2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

On the Performance of Selective Adaptation in State Lattices for Mobile Robot Motion Planning in Cluttered Environments

Michael E. Napoli, Harel Biggie, and Thomas M. Howard

***Abstract*— Autonomous mobile robots require motion planning algorithms that match limitations of on-board computing resources to safely navigate complex environments. In situations where near-optimality is preferential to runtime performance, search spaces that optimize their local connectivity to improve the global optimality of generated solutions are desirable. However, not all nodes in the search space benefit equally from optimization which can result in an inefficient use of computational resources and unnecessary increase in runtime. To address this limitation, we propose an approach called the Selectively Adaptive State Lattice which uses a heuristic based on the local environment to selectively perform optimization and obtain a balance between runtime performance and relative optimality. We present a statistical evaluation of local connectivity optimization and global search with the State Lattice, Adaptive State Lattice, and Selectively Adaptive State Lattice algorithms in randomly generated obstacle fields. We further highlight the performance of each method on a Clearpath Robotics TurtleBot2 in a qualitative physical experiment.**

Fig. 1: A Clearpath Robotics TurtleBot2 mobile robot posed with the problem of navigating a cluttered obstacle field composed of traffic cones with limited computing resources.

I. Introduction

The capabilities of autonomous mobile robots have seen dramatic improvements due to advances in sensing technologies, miniaturization and acceleration of computing resources, and novel algorithms and models for intelligent navigation. These advances have enabled autonomous vehicles to safely navigate in dynamic urban and off-road environments. In certain applications (such as planetary exploration or nuclear inspection) it is crucial to minimize risk, energy consumption, and vehicle wear during planning as human intervention may be limited or unavailable entirely.

A central challenge of robot navigation in this domain is finding optimal maneuvers with limited computing resources (Figure 1). Feasibility of planned maneuvers is important to guarantee path traversability, as is efficiency of search towards a solution. Graph or sampling based approaches are commonly used to generate near-optimal plans in environments where many homotopically distinct solutions exist. The choice of search algorithm and state-space representation is typically a compromise between efficiency, optimality, and feasibility of generated plans.

In applications where computational resources may be limited or shared by scientific instruments, algorithms that minimize their memory footprint may prove advantageous. The challenge for such approaches is to optimize the distribution of a limited number of nodes and edges in a graph to improve the global optimality of generated solutions. Algorithms which adapt search spaces to their local environment also have applications in domains where robots repeatedly travel similar routes in cluttered, unstructured environments, such as logistics for mines, farms, worksites, and warehouses.

This paper investigates the optimality and runtime performance of a certain class of recombinant motion planning search spaces called State Lattices (SLs) [1]. SLs pre-encode the dynamics of robot maneuvers into a trajectory library or control set that is repeatedly expanded to generate a graph which can be searched with heuristic search. Adaptive State Lattices (ASLs) adapt nodes during this expansion based on the aggregate cost of local edges to improve the performance of motion planning in such graphs. Experiments have demonstrated that in sufficiently complex environments, the ASL generated better solutions faster and with less memory that the SL counterpart at finer state resolutions [2]. A limitation of the ASL is that adaptation is a computationally expensive operation and is uniformly applied to the search space including regions with minimal cost variations and many local optima. We hypothesize that a heuristic for selective adaptation of nodes that considers the cost field of the local environment will improve the optimality of generated solutions while reducing the runtime requirements without increasing the memory footprint.

We present five main contributions in this paper. First, we investigate the relative performance of SL and ASL in obstacle fields generated by Poisson forest distributions and examine statistics on the relative improvement of nodes through optimization. Second, we propose the Selectively Adaptive State Lattice (SASL), which selectively enables or disables local connectivity optimization using a heuristic

M.E. Napoli, H. Biggie, and T.M. Howard are with the Department of Electrical and Comptuer Engineering, University of Rochester, Rochester, NY, 14627, mnapoli@ece.rochester.edu

based on the local environment at each node during search. Third, we demonstrate the procedure for selecting heuristic thresholds based on Normalized Mean Cell Cost (NMCC) from SL and ASL experiments. Additionally, we evaluate the relative optimality, node expansions, and runtime of the SASL with respect to the SL and ASL. Lastly, we demonstrate the qualitative performance of generated solutions in simulated obstacle fields and on a Clearpath Robotics TurtleBot2 mobile robot platform.

II. Related Works

Advances in mobile robot motion planning have resulted in many classifications of algorithms that exploit a variety of methods for sampling the state-action space. Rapidly-exploring Random Trees (RRTs) are an example of probabilistic algorithms that repeatedly select random samples in the robot's configuration space [3]. Samples are drawn at each iteration and connected to a growing tree structure. During this process, the robot's differential and environmental constraints are satisfied by encoding the vehicle's dynamics and applying collision checking. Once a specified goal region is connected to the tree, a path or sequence of actions is generated from the start node. The work in [4] proved that in certain conditions, the RRT converges to a suboptimal solution with probability 1. Additionally, it introduced the RRT* algorithm, which almost surely converges to the optimum as the number of samples approach infinity.

Probabilistic Roadmaps (PRMs) are another family of random sampling algorithms which have been applied to mobile robot motion planning [5]. This algorithm describes an iterative procedure which grows an undirected graph where the nodes represent sampled free configuration space and the edges are collision free holonomic connections between nodes. The graph is grown through a learning phase which randomly selects samples and searches for collision free connections to a subset of existing nodes using a fast local planner. During this phase, a random bounce is incorporated to expand nodes which may be in difficult regions of the environment. After the learning phase, a query phase is performed which tries to connect the goal node to the start. As with RRTs, PRMs can generate solutions in cluttered environments after a sufficient number of samples.

While probabilistic sampling algorithms have found much success in the motion planning community, they often suffer from four limitations. The first is that probabilistic sampling convergence can often be inefficient as the algorithm explores uniformly across the search space, resulting in extraneous memory consumption due to storage of unused samples. Secondly, the obtained solutions are often highly suboptimal due to the stochastic sampling process and require post search smoothing. A third limitation is that these approaches often improve their optimality by increasing the number of samples, which is a disadvantage in systems with limited memory resources. Lastly, feasible edges must be generated online, however modern methods for quickly generating edges in high-dimensional state-spaces are often suboptimal. To address these limitations, the SL was introduced which utilized a regular sampling of the state-action space. Vehicle dynamics are pre-encoded in the trajectory library or control set, resulting in dynamically feasible solutions at no increased runtime expense [1]. The control set consists of sampled inputs such as body frame linear velocity and curvature and are defined using parametric functions of time or distance, such as polynomial spirals [6]. Optimal strategies for control set generation demonstrate the effectiveness of SL for kinodynamic robot motion planning in cluttered environments [7]. The planning search space often consists of planar position (x, y), heading (θ), and curvature (κ) resulting in a state vector $\mathbf{x}$. The state can be extended to incorporate fewer or additional dimensions such as velocity or time [1], [8]. SLs have been shown to be effective for mobile robot navigation in urban environments [9].

SLs are well suited to motion planning for autonomous systems with memory resource limitations since they are capable of finding globally optimal solutions with respect to a predetermined sampling density. The optimality of obtained solutions approach that of the continuum at finer sampling with the concession of efficiency [1]. The resolution is determined *apriori* and therefore, chosen irrespective of the environment. As a result, cases may occur where a solution exists in the continuum but not within the chosen density. Approaches to mitigate this include finer sampling or use of a hierarchical structure of state lattices with graduated fidelity [10]. These solutions are still limited by the fundamental issue that the SL is only resolution complete, meaning all solutions are considered but only within the chosen resolution. An alternative approach is to apply local connectivity optimization to sampled nodes. Many state of the art motion planning algorithms apply local optimization to improve the overall optimality of obtained solutions. Examples of this include Regionally Accelerated Batch Informed Trees* (RABIT*), which applies local optimization to accelerate a BIT* search [11], and Covariant Hamiltonian Optimization for Motion Planning (CHOMP), which refines trajectories and quickly converges to locally optimal solutions [12]. To overcome the SL limitations, local connectivity optimization is applied to all expanded nodes in the ASL [2]. However, the ASL's improvement in optimality comes at a significant computational cost relative to the SL at equal state-action space sampling resolutions.

The central issue of relaxing the regular sampling must be overcome if SLs are to be used effectively in complex environments. The proposed SASL approach improves the runtime performance of the ASL by selectively optimizing the search space with respect a local environment based heuristic. The rest of this paper is organized as follows. In Section III we review the ASL algorithm, as it is the basis for the SASL, and perform an evaluation of local connectivity optimization in randomly generated obstacle fields. Section IV introduces the SASL and discusses the chosen heuristic (NMCC). Experimental results for the SASL are described in Section V, which illustrate its performance against the SL and ASL baselines. We conclude with a discussion of contributions and directions for future work in Section VI.

III. Local Connectivity Optimization in State Lattices

A. Adaptive State Lattice

The ASL minimizes the aggregate control set cost using gradient descent to adjust each node's state vector with respect to the local environment [2]. While the solutions obtained at equivalent resolutions were shown to often be lower in cost for sufficiently complex environments than the baseline SL, computational run times were higher when comparing the two approaches at the same resolution. The edge cost is defined as the sum of the path length (s_f) and the integrated path cost ($\mathcal{L}$) over the cost map as shown in equation 1. The parent node's state vector is represented as $\mathbf{x}_p$ and the n^{th} child's is $\mathbf{x}_n$. In practice, since the cost map is a discrete sampled representation of the environment, the integral in equation 1 is computed numerically.

$$J(\mathbf{x}_p, \mathbf{x}_n) = s_{f,n} + \int_{s_{0,n}}^{s_{f,n}} \mathcal{L}(\mathbf{x}_p, \mathbf{x}_n, s) ds \tag{1}$$

The objective function minimized during optimization is the aggregate edge cost of N child nodes as shown in equation 2.

$$J_{agg}(\mathbf{x}_p) = \sum_{n=1}^{N} J(\mathbf{x}_p, \mathbf{x}_n) \tag{2}$$

The optimization of the aggregate edge cost is performed using gradient based methods. The gradient is estimated numerically using forward differencing as shown in equation 3.

$$\begin{aligned} \Delta_j &= \begin{cases} \Delta, & i = j \\ 0, & \text{otherwise} \end{cases} \\ \frac{\delta\left(J(\mathbf{x}_p)\right)}{\delta(x_j)} &\approx \frac{J\left(\mathbf{x_{p,j}} + \Delta_j\right) - J\left(\mathbf{x}_p\right)}{\Delta} \end{aligned} \tag{3}$$

In the original ASL formulation, the gradient is then used to update the parent node's state vector by stepping in the direction of greatest descent using a step length (α) as shown in equation 4. The implementation in this work additionally uses a line search before the update step to assist in convergence. The full optimization procedure is outlined in algorithm 1.

$$J_{k+1}(\mathbf{x}_p) = J_k(\mathbf{x}_p)_k - \alpha \nabla J_k(\mathbf{x}_p) \tag{4}$$

The returned parent node state vector ($\mathbf{x}_p$) is optimized with respect to the environment and thus the resulting edges tend to traverse safer regions. Figure 2 highlights edge adaptation after optimization and visualizes this effect. The expense of the local connectivity optimization is due in part to the number of trajectory evaluations that are required. Specifically, for every computation of an element in the gradient, a trajectory must be generated to each of the parent node's N children. If there are b number of variables in the parent node's state vector $\mathbf{x}_p$, this requires $N(b+1)$ trajectory generation calls per gradient descent iteration.

Algorithm 1: Local connectivity optimization

Input : Parent node $\mathbf{x}_p$, child nodes $\mathbf{x}_{p,1:N}$, step length α, line search parameter β, finite difference step Δ
Output: Parent node (adapted) $\mathbf{x}_p$

Adapt
  $k \leftarrow 1$
  $J_k \leftarrow$ ComputeCost$(\mathbf{x}_p, \mathbf{x}_{p,1:N})$
  while $k \leq K$ **do**
    $\nabla J_k \leftarrow$ EstimateGradient$(\mathbf{x}_p, \mathbf{x}_{p,1:N}, \Delta, J_k)$
    $\mathbf{x}_p, J_{k+1} \leftarrow$ LineSearch$(\mathbf{x}_p, \mathbf{x}_{p,1:N}, J_k, \nabla J_k, \alpha, \beta)$
    if *converged* **then**
      break
    end
    $k \leftarrow k + 1$
  end
  return $\mathbf{x}_p$
end

ComputeCost $(\mathbf{x}_p, \mathbf{x}_{p,1:N})$
  $J \leftarrow 0$
  for $n \leq N$ **do**
    $J \leftarrow J + \mathbf{s_{f,n}} + \int_{s_{0,n}}^{s_{f,n}} \mathcal{L}(\mathbf{x}_p, \mathbf{x}_n, s) ds$
  end
  return J
end

EstimateGradient $(\mathbf{x}_p, \mathbf{x}_{p,1:N}, \Delta)$
  $\nabla J \leftarrow \mathbf{0}$
  foreach *element* j *in* $\mathbf{x}_p$ **do**
    $x_{p,j} \leftarrow x_{p,j} + \Delta$
    $\nabla J \leftarrow$ ComputeCost$(\mathbf{x}_p, \mathbf{x}_{p,1:N}) - J$
    $x_{p,j} \leftarrow x_{p,j} - \Delta$
  end
  $\nabla J \leftarrow \frac{\nabla J}{\Delta}$
  return ∇J
end

LineSearch $(\mathbf{x}_p, \mathbf{x}_{p,1:N}, J_k, \nabla J_k, \alpha, \beta)$
  $\hat{\alpha} \leftarrow \alpha$
  $\hat{\mathbf{x}}_p \leftarrow \mathbf{x}_p - \hat{\alpha} \nabla J$
  $J_{k+1} \leftarrow$ ComputeCost$(\hat{\mathbf{x}}_p, \mathbf{x}_{p,1:N})$
  while $J_{k+1} \geq J_k$ **do**
    $\hat{\alpha} \leftarrow \beta \hat{\alpha}$
    $\hat{\mathbf{x_p}} \leftarrow \mathbf{x}_p - \hat{\alpha} \nabla J$
    $J_{k+1} \leftarrow$ ComputeCost$(\hat{\mathbf{x}}_p, \mathbf{x}_{p,1:N})$
  end
  return $\hat{\mathbf{x}}_p, J_{k+1}$
end

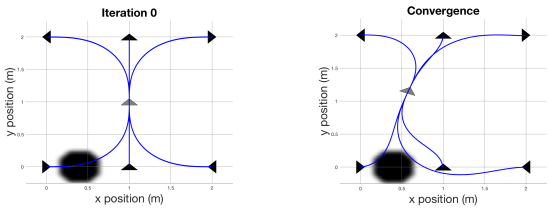
To minimize the amount of computations these trajectory generation subroutines need to perform, a coarse look up table is provided to initialize them.

The optimization process effectively improves the quality of the sample before placing a node on the open list [2]. However, to incorporate these modified costs, the heuristic graph search algorithm must be revised. Equivalently to the regular state lattice, the expansion step is performed on the top node in the open list [1]. Each of the next level child nodes (referred to as the L1 children) are generated. Every L1 child is expanded (generating L2 children) and their state vectors are individually optimized using the procedure in algorithm 1 [2]. If an L1 node has previously been adapted, it is not adapted again and the optimization procedure is bypassed. After the optimization procedure, all new L1 nodes are put on the sorted open list with the new edge costs. The modified heuristic graph search is outlined in algorithm 2.

Algorithm 2: Adaptive state lattice heuristic search

Input : Start node $\mathbf{x}_s$, goal node $\mathbf{x}_g$, step length α, line search parameter β, finite difference step Δ
Output: Trajectory of nodes $\mathbf{x}(t)$

Main
  OPEN $\leftarrow \mathbf{x}_s$
  CLOSED $\leftarrow \emptyset$
  while *OPEN* $\neq \emptyset$ **do**
    $\mathbf{x}_{next} \leftarrow$ get top from OPEN
    if $\mathbf{x}_{next} == \mathbf{x}_g$ **then**
      return $\mathbf{x}_g$
    end
    $\mathbf{X}_{L1} \leftarrow$ EXPAND($\mathbf{x}_{next}$)
    foreach $\mathbf{x}_{L1}$ *in* $\mathbf{x}_{next}$ **do**
      if $\mathbf{x}_{L1}$ *is not in OPEN* **then**
        $\mathbf{X}_{L2} \leftarrow$ EXPAND($\mathbf{x}_{L1}$)
        adapt($\mathbf{x}_{L1}, \mathbf{X}_{L2}, \alpha, \beta, \Delta$)
        OPEN $\leftarrow \mathbf{x}_{L1}$
      end
    end
    closed $\leftarrow \mathbf{x}_{next}$
    sort(OPEN)
  end
end

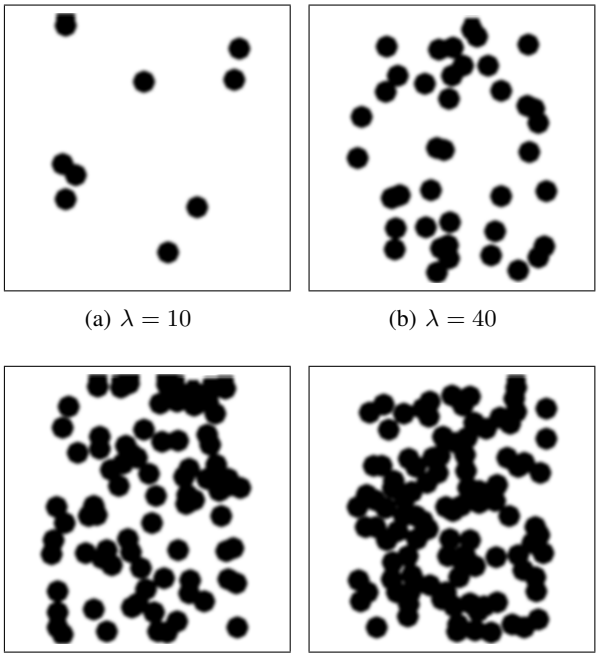


(a) Iteration 0 (b) Convergence

Fig. 2: Parent node state optimization visualization. The optimization procedure adjusts the parent node state vector until its edges are traversing fewer high cost (dark) regions.

B. Initial Evaluation

We desired to quantify the relative performance of the ASL compared to the SL. To achieve this, a Poisson forest procedure was used to generate a series of random worlds with increasing obstacle density similar in approach to the work in [13]. Each random world was 20 x 20 meters in size and contained two obstacle free regions representing the start and goal regions. The expected level of occurrences (λ) was varied from 10 to 100 and the number of constant sized disk shaped obstacles was chosen from a Poisson distribution. Each of these were placed in a non-overlapping, uniformly distributed random location in the allowable region. Post instantiation, the worlds were then C-space expanded assuming a cylindrical robot. Nodes from the search algorithms were not allowed to expand into these zones. The maps were expanded further and smoothed allowing the original inadmissible regions to remain intact and approximate a penalty for proximity to obstacles. Finally, the maps were cropped to bound the search process through the obstacle field. Example cost map image representations are shown in Figure 3.



(a) $\lambda = 10$ (b) $\lambda = 40$

(c) $\lambda = 70$ (d) $\lambda = 100$

Fig. 3: Random world cost maps for varying values of λ. The difficulty of the planning task are qualitatively visualized for a span of λ. As the expected rate of obstacle occurrence increases, the more cluttered the planning space becomes.

The configuration space in this work consists of 2D position, heading, and curvature resulting in a four dimensional search space. However, the curvature at each node in this space is constrained to be $\kappa = 0$, which effectively reduces the dimensionality of the search space to three. As such, all actions generated between nodes are forced to begin and end with zero curvature. A control set was selected

to emulate the set used in [2] which samples the search space in intervals of one meter for position and $45°$ for heading. There has been much prior work in quantifying the state lattice performance over the control set outdegree for a variety of environments [14], therefore the work presented restricts itself to the chosen control set to serve as a nominal choice of motion primitives.

The percent improvement of the objective function (equation 2) was computed every time the optimization procedure was called. This was calculated using the definition in equation 5. In this case, improvement is actually a decrease in the objective value resulting in a negative percentage when compared to the initial objective. The percentage is negated reflecting that the improvement is intuitively positive.

$$\% \text{ improvement} = \frac{J_{optimized} - J_{initial}}{J_{initial}} \times -100 \qquad (5)$$

To better understand the impact of local connectivity optimization across search queries in different obstacle densities, a sample world for every level of clutter ($0 \leq \lambda \leq 100$) was chosen and the histogram of percent improvement is shown in Figure 4. It is observed that a majority of optimizations resulted in little to no improvement in the objective function. However, these optimization calls are computationally expensive since numerical estimates of the gradient are still required. The results indicate that it could be beneficial to introduce a paradigm which selectively optimizes nodes in the search space based on a local heuristic.

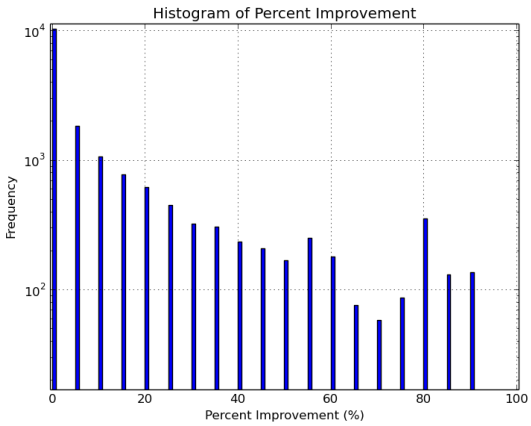


Fig. 4: Histogram of percent improvement. Our hypothesis that non-uniform application of local connectivity optimization could generate solutions with similar optimality but more efficient sequential search is supported by the significant number of nodes that yield little to no improvement.

IV. Selectively Adaptive State Lattice

A disadvantage of the ASL is application of local connectivity optimization to all nodes. Significant improvement of the objective function may not always be obtained even when the cost is high. This leads to inefficient use of computational resources which are spent optimizing regions with small gains. To mitigate this, the SASL computes a local heuristic over the cost map patch spanned by the $L1$ node expansion as illustrated in Figure 5. The result of this computation is compared against a threshold value and the adaptation is performed if the threshold criteria is met. We present a heuristic $h_{adapt}(\mathbf{x})$ based on Normalized Mean Cell Cost (NMCC) to serve as a nominal metric for selective adaptation, however alternative heuristics may be used instead. NMCC is described as the average cost of all local cells $\mathbf{m}(\mathbf{x})$, where $\mathbf{m}(\mathbf{x})$ is composed of a set of sampled positions $m_i \in \mathbf{m}(\mathbf{x})$ within the footprint of the control set for the node corresponding to state $\mathbf{x}$. Every cell cost $c(m_i)$ is normalized by the maximum representable cell cost c_{max}.

$$h_{adapt}(\mathbf{x}) = \frac{1}{|m(\mathbf{x})|} \sum_{m_i \in \mathbf{m}(\mathbf{x})} \frac{c(m_i)}{c_{max}} \qquad (6)$$

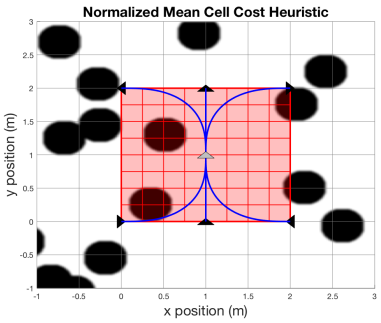


Fig. 5: Notional visualization of the heuristic which is calculated over the cells spanned by the current control set expansion. All red cells (coarsely quantized to 0.25 m in x, y for illustration purposes) are included in the calculation

The updated graph search with selective adaptation is outlined in algorithm 3. The additional selective adaptation heuristic permits optimization below a chosen threshold and begins at line 12. Leveraging the same data used to obtain Figure 4, the NMCC and the percent improvement in the objective function were recorded and the statistics are shown in Figure 6. It is observed that the optimization of regions with higher heuristic values can result in a large variation of improvement. A subset of these regions are likely very difficult to optimize with respect to and may not be worth the computational resources. Furthermore, the results indicate that regions with lower NMCC tend to provide moderate improvement. It appears that these regions are easier to optimize and that it would be advantageous to perform the adaptation. Based these results, four levels of heuristic thresholding were chosen to examine the performance of this selective process.

V. Experimental Results

A. Simulation Experiment

To comprehensively evaluate the performance of the proposed SASL in the context of ASL and SL, a statistical study was conducted and the relative optimality, number of node expansions, and runtime were recorded for

Algorithm 3: Selectively adaptive state lattice search

Input : Start node $\mathbf{x}_s$, goal node $\mathbf{x}_g$, heuristic threshold h_{thresh}, step length α, line search parameter β, finite difference step Δ
Output: Trajectory of nodes $\mathbf{x}(t)$

Main
  OPEN $\leftarrow \mathbf{x}_s$
  CLOSED $\leftarrow \emptyset$
  while *OPEN* $\neq \emptyset$ **do**
    $\mathbf{x}_{next} \leftarrow$ get top from OPEN
    if $\mathbf{x}_{next} == \mathbf{x}_g$ **then**
      return $\mathbf{x}_g$
    end
    $\mathbf{X}_{L1} \leftarrow$ EXPAND($\mathbf{x}_{next}$)
    foreach $\mathbf{x}_{L1}$ *in* $\mathbf{x}_{next}$ **do**
      if $\mathbf{x}_{L1}$ *is not in OPEN* **then**
        $h_{adapt} \leftarrow$ heuristic($\mathbf{x}_{L1}$)
        if $h_{adapt} \leq h_{thresh}$ **then**
          $\mathbf{X}_{L2} \leftarrow$ EXPAND($\mathbf{x}_{L1}$)
          adapt($\mathbf{x}_{L1}, \mathbf{X}_{L2}, \alpha, \beta, \Delta$)
        end
        OPEN $\leftarrow \mathbf{x}_{L1}$
      end
    end
    closed $\leftarrow \mathbf{x}_{next}$
    sort(OPEN)
  end
end

each of the algorithms. Four levels of heuristic threshold $\left(c_{thresh} = \begin{bmatrix}0.55 & 0.65 & 0.75 & 0.85\end{bmatrix}\right)$ for the SASL approach were chosen as described in Section IV. All three motion planning algorithms were required to plan over every combination of starting states, goals, and random worlds with varying degrees of clutter generated using the Poisson forest procedure described previously in Section III-B. The results were obtained by assessing the performance over 8100 simulations for each algorithm.

B. Simulation Results

The resulting relative optimality, number of node expansions, and runtime vs obstacle occurrence rate are shown in Figures 7, 8, and 9 respectively. The runtime results were obtained using an Intel Xeon(R) CPU E5-2620 v3 2.40GHz processor. The relative optimality (J_{rel}) is the ratio of the free space solution cost obtained by the state lattice and the cost of the solution achieved by the algorithm in the Poisson forest. While this metric does not provide the true relative optimality compared to the continuum, it does provide a bound since the free space solution cost of the SL will never be lower than the continuum optimum solution.

The results from Figure 7 show that the ASL achieves the highest relative optimality and the SL attains the lowest. As the heuristic threshold increases from 0.55 to 0.85 the SASL relative optimality approaches that of the ASL. The SASL

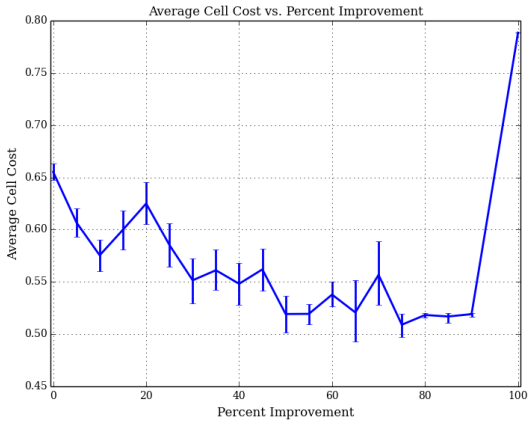


Fig. 6: NMCC vs percent improvement of aggregate control set cost with 95% confidence intervals. Higher NMCC result in a wide variance of improvement whereas the lower costs tend to provide more consistent but smaller improvement.

runtime (Figure 9) increases with the heuristic threshold, however it remains significantly lower than the ASL, even for the largest threshold value. The number of node expansions, as an indication of memory requirements (Figure 8), in the ASL and SASL (all threshold values) are consistently lower than the SL in lower clutter environments with comparable results between the two. In higher clutter regions, the SL begins to fail to find solutions due to resolution completeness. However, in some cases, the SASL and ASL are able to overcome these limitations and completely explore the space. The SASL tends to expand fewer nodes in higher clutter worlds ($\lambda \geq 90$) as a direct result of the threshold chosen. However, even with fewer node expansions, the SASL relative optimality is similar to ASL. The results indicate that the SASL is capable of providing comparable relative optimality to the ASL with reduced computational runtime and memory requirements in a Poisson forest.

C. Comparative Results

To analyze the comparative differences in solutions generated by the SL, ASL, and SASL, trajectories are visualized in cost maps where $\lambda = 40$ and $\lambda = 70$ in Figure 10. In the $\lambda = 40$ case, it is observed that the SL solution exhibits a higher cost ($J = 23.49$) than the ASL ($J = 20.04$) and SASL ($J = 20.04$), but that the SL runtime is lower ($t = 0.819\,\text{sec}$) than the ASL ($t = 11.415\,\text{sec}$) and SASL ($t = 6.910\,\text{sec}$). The SASL performed 33.8% fewer adaptations than the full ASL and was able to achieve almost identical optimality. For the $\lambda = 70$ case, we again observe that the solution for SL is higher ($J = 126.8$) than the ASL ($J = 91.75$) and SASL ($J = 112.32$) and that the runtime of the SL is lower ($t = 1.403\,\text{sec}$) than the ASL ($t = 43.093\,\text{sec}$), and SASL ($t = 10.390\,\text{sec}$). The SASL performed 80.0% fewer adaptations than the ASL, but at the expense of some optimality. However, the SASL optimality was still improved over the SL. The trend exhibited in the simulation

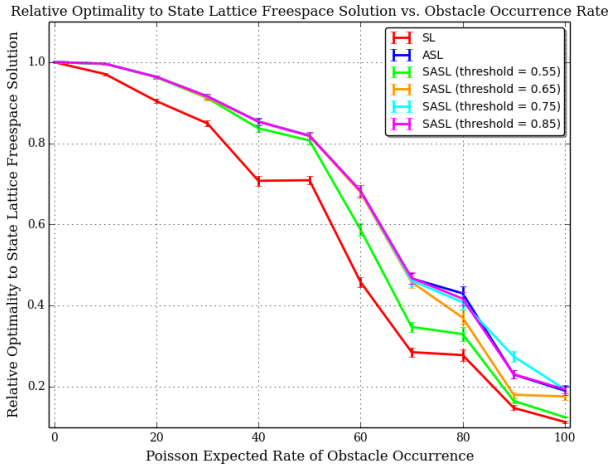


Fig. 7: Relative optimality vs Poisson expected rate of obstacle occurrence with 95% confidence intervals. The SASL maintains similar performance to the ASL for longer as the heuristic threshold is increased. The SASL performance does not appear to drop below that of the SL.

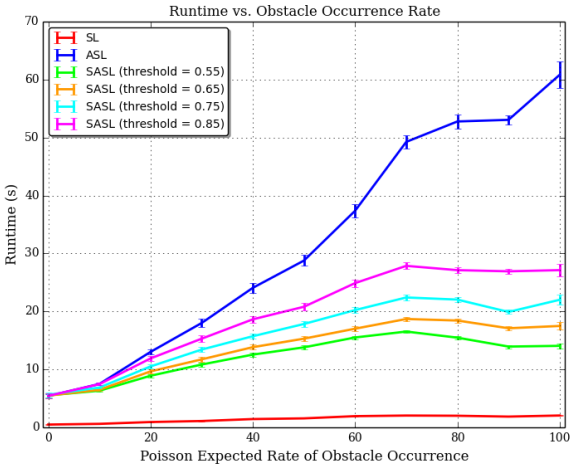


Fig. 9: Runtime vs Poisson expected rate of obstacle occurrence with 95% confidence intervals. The SASL maintains a runtime that is statistically less than or equal to the ASL. As the heuristic threshold increases, the SASL is able to maintain performance similar to the ASL for higher rates at the increase of some runtime

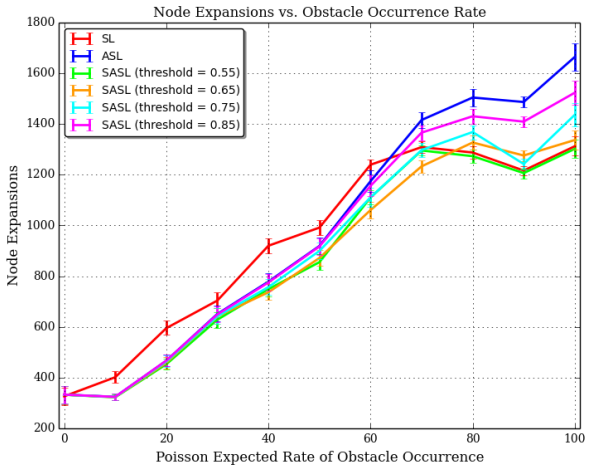


Fig. 8: Number of node expansions vs Poisson expected rate of obstacle occurrence with 95% confidence intervals. The SASL maintains performance between the ASL and the SL. The SASL and ASL expansions increase in the higher clutter obstacle fields whereas the SL fails to find solutions.

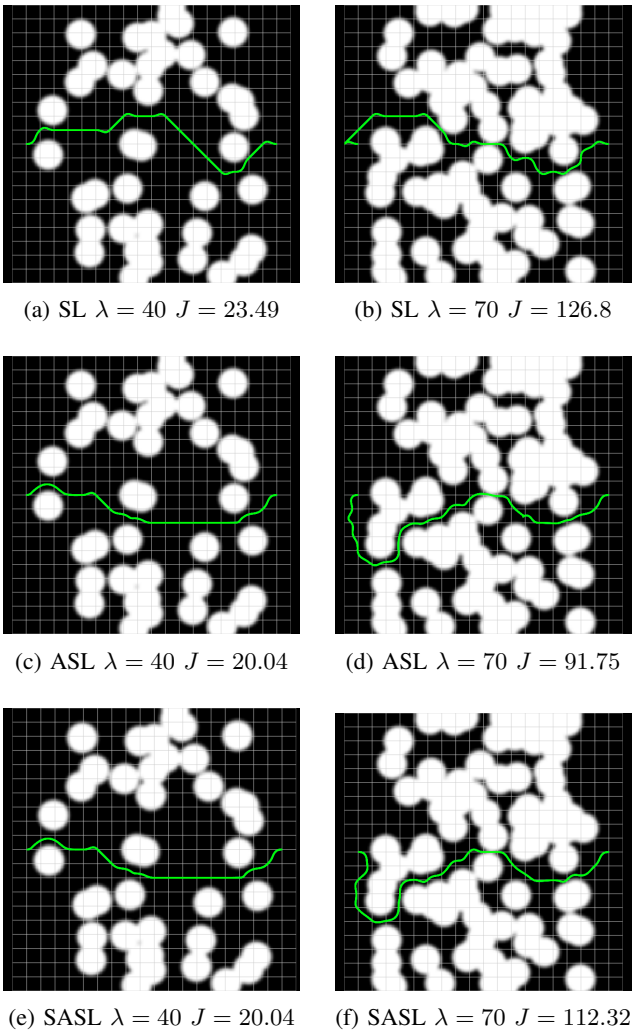
experiments is observed where the SASL displayed a degree of optimality close to the ASL but with improved runtime. Most interestingly, it is observed that the ASL and SASL solutions lie in the same homotopy class, while the SL does not. From this we can conclude that post-search relaxation of the generated trajectory from the SL would not necessarily converge to the same location as the ASL or SASL.

The comparative results in Section V-C are further corroborated in a physical experiment using a Clearpath Robotics TurtleBot2 mobile robot with the setup shown in Figure 1. Each algorithm was tasked with planning through an obstacle field of nominal clutter and the results are visualized in Figure 11. The SASL produced a path in the same homotopy class as the ASL with reduced runtime ($t = 2.12\,\text{sec}$ and $t = 2.61\,\text{sec}$ respectively) and an expected decrease in relative optimality ($J_{rel} = 0.37$ and $J_{rel} = 0.53$ for the SASL and ASL respectively). The SL produced a path in an entirely different homotopy class but faster ($t = 0.43\,\text{sec}$) with significantly reduced relative optimality ($J_{rel} = 0.22$).

VI. Conclusions and Future Work

Algorithms for improving the relative optimality of motion planning search spaces without increasing the memory footprint are important for mobile robots with limited computing resources. Such algorithms are particularly enabling for mobile robots operating in domains where proximity to the global optimum (representing a minimization of time, risk, and energy consumption) is preferable to runtime performance, which is particularly relevant for platforms in planetary exploration or nuclear inspection where rescue of the platform may be difficult or impossible. The studies and evaluation of SL, local connectivity optimization in ASL, and performance of the proposed SASL demonstrates the utility of prioritizing nodes for optimization based on thresholds identified from simulation statistics. Simulation studies demonstrate that, for particular levels of obstacle density, the SASL achieves a relative optimality near the ASL but with a sizable reduction in runtime. Qualitative select simulated and physical experiments further demonstrate and visualize this.

We are also interested in evaluating the performance of adaptive recombinant motion planning search spaces with sampling-based approaches to consider the tradeoffs in optimality, feasibility, efficiency, and memory requirements.



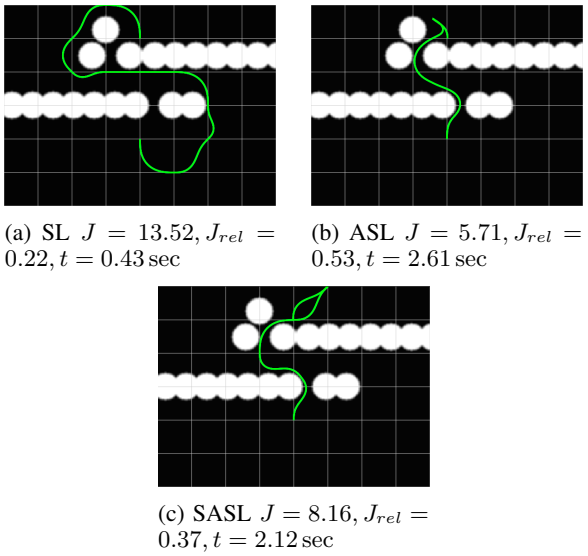
(a) SL $\lambda = 40$ $J = 23.49$ (b) SL $\lambda = 70$ $J = 126.8$

(c) ASL $\lambda = 40$ $J = 20.04$ (d) ASL $\lambda = 70$ $J = 91.75$

(e) SASL $\lambda = 40$ $J = 20.04$ (f) SASL $\lambda = 70$ $J = 112.32$

Fig. 10: Generated trajectories for SL, ASL, and SASL for $\lambda = 40$ and $\lambda = 70$. The SASL used a heuristic threshold of 0.55. The ASL and SASL generated lower cost paths by applying local connectivity optimization.

The presented studies also open up new questions on how to improve the relative performance of recombinant motion planning search spaces. The presented results utilize a simple threshold based on local cost map values. In ongoing work we, are investigating models for learning this threshold to identify candidates that will yield the most improvement towards the global optimal solution and identifying how to use information from prior search results to seed this decision on local connectivity optimization. We are further exploring algorithms for selectively modifying the density of control sets during search. Lastly, we are investigating mechanisms for improving runtime performance of cost estimates during adaptation.



(a) SL $J = 13.52, J_{rel} = 0.22, t = 0.43\,\text{sec}$ (b) ASL $J = 5.71, J_{rel} = 0.53, t = 2.61\,\text{sec}$

(c) SASL $J = 8.16, J_{rel} = 0.37, t = 2.12\,\text{sec}$

Fig. 11: SL, ASL, SASL (threshold = 0.35) applied to a physical obstacle field with nominal clutter. ASL and SASL solutions are in the same homotopy class but the SL is not.

REFERENCES

[1] M. Pivtoraiko and A. Kelly , "Efficient constrained path planning via search in state lattices," in *The 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, September 2005.

[2] T. Howard, *Adaptive Model-Predictive Motion Planning for Navigation in Complex Environments*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Aug. 2009.

[3] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," tech. rep., 1998.

[4] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, pp. 846–894, June 2011.

[5] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, Aug 1996.

[6] T. M. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *The International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, 2007.

[7] M. Pivtoraiko and A. Kelly, "Kinodynamic motion planning with state lattice motion primitives," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2172–2179, Sept. 2011.

[8] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1879–1884, Oct 2009.

[9] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *International Journal of Robotics Research*, vol. 28, pp. 933–935, Aug. 2009.

[10] M. Pivtoraiko and A. Kelly, "Differentially constrained motion replanning using state lattices with graduated fidelity," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2611–2616, Sept 2008.

[11] S. Choudhury, J. D. Gammell, T. D. Barfoot, S. S. Srinivasa, and S. Scherer, "Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 4207–4214, IEEE, 2016.

[12] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*, pp. 489–494, May 2009.

[13] S. Karaman and E. Frazzoli, "High-speed flight in an ergodic forest," *CoRR*, vol. abs/1202.0253, 2012.

[14] M. Pivtoraiko, *Differentially Constrained Motion Planning with State Lattice Motion Primitives*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, February 2012.