Semi-Lazy Rearrangement Solver for Accelerating Task Planning Using SPITE Dynamic Roadmaps

Marta Markowicz[1], Stav Ashur[1], James Motes[1] and Nancy M. Amato[1]

***Abstract*— In construction, we require collision-free planning to work in complex environments full of materials and debris. We introduce a rearrangement planning approach that augments a task planner leveraging lazy motion validation with a dynamic roadmap that quickly approximates the feasibility of motions after object rearrangement. This approach enables the task planner to produce candidate plans that are more likely to yield valid motion plans without significant overhead, thereby reducing the number of iterations between task planning and motion validation layers and decreasing the overall planning time.**

I. Introduction

Rearrangement planning is a critical capability of robots operating in construction environments, enabling safe and efficient operations in cluttered workspaces. Common tasks may include structure assembly of prefabricated components, materials transport, and tool retrieval. In more constrained scenarios, rearrangement of other objects may be necessary to enable an item's retrieval.

Formally, in the robot rearrangement planning problem, we are given a robot r, an environment E containing n static objects $O = \{o_1, ..., o_n\}$ with starting positions $\{s_1, ..., s_n\}$, and n sets of target positions $\{T_1, ..., T_n\}$. Our goal is to find a sequence of valid operations of r which will result in new object positions $\{t_1, ..., t_n\}$ such that $t_i \in T_i$ for $i \in \{1, ..., n\}$.

Moving construction materials (e.g., steel beams, prefab panels) to temporary positions during assembly often creates spatial conflicts in tightly coordinated worksites, restricting future operations. For instance, displacing rebar stacks might block crane access for subsequent concrete pours. The complexity of motion planning, inherent to rearrangement planning, is thus compounded by the combinatorial challenge of finding a valid sequence of operations leading to the desired result, which may require a number of steps exponential in n (e.g. the Towers of Hanoi).

Due to the unintuitive nature of the configuration space (C-space) of robots with a high number of degrees of freedom (DOF), deciding which operations will be rendered (temporarily) infeasible after a single object has been moved, requires an expensive computation of many motion planning problems. The exponential size of the combinatorial search space means that computing the feasibility of every possible operation conditioned on moving an object to a new location is impractical. Simpler heuristics are thus required for efficiently pruning the state search tree. The authors in [1] present a lazy rearrangement solver (LRS) which checks only the grasp positions for collisions with object start and goal positions to generate *reachability constraints* used to prune a search tree over object positions. Once a sequence of object positions is found, robot motions for transitioning between object positions are computed to validate the proposed sequence of object positions. Failed motion plans lead to further pruning of the search tree and additional search iterations.

In this short paper, we propose the semi-Lazy Rearrangement Solver (sLRS) which augments the LRS approach with a heuristic motion validity check during search time by leveraging the SPITE method [2], originally designed for motion planning in modified environments. SPITE takes a roadmap graph as input, generates simple geometries approximating the swept volume associated with each roadmap edge, and stores them in AABB-trees. When an objects position is updated, the AABB-trees are used for fast intersection checks, and roadmap edges are labeled as valid or of unknown validity. SPITE naturally allows for a quick update of a roadmap's approximate validity with respect to a new set of object positions. This updated roadmap can provide a quick estimate of the likelihood that there exist a valid path for moving an object with respect to the current position of other objects allowing for the search to be biased towards actions with high motion validity probability.

The preemptive approximate motion validity check and the search bias it generates reduces the total number of motion planning queries in problems for which the path approaching grasp configurations are likely to be in conflict with potential object positions. For an invalid path, the LRS would only discover the conflict after generating a rearrangement plan and validating all of the motions prior to the invalid one. We include a proof-of-concept experiment demonstrating the potential of this approach in Section III.

II. Method

In this section, we provide an overview of the augmentation of LRS with SPITE to create the semi-Lazy Rearrangement Solver. For details on the underlying solver or on SPITE, please see [1] or [2] respectively.

A. Pre-processing

Given a set of object start and goal positions (and a set of potential object positions for temporary placement), we generate a set of grasp configurations to pick/place objects at these positions. These are used to generate the *reachability*

[1]Marta Markowicz, Stav Ashur, James Motes, and Nancy M. Amato are with the Siebel School of Computing and Data Science at the University of Illinois Urbana-Champaign, 201 N Goodwin Ave, Urbana, IL 61801, United States of America, {martasm2, stava2, jmotes2, namato}@illinois.edu

constraints from LRS [1] and prime the roadmap used in SPITE [2].

We then use PRM [3] to generate a roadmap (ignoring all movable objects) which attempts to connect the grasp configurations in a single connected component (we assume that this is successful). Next, the SPITE algorithm creates *outer-approximations* of swept volumes of every edge in the input roadmap and every movable object in the environment, where an outer-approximation of a body B is a body B' that fully contains it. These approximations are stored in an AABB-tree to allow for fast intersection checks. Given a new set of object positions, in our case, the result of a potential rearrangement step changing the positions of some objects, we can quickly estimate changes in edge validities by using the AABB-tree. This allows for fast, semi-lazy collision checks.

B. Lazy Rearrangement Solver Query

The LRS method considers a search space of object placements from a set of candidate positions. The edges of the search spaces represent the transition of a single object to its goal position. The method uses the precomputed reachability constraints to prune the search space neighbors of a *state*, a set of object configurations, by only considering transitions which have a valid grasp position at both the object's current position and its goal position. This removes neighbors which would otherwise be found invalid only after full motion validation.

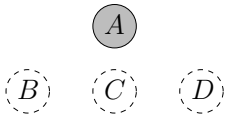
LRS creates a search tree over object placements rooted at the starting configuration of the objects (Figure 1). Edges in this tree represent the transition of a single object (usually to its goal position). The method uses precomputed reachability constraints to prune edges between states s and s' created by moving an object o, if there exists no valid grasp position for o in s' due to the rest of the object positions in the configuration represented by that state. This removes neighbors which would otherwise be found invalid only after a full motion validation of the object movement.

C. Semi-Lazy Motion Validation

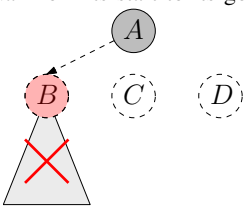
In the original LRS method [1], no motion validation is done during search time, but rather, the solver produces a candidate plan of object movement actions which transitions the system from the start configuration to the goal configuration of object positions. If an edge fails a motion validation, that is, the pick/place task it describes is found to be infeasible, it is pruned from the search tree, and the search resumes, looking for an alternative sequence of object movements.

LRS significantly reduces the total amount of motion planning required to successfully find a valid plan, but relies on the reachability constraints, i.e., only grasping configurations, to decide whether an edge between states is likely to be valid.

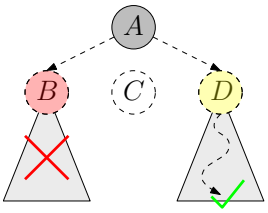
In this work, we further inform the tree search of motion viability, by performing semi-lazy motion validation using the SPITE dynamic roadmap. The dynamic roadmap is initialized by performing heuristic collision checking against



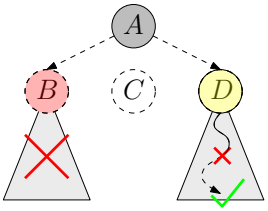
(a) Initial Expansion: The search begins by considering moving each object individual from its start to its goal position.



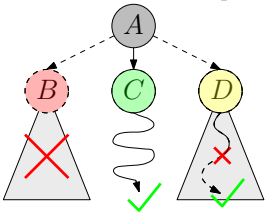
(b) Reachability Prune: The movement of any object for which its current position or goal position violates a reachability constraint is pruned. No further expansion is considered from that object configuration as indicated by B and its subtree which is never explored.



(c) Candidate Plan: The search continues until a set of object movements transitions the system from the start configuration to the goal configuration as shown in D and its abstracted subtree. sLRS performs semi-lazy motion validation during this expansion, so candidate plans are more likely to have valid motions than the original LRS approach.

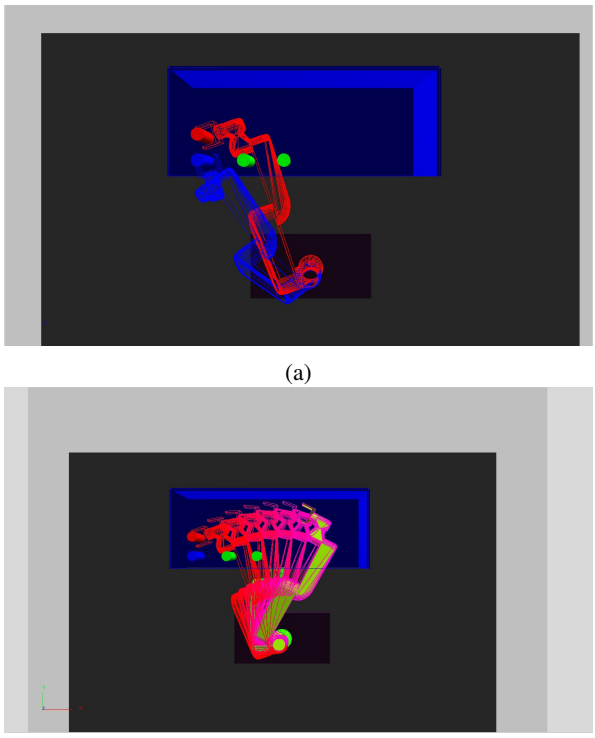


(d) Motion Prune: Each proposed object movement in the candidate plan must be fully validated by a motion planner. Any movement found to not have a valid robot motion is pruned (along with its subtree) as shown in D and its candidate plan in the subtree.



(e) Final Solution: The search continues until a candidate plan is found and fully validated in C and its candidate plan.

Fig. 1: This figure depicts the steps of (s)LRS (a-e). Each of the four states marked A, B, C, and D represent a configuration of all of the objects. Extended cones represent abstracted subtrees. Red Xs represent pruned subtrees. Squiggled arrows represent paths through the subtree. Check marks represent found candidate plans.



(a)

(b)

Fig. 2: A scenario showcasing the intuitive difference between LRS and sLRS. The figure shows a top view of shelf, with its top face made transparent for clarity, that contains three cylindrical objects, and a 6 DOF manipulator with a simple endeffector. In Figure 2a start (blue) and target (red) positions of an object are shown alongside valid grasp positions of the robot. In Figure 2b a grasping motion is shown using intermediate configurations. Such an edge in the state graph may be pruned using the motion validation in sLRS but explored when only using reachability pruning in LRS.

all environment obstacles, and all roadmap edges are labeled either valid or unknown. When estimating the validity of a transition between states s and s', the moved object o is passed into the SPITE update function, which updates the roadmap edge validities to account for its suggested position. We then query this roadmap for a path between o's positions in s and s', and compute a likelihood of motion feasibility from the ratio of known valid edges to total edges in the path. We then use the inverse of this ratio as a weight function in our tree search, biasing the search towards sequences of object movements with higher likelihoods of having viable motions. See Figure 2 for an illustration.

Since the candidate path generated during state search space pruning can include both valid and unknown edges, we must still perform full collision checks on the unknown edges to determine their validity (just as LRS performs full motion validity at this stage). If we find invalid edges in the candidate path, we must continue checking potential paths in the roadmap with valid and unknown edges until we are

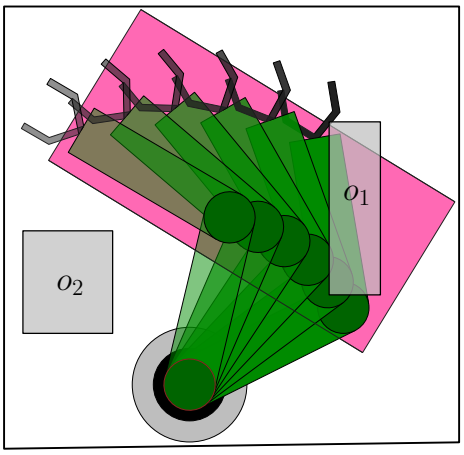


Fig. 3: A manipulator r with 2DOFs in a 2D environment E with obstacles o_1 and o_2. The outer-approximation of one of the links of r is an OBB (pink). The obstacles have outer-approximations in the form of AABB (gray). An edge e of the roadmap is shown as a sequence of intermediates. o_1 renders the edge of in the c-space of (r, E) "unknown", while o_2 has no effect on it.

able to confirm a fully valid path. Similarly to LRS, if no such path is found, the neighbor is pruned from the tree, and the tree search resumes.

By using SPITE for semi-lazy motion validation we are able to efficiently bias the search space by estimating the validity of transitions resulting in fewer edges between states later found to describe invalid motions, and thus fewer total motion plans, fewer iterations of object movement sequencing and motion validation, and faster total runtimes as illustrate in the next section.

D. Non-monotonic Problems Instances

Non-monotonic problems are those in which an object must be moved to an intermediate position before being moved to its goal. This is common when there are reachability dependencies between object start and goal positions which necessitate the non-monotonic actions. LRS addresses this by considering random perturbations of object positions, validating the motions to achieve these perturbations. In this iteration of the work, we maintain the original LRS approach. We discuss how SPITE may be leveraged for more intelligent perturbations in Section IV.

III. Evaluation

To demonstrate the enhanced performance of sLRS we ran preliminary experiments on a rearrangement problem using a 6 DOF manipulator. The environment consists of a shelf with 3 cylindrical objects, with a 3×6 grid of potential object placements on the shelf, as shown in Figure 4.

We ran 50 trials with randomly generated start and goal positions for the objects selected from the set of positions highlighted in Figure 4. The two back corner positions were not considered as the motion difficulty associated with them obfuscated the impact of the proposed contribution.

	LRS		**sLRS**	
Metric	Median	Mean	Median	Mean
Reachability Prunes	5	18.42	4	18.36
Motion Prunes	0.5	2.17	0	1.85
Perturbations	0.5	3.17	0	3.17
Time Taken (s)	8.75	17.20	8.68	16.12

TABLE I: Statistics for sLRS vs. LRS

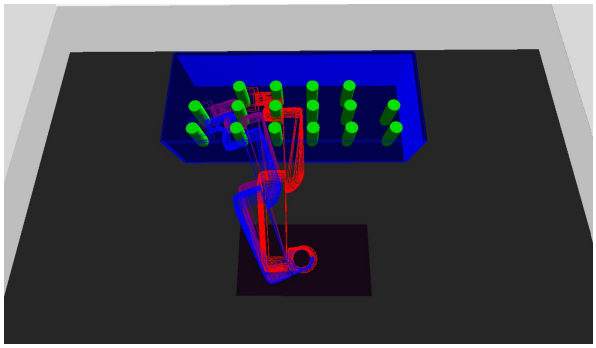


Fig. 4: An illustration of the environment described in Section III. Several configurations are shown where the robot is grasping one of the cylindrical objects. Note that the top face of the shelf is transparent for the sake of clarity.

In this simple, proof of concept scenario, the amount of pruning due to motion validation is not high for either method, as there are only 3 objects randomly configured among 16 positions. As such, there are many problem instances in which the position of the objects do not impede the movement of other objects by the robot.

Proportionally, we see a 15% decrease in the average number of motion validations, which leads to a small reduction in average planning time. We expect to see both of these percentages increase as the density of the objects in the scene increases.

***Implementation details**:* Our code was developed using the Parasol Planning Library (PPL [4]). All experiments were run on a laptop computer with an Intel core i7-11800H at 2.30GHz, with 32 GB of RAM.

IV. Conclusion and Future Work

In this extended abstract, we present the semi-Lazy Rearrangement Solver, an extension of the Lazy Rearrangement Solver [1] that exploits the dynamic roadmap offered by SPITE [2] to bias the search towards object movements that are likely to have valid motions. We provide a proof-of-concept rearrangement scenario in Section III with preliminary results demonstrating the expected behavior for the limited problem size.

In future work, we will leverage the dynamic roadmap's ability to estimate motion feasibility after object movement to not only bias the tree search, but to select object perturbation candidates based on the increased motion viability after their removal and choose target positions which minimize the likelihood of requiring further perturbations.

References

[1] R. Wang, K. Gao, J. Yu, and K. Bekris, "Lazy rearrangement planning in confined spaces," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 32, 2022, pp. 385–393.

[2] S. Ashur, M. Lusardi, M. Markowicz, J. Motes, M. Morales, S. Har-Peled, and N. M. Amato, "SPITE: Simple polyhedral intersection techniques for modified environments," in *Algorithmic Foundations of Robotics XVI (WAFR)*, 2024.

[3] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation (TRA)*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[4] P. Lab, "Parasol planning library (PPL)," https://github.com/parasollab/open-ppl, 2025.