable types (to include integers, real numbers, arrays, etc.) and relations, which are backed by efficient solvers for the corresponding formal theories [7].

Basic SAT-based symbolic planning [32] (1) creates Boolean variables for each action and symbol, (2) adds constraints encoding the initial and goal state, (3) constrains variables to remain consistent between steps (i.e., the "frame axioms"), (4) constrains actions to imply their preconditions and effects, and (5) iteratively increases the number of plan steps until the resulting formula is satisfiable. Dantam et al. [1] extend this core approach by encoding discretized geometric state and using the Z3 SMT solver's [23] incremental constraint stack to reuse solver effort when adding plan steps.

We further extend the approach of Dantam et al. [1] with a custom theory that improves performance and expressivity. We specificially (1) enforce the frame axioms, precondition

constraints, and other planning constraints implicitly via Z3's *user propagator*, reducing the number of long-lived clauses, (2) relax the symbolic planning problem by omitting geometric propositions from precondition formulae, and (3) do not discretize the continuous state. We outline TMIT*'s symbolic planner in Secs. IV-B1 to IV-B5.

1) State encoding: In contrast to [1], we create Boolean indicator variables only for each *discrete* symbol (i.e., those corresponding to predicates in $\mathcal{P}_{\mathrm{D}}$) at each step of the plan and omit the geometric symbols. We also do not discretize or symbolically represent the geometric state in the symbolic planning problem, relaxing the symbolic planning problem by optimistically ignoring constraints from geometric predicates. This frees the SMT solver from explicitly considering geometric relations.

2) Action encoding: We create a Boolean indicator variable, $a_\alpha^j \in \mathbb{B}$, for each action, α, at each step j. This indicator variable is `true` if the plan takes the associated action at step j and `false` otherwise. Our custom theory implicitly enforces the precondition and effect constraints, $a_\alpha^j \implies \phi_\alpha^{j-1} \wedge \psi_\alpha^j$, where ϕ_α^{j-1} is the precondition of α encoded using the variables for the relevant discrete symbols at step $j-1$, and ψ_α^j is the effect of α applied at step j. When Z3 assigns a value to an action indicator variable, we assert any unsatisfied clauses in the relevant precondition and effect constraints. This improves performance by reducing the number of clauses for the SMT solver to validate.

3) Frame axioms and action mutexes: We also implicitly enforce the frame axioms and action mutex constraints. The frame axioms specify that a symbol's value at step j is the same as at the preceding step unless an action that modifies it is chosen at step j. This ensures that variable values remain consistent between steps. When Z3 assigns a symbol indicator variable's value at a step, we assert the relevant frame axiom. Action mutex constraints force Z3 to choose only a single action per step.

4) The symbolic planning loop: The symbolic planning loop is a typical incremental SAT-based planner [1]. We first assert the initial discrete state and add plan steps up to a heuristically determined minimum plan length[3]. Each step adds constraints asserting the discrete state at the step and the action transition constraints and frame axioms (Secs. IV-B2 and IV-B3). We then assert the goal constraint and invoke the Z3 SMT solver [23] to attempt to find a solution. We add new steps and reinvoke the solver until a satisfying assignment exists. We extract a plan from a solution by checking which action indicator variables are `true` at each step.

5) Generating alternative plan candidates: TMIT* has two methods of generating alternative symbolic plan candidates. The first enumerates all candidate symbolic plans by requiring new plans to differ from previous plans by at least one action, ensuring completeness [1]. This method adds constraints:

$$\neg \bigwedge_{1 \le j \le n} \left(\bigwedge_{\alpha \in \mathcal{A}} a_\alpha^j = \mathtt{value}_k(a_\alpha^j) \right) \tag{4}$$

for a current solution with length n, and where $\mathtt{value}_k$ returns the value of a variable in the k-th candidate plan.

A *prefix* of a symbolic plan is a subsequence of the actions in the plan starting with the initial action. The second method forces new plans to avoid *failing prefixes* of symbolic plan candidates by adding constraints of the same form as Eq. (4), but only until the first step for which precondition-satisfying state sampling failed. Precondition-satisfying state sampling may fail if sample projection (Sec. IV-A) fails to converge, if it converges to a local minimum off the precondition-satisfying manifold, or if the resulting state is invalid (i.e., in collision). This can quickly eliminate broader groups of candidate symbolic plans and find a solution more efficiently, but may remove prefixes that would prove feasible with more computation. We use this prefix-blocking method in the experiments of Sec. VI.

C. Motion Planner Integration

TMIT* solves the motion planning subproblem by building upon AIT* [6], an almost-surely asymptotically optimal batch-sampling-based motion planner, to find geometrically valid instantiations of candidate symbolic plans. We extend AIT*'s batch sampling to (1) sample states in each reachable mode and (2) sample action precondition-satisfying states only if a uniform sample is within a tunable distance threshold (e.g., the connection radius [28]) of the precondition-satisfying region.

A mode is *reachable* if either it is the initial mode (i.e., the initial discrete state and scene) or we have sampled a precondition-satisfying state for an action that transitions to it from a reachable mode. Sampling batches across the reachable modes uniformly increases the resolution of AIT*'s RGG (Sec. II-C) and allows TMIT* to implicitly reevaluate old candidate symbolic plans without backtracking.

The set of precondition-satisfying states for an action is often a manifold of measure zero in the ambient configuration space due to dimensionality-reducing constraints and therefore has zero probability of being sampled with uniform-random sampling. Computing precondition-satisfying states is computationally expensive relative to uniform-random sampling, and many such states are challenging to reach with a motion plan (e.g., states close to objects being manipulated).

We avoid wasted effort by projecting uniform-random samples onto precondition-satisfying manifolds only when the samples are within a tunable distance threshold (e.g., [28]) of the manifold[4], which effectively *inflates* the manifold to positive measure. This strategy avoids computing unusable precondition-satisfying samples by only invoking this process starting from states that are (1) in the RGG and (2) close enough to a precondition region to improve the solution cost. Starting from valid states close to a precondition region may additionally improve the likelihood that the resulting precondition-satisfying sample will be valid. Alg. 1 shows the multimodal batch sampling function. `SampleValid` draws each state uniformly at random from the valid configuration space of a given mode. The viable actions (line 8) of a mode, $\mathcal{M} \in \mathbb{M}$, are symbolic actions that (1) are used at $\mathbb{M}$ in a candidate symbolic plan and (2) have been attempted less than a heuristically determined number of times. Attempting an action means trying to sample a valid state satisfying its precondition constraint. The heuristic limit on attempts per

[3]Zero, unless we have a better estimate for a problem.

[4]Sampling a precondition-satisfying state takes on the order of 10–100 optimizer iterations; testing the distance takes less than one [8].

Algorithm 1: Multimodal batch sampling

Input: Mode queue ω, connection radius μ, goal ϕ_g
Output: Batch of samples B

$B \leftarrow \{\}$
while $|\omega| > 0$: // All reachable modes
  $\mathcal{M} \leftarrow$ `pop`(ω)
  while `NeedSamples`(B):
    $s \leftarrow$ `SampleValid`$(\mathcal{M})$
    $B \leftarrow B \cup \{s\}$
    if $\phi_g(s)$: **return** B
    for $\alpha_i \in \mathcal{M}$.`viable_actions`:
      if $d(\phi_i, s) < \mu$:
        $s' \leftarrow$ `SamplePrecond`(ϕ_i, s)
        if `IsValid`(s'):
          $B \leftarrow B \cup \{s'\}$
          `UpdateModes`(ω, s')
if `NoActions`$()$: `IncreaseBudget`$()$
if not `AtGoal`$()$ **or** `NoActions`$()$: `NewTaskPlan`$()$
return B

action provides a budget of computation per candidate symbolic plan; `IncreaseBudget` increments this heuristic threshold. `NoActions` tests if any actions in any reachable mode are viable, and $d(\cdot, \cdot)$ returns the distance from a state to the nearest precondition-satisfying state. `SamplePrecond` projects the given state onto the manifold of precondition-satisfying states by gradient-based optimization. `UpdateModes` adds newly reached modes to the mode queue, `AtGoal` checks if the total set of samples contains goal mode states, and `NewTaskPlan` invokes the task planner (Secs. IV-B4 and IV-B5).

The mode queue is ordered to prioritize recently reached modes. This ordering creates behavior akin to the "enforced hill climbing" of the FastForward (FF) task planner [33] by continuing the search in the resulting mode when an action succeeds, effectively following the corresponding task plan candidate as far as possible. Alg. 1 returns early if it reaches a goal-satisfying state. The mode queue persists across invocations of Alg. 1 for the same batch.

D. Considerations for Multimodal AIT*

Computing exact distance in the hybrid configuration space is PSPACE-complete[5] [34]. We conservatively over-approximate configuration space distance by assuming states in different modes are infinitely far apart if we do not have a successful transition between their modes. This approximation is not a metric, but suffices in practice.

AIT* uses the reverse search to calculate a heuristic to guide the forward search (see [6] for details). This approximation must account for the conservative over-approximation of intermode distance. The reverse search therefore uses forward-direction transitions and distances between modes (distances are directionally symmetric within a mode).

E. Implementation

We provide a proof-of-concept implementation of TMIT* in C++[6]. We use the implementation of AIT* and associated sampling-based motion planning utilities from the Open Motion Planning Library [35] and use Bullet [36] for collision checking. The geometric predicate implementation is an improved version of [8] using the Autodiff [37] library for automatic differentiation, NLOpt [38] for optimization, and a bespoke dual-number automatic differentiation implementation in LuaJIT for predicate functions.

The planner's input is simpler than most other TMP solvers and does not include specialized samplers or planners, or prediscretized state. It requires only a description of the initial scene, a specification of the robot morphology and kinematics, object and robot geometries, the symbolic planning domain and problem, and functions for geometric predicates.

V. Analysis

The probabilistic completeness and almost-sure asymptotic optimality of TMIT* follow from the corresponding properties of AIT*. We sketch proofs of these properties for TMIT*. In the following, assume that precondition regions are convex, and that all precondition-satisfying states in a complete feasible plan are surrounded by a valid hyperball [5].

Theorem 1: *Probabilistic Completeness*
Given a TMP problem as in Def. 9, the probability that TMIT* does not find a solution goes to zero as the number of samples taken approaches infinity.

Proof. TMIT* eventually attempts every possible symbolic plan candidate, since SATPlan (Sec. IV-B) is complete [32], and the constraints described in Sec. IV-B5 ensure that candidate symbolic plans are distinct. The symbolic planner will be invoked infinitely often until a TMP solution is found, since Alg. 1 requests a new task plan whenever it fails to reach the goal mode or runs out of viable actions. Actions have a finite attempt budget and the mode queue is always emptied before completing a batch. AIT* is probabilistically complete [6] and will eventually sample within the connection radius of each precondition region infinitely often. Each precondition-satisfying manifold is convex by assumption, so projecting uniform-random samples onto the precondition regions will eventually sample every point in the manifold. Thus, we will find a valid path through our planning space if one exists, and therefore are probabilistically complete. □

The sketch of almost-sure asymptotic optimality is similar.

Theorem 2: *Almost-Sure Asymptotic Optimality*
Given an optimal TMP problem as in Def. 10, TMIT* converges asymptotically to an optimal-cost solution with probability one as the number of samples approaches infinity.

Proof. AIT* is asymptotically optimal within each mode and its reverse search heuristic is computed globally across modes. TMIT* samples precondition-satisfying states whenever a uniform-random sample is within the connection radius of the precondition region. TMIT*'s symbolic planner is complete, each precondition region is inflated to positive measure, and precondition regions are convex by assumption, so TMIT* will asymptotically sample better precondition-satisfying states for every viable action. Thus, as the number of uniform-random

[5]The distance between two states is a function of the shortest sequence of mode transitions between them, which is an optimal symbolic plan.
[6]https://robotic-esp.com/code/tmitstar

samples approaches infinity, we asymptotically converge to the optimal action sequence and associated optimal motion plan (the optimal task and motion plan) with probability one. □

VI. Evaluation

Directly comparing TMP solvers is challenging due to differing definitions of the TMP problem. TMIT* does not assume prediscretization of continuous state [1] or specialized blackbox precondition samplers [17], which makes its problems harder. We perform a cold-data comparison to Planet [8], as its problem definition and assumptions are closest to TMIT*'s. All experiments were run on an AMD Ryzen 7 2700X CPU with 32 GB of RAM and use a PR2 robot model with 14 controllable joints and a planar mobile base (17 total degrees of freedom). We evaluate on two common TMP tasks: clutter clearing and shelf rearrangement. We use batches of 50 samples per mode, a batch budget (the number of batches before requesting a new symbolic plan candidate) of five for clutter clearing and two for shelf rearrangement, and an initial action-attempt budget of one. We use Euclidean distance within modes and optimize path length.

a) Clutter Clearing: A set of colored sticks is initially scattered on one of two tables (Fig. 1). The goal is to place each stick on one of two other tables corresponding to its color. Every stick must be manipulated, and sticks occlude others in their initial positions, so solutions require reasoning over a long series of symbolic actions to move the sticks in a geometrically valid order. The action space contains $16 \times |\mathcal{O}|$ symbolic actions for each instance size and each symbolic action has an infinite number of possible continuous instantiations. Fig. 4 shows results for clutter clearing.

b) Shelf rearrangement: The robot must retrieve and move a set of objects between two stacked shelf surfaces (Fig. 2). The target objects are initially placed deep within the shelves and are surrounded by increasing numbers of distractor objects. Motions are geometrically constrained by the shelves and the distractor objects must either be moved out of the way or maneuvered around to reach the target objects. The action space contains $6 \times |\mathcal{O}|$ symbolic actions for each instance size. Fig. 5 shows results for shelf rearrangement.

A. Qualitative Discussion

Fig. 4a shows that TMIT* significantly outperforms Planet in initial solution time, often by an order of magnitude. The results for Planet constitute 10 successful trials of each instance size; in contrast, the TMIT* results constitute 100 trials where timeouts are considered to have taken infinite time. Fig. 4b shows that TMIT* finds initial solutions for most instances quickly but that larger problem instances are more likely to either time out or have higher variance in initial solution times. This distribution reflects the greater geometric and symbolic challenge of the more complex instances. Fig. 4c shows median solution costs for 100 trials of TMIT* on instances of clutter clearing with 3–5 target objects. While the largest drop in median cost corresponds to initial solution discovery, the trends show that TMIT* makes consistent progress toward lower-cost solutions.

Fig. 5a demonstrates the relative effect of minimum solution length versus the number of objects in a problem environment on TMIT*'s initial solution performance. Although there are more objects present in large shelf rearrangement instances than large clutter clearing instances, the number of actions necessary to solve a shelf rearrangement problem is generally lower than the number required for a comparably large clutter clearing problem. This results in TMIT* finding solutions for large shelf rearrangement instances faster than for large clutter clearing problems. TMIT*'s motion-planner-guided sampling of continuous action parameters also sometimes allows it to find grasp poses for the target blocks that carefully reach past the distractor objects and reduce the overall plan length. Fig. 5c shows TMIT* optimizing costs for instances of the shelf rearrangement problem.

VII. Conclusions

TMIT* is a novel approach to almost-surely asymptotically optimal TMP. It extends work on constraint-based symbolic planning [1], distance-based predicate representation [8], and batch-sampling-based optimal motion planning [6]. TMIT* solves a relaxed symbolic planning problem with a novel SMT-based makespan-optimal symbolic planner to generate candidate sequences of actions, then attempts to find geometrically valid instantiations of these actions through asymmetric bidirectional batch-sampling-based motion planning in a hybrid multimodal state space. It uses a differentiable distance-based representation of geometric predicates to guide parameter sampling and sample action-precondition-satisfying states through gradient-based optimization. When candidate symbolic plans are not feasible, it generates alternatives by blocking invalid action sequence prefixes; however, it is able to continue to consider older candidate plans without backtracking by continuing the motion planning process.

Asymmetric bidirectional motion planning is well-suited to TMP because it gains information about action feasibility before paying the cost of validating a candidate plan's edges. Incrementally improving a RGG further allows planners to reuse motion planning effort across candidate plans. Future work may investigate using asymmetric bidirectional motion planning algorithms better suited for complex cost functions, such as Effort Informed Trees (EIT*) [6].

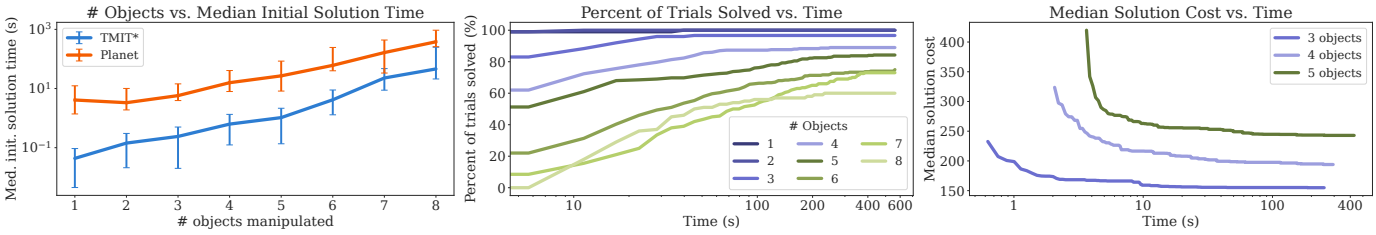
Encoding task planning as SMT via a custom theory offers untapped potential performance improvements for TMP. It provides an easy extension point for a "theory of TMP", incorporating geometric information such as reachability or action feasibility into the symbolic planner. We leave exploration of this capacity for future work.

Future work could also investigate accelerating the discovery of initial solutions by explicitly biasing RGG growth toward task-relevant regions.

References

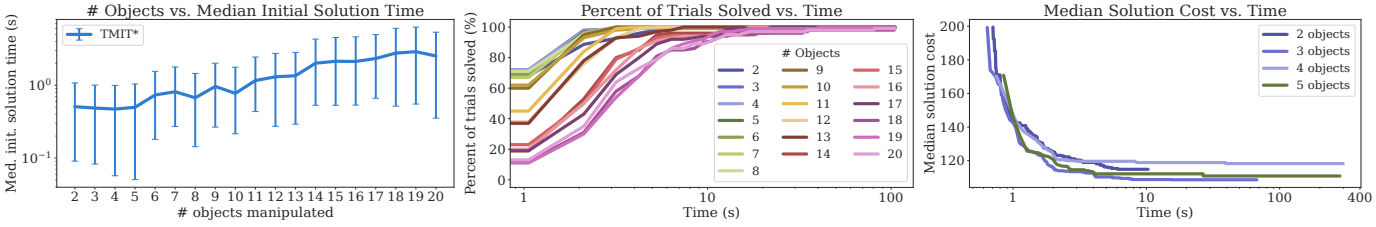
[1] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki. "An Incremental Constraint-Based Framework for Task and Motion Planning". In: *IJRR* 37.10 (2018).

[2] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. "Integrated Task and Motion Planning". In: *Annu. Rev. Control Robot. Auton. Syst.* (2020).



(a) (b) (c)

Fig. 4: Results for 100 trials of clutter clearing. Figs. 4a and 4c show median time or cost (respectively); the error bars in Fig. 4a show a 95% confidence interval of the median. Each trial had 600 seconds of planning time; trials which failed to find a solution within this bound were counted as having infinite duration and cost. Trials in Fig. 4c terminated early if their cost converged. Fig. 4b show the cumulative percentage of problems solved as a function of time for each instance size. Timeouts are more frequent for larger instance sizes, reflecting the greater geometric difficulty of the problem.



(a) (b) (c)

Fig. 5: Results for 100 trials of the shelf rearrangement problem. Figs. 5a and 5c show median time or cost (respectively); the error bars in Fig. 5a show a 95% confidence interval of the median. Each trial had 300 seconds of planning time; trials which failed to find a solution within this time were counted as having infinite duration and cost. Fig. 5b shows the cumulative percentage of problems solved as a function of time for each instance size.

[3] M. Toussaint. "Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning". In: IJCAI. 2015.

[4] W. Vega-Brown and N. Roy. "Asymptotically Optimal Planning under Piecewise-Analytic Constraints". In: WAFR. 2016.

[5] R. Shome, D. Nakhimovich, and K. E. Bekris. "Pushing the Boundaries of Asymptotic Optimality in Integrated Task and Motion Planning". In: WAFR. 2020.

[6] M. P. Strub and J. D. Gammell. "AIT* and EIT*: Asymmetric Bidirectional Sampling-Based Path Planning". In: *IJRR* (2022).

[7] A. Biere, M. Heule, and H. van Maaren. "Handbook of Satisfiability". IOS Press, 2009.

[8] W. Thomason and R. A. Knepper. "A Unified Sampling-Based Approach to Integrated Task and Motion Planning". In: ISRR. 2019.

[9] Z. Kingston, A. M. Wells, M. Moll, and L. E. Kavraki. "Informing Multi-Modal Planning with Synergistic Discrete Leads". In: ICRA. 2020.

[10] K. Hauser and J.-C. Latombe. "Multi-Modal Motion Planning in Non-expansive Spaces". In: *IJRR* 29.7 (2010).

[11] S. Cambon, R. Alami, and F. Gravot. "A Hybrid Approach to Intricate Motion, Manipulation and Task Planning". In: *IJRR* 28.1 (2009).

[12] R. Alami, T. Siméon, and J.-P. Laumond. "A Geometrical Approach to Planning Manipulation Tasks". In: ISRR. 1989.

[13] K. Hauser and V. Ng-Thow-Hing. "Randomized Multi-Modal Motion Planning for a Humanoid Robot Manipulation Task". In: *IJRR* 30.6 (2011).

[14] J. Barry, K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez. "Manipulation with Multiple Action Types". In: ISER. Vol. 88. 2013.

[15] E. Plaku, L. E. Kavraki, and M. Y. Vardi. "Motion Planning With Dynamics by a Synergistic Combination of Layers of Planning". In: *IEEE T-RO* 26.3 (2010).

[16] P. S. Schmitt, W. Neubauer, W. Feiten, K. M. Wurm, G. V. Wichert, and W. Burgard. "Optimal, Sampling-Based Manipulation Planning". In: ICRA. 2017.

[17] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. "PDDLStream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning". In: ICAPS. 2020.

[18] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. "Batch Informed Trees (BIT*): Sampling-based Optimal Planning via the Heuristically Guided Search of Implicit Random Geometric Graphs". In: ICRA. 2015.

[19] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. "Combined Task and Motion Planning through an Extensible Planner-Independent Interface Layer". In: ICRA. 2014.

[20] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. "FFRob: Leveraging Symbolic Planning for Efficient Task and Motion Planning". In: *IJRR* (2017).

[21] T. Ren, G. Chalvatzaki, and J. Peters. *Extended Task and Motion Planning of Long-horizon Robot Manipulation*. 2021.

[22] H. A. Kautz and B. Selman. "Planning as Satisfiability". In: ECAI. Vol. 92. 1992.

[23] L. de Moura and N. Bjørner. "Z3: An Efficient SMT Solver". In: *TACAS*. LNCS. Springer, 2008.

[24] M. Cashmore, M. Fox, D. Long, and D. Magazzeni. "A Compilation of the Full PDDL+ Language into SMT". In: ICAPS. AAAI Press, 2016.

[25] T. Migimatsu and J. Bohg. "Object-Centric Task and Motion Planning in Dynamic Environments". In: *IEEE RA-L* (2019).

[26] S. M. LaValle. "Planning Algorithms". Cambridge University Press, 2006.

[27] M. Ghallab, D. Nau, and P. Traverso. "Automated Planning and Acting". Cambridge University Press, 2014.

[28] S. Karaman and E. Frazzoli. "Sampling-Based Algorithms for Optimal Motion Planning". In: *IJRR* 30.7 (2011).

[29] J. Nocedal and S. J. Wright. "Numerical Optimization". 2nd ed. Springer Series in Operations Research. Springer, 2006. 664 pp.

[30] S. Osher and R. Fedkiw. "Constructing Signed Distance Functions". In: *Level Set Methods and Dynamic Implicit Surfaces*. Applied Mathematical Sciences. Springer, 2003.

[31] W. Thomason and H. Kress-Gazit. *Counterexample-Guided Repair for Symbolic-Geometric Action Abstractions*. 2021.

[32] H. Kautz, D. McAllester, and B. Selman. "Encoding Plans in Propositional Logic". In: KR. 1996.

[33] J. Hoffmann and B. Nebel. "The FF Planning System: Fast Plan Generation through Heuristic Search". In: *JAIR* 14 (2001).

[34] W. Vega-Brown and N. Roy. "Task and Motion Planning Is PSPACE-Complete". In: AAAI. 2020.

[35] I. A. Şucan, M. Moll, and L. E. Kavraki. "The Open Motion Planning Library". In: *IEEE R Robot Autom. Mag.* 19.4 (2012).

[36] E. Coumans et al. *Bullet Physics Library*. Version 3.21. 2021.

[37] A. M. Leal. *Autodiff, a Modern, Fast and Expressive C++ Library for Automatic Differentiation*. 2018.

[38] S. G. Johnson. *The NLOpt nonlinear optimization package*.