Learning Models for Predictive Adaptation in State Lattices

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

Abstract Approaches to autonomous navigation for unmanned ground vehicles rely on motion planning algorithms that optimize maneuvers under kinematic and environmental constraints. Algorithms that combine heuristic search with local optimization are well suited to domains where solution optimality is favored over speed and memory resources are limited as they often improve the optimality of solutions without increasing the sampling density. To address the runtime performance limitations of such algorithms, this paper introduces Predictively Adapted State Lattices, an extension of recombinant motion planning search space construction that adapts the representation by selecting regions to optimize using a learned model trained to predict the expected improvement. The model aids in prioritizing computations that optimize regions where significant improvement is anticipated. We evaluate the performance of the proposed method through statistical and qualitative comparisons to alternative State Lattice approaches for a simulated mobile robot with nonholonomic constraints. Results demonstrate an advance in the ability of recombinant motion planning search spaces to improve relative optimality at reduced runtime in varyingly complex environments.

1 Introduction

Recent advances in sensors, computing, and algorithms for perception, planning, and control have allowed unmanned ground vehicles (UGVs) to be deployed in increasingly challenging and harsh conditions. Applications such as planetary exploration, nuclear inspection, and resource extraction could require robots to traverse environments that are dangerous or difficult for human operators and situations where

M. E. Napoli (✉) · H. Biggie · T. M. Howard
University of Rochester, Rochester, NY 14627, USA
e-mail: mnapoli@ur.rochester.edu

H. Biggie
e-mail: hbiggie@u.rochester.edu

T. M. Howard
e-mail: thomas.howard@rochester.edu

M. Hutter and R. Siegwart (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 5, https://doi.org/10.1007/978-3-319-67361-5_19

assistance (if at all available) may be limited to remote teleoperation of the platform. In these situations, the UGV is dependent on its own autonomy to reliably balance the minimization of vehicle risk with energy consumption and achieve mission success. Often, risk avoidance in these fields is paramount and optimality of achieved planning solutions is strongly desired. In environments with many homotopically distinct classes, graph and sampling approaches have been employed successfully to find near optimal solutions. However, these algorithms provide higher fidelity plans often at the expense of computational runtime. This compromise between optimality and efficiency is fundamental to mobile robot motion planning and many algorithms feature a balance between the two.

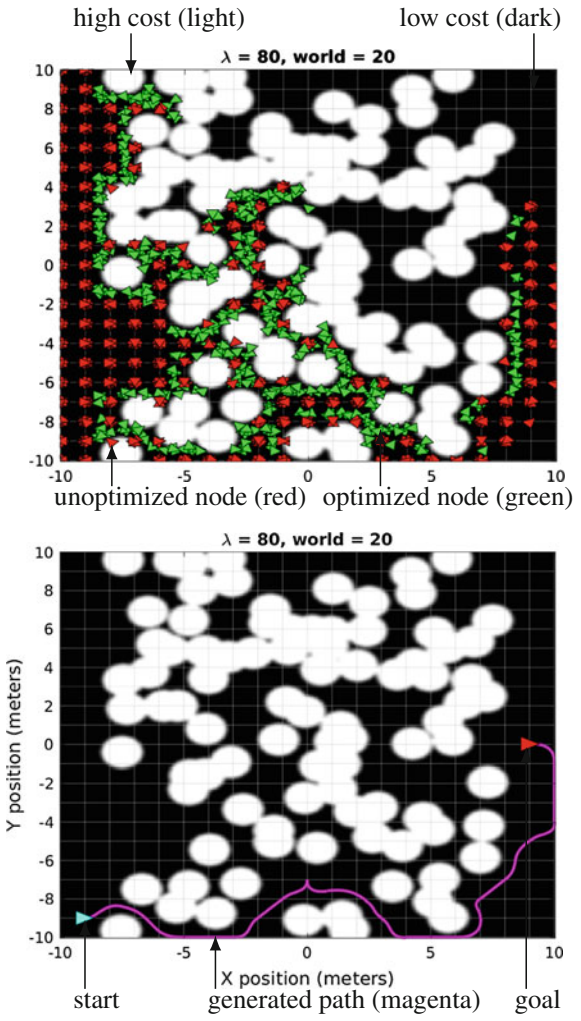


Fig. 1 An illustration of optimized and unoptimized nodes expanded during heuristic search (above) and the path generated by the PASL algorithm (below)

An approach to motion planning that considers memory, differential, and environmental constraints involves modeling the continuum of actions and states in a recombinant search space graph structure known as a State Lattice (SL) [1]. Such a formulation converts the motion planning problem into a graph search which can be solved using a variety of existing algorithms. It has also been shown that, in sufficiently complex environments, applying local optimization can improve the optimality of generated solutions [2]. A limitation of this approach, referred to as the Adaptive State Lattice (ASL), is that it indiscriminately optimizes every node that is encountered in the search space. This paradigm results in cycles spent optimizing regions that are not complex or for which optimization is too difficult and are unlikely to impact the resulting trajectory. This paper introduces a novel extension of the ASL, referred to as the Predictively Adapted State Lattice (PASL) which exploits a learned predictive model to adapt the representation of the search space by anticipating the amount of improvement to be obtained when applying optimization over local regions (Fig. 1). The addition of this model allows the PASL to focus optimization on regions where sufficient improvement is expected. The result from the PASL is a feasible solution with comparable optimality and memory requirements to the ASL approach, but with reduced computational runtime.

This paper presents three contributions, the first of which is the algorithm that incorporates a learned model for predictive adaptation in adaptive state lattices (PASL). The second is a statistical evaluation of the performance of PASL in both runtime and relative optimality, a metric which refers to the nearness of a solution to the global optimum, in a selection of randomly generated obstacle fields. This study assesses the PASL against a comparative baseline of SL, ASL and a heuristic based approach to selective adaptation referred to as the Selectively Adaptive State Lattice (SASL) [3]. The final contribution is a qualitative comparison between the solutions obtained with all four algorithms, which visualizes their performance in a random world of nominal complexity.

2 Related Works

Decades of research and advancements have led to a myriad of algorithms which address the motion planning problem in a variety of ways. These algorithms are often classified by the mechanism they exploit to sample the continuum of states and actions. One such classification are probabilistic approaches which leverage iterative random sampling and are often probabilistically complete. The Probabilistic Roadmap (PRM) is a probabilistic approach which randomly samples points in the admissible robot configuration space [4]. Subsequently, the sampled points are connected using a fast local planner. Rapidly-exploring Random Trees (RRTs) are another probabilistic sampling approach, which iteratively sample the state space continuum and expand a tree structure towards these sampled points until a connection to a goal region is made [5]. Extensions of this algorithm that bias sampling towards the goal region and utilize a bidirectional variant were shown to lead to faster

convergence. Further research has been explored to extend the representation within the RRT from a binary (admissible/inadmissable) to continuous representation which considers path optimality [6, 7]. It was proven that the RRT algorithm converges to a suboptimal solution and a new extension was proposed (called RRT*) which almost surely converges to the optimum [8]. Additional methods have been developed to include homotopy aware approaches, bi-directional RRT* variants and application of local optimization techniques which improve global optimality [9–11].

A limitation of approaches based on probabilistic sampling is the memory efficiency of the resulting search space. Another family of motion planning algorithms, referred to as the State Lattice (SL), is a recombinant search space approach that constructs a graph structure by regularly sampling the mobile robot state-action space [1]. These algorithms, which have been applied for navigation in autonomous automobiles [12] and planetary rovers [13], represent samples in the state space as nodes on the recombinant lattice and the edges between them are described by feasible actions that pre-encode system dynamics. The state space utilized in the original SL work consists of position, heading and curvature such that the state vector may be represented as $\mathbf{x} = \begin{bmatrix} x & y & \theta & k \end{bmatrix}$. Actions used in this work are parameterized functions (such as clothoids or polynomial spirals) of curvature and linear velocity and are generated using constrained optimization techniques [14, 15]. The state lattice is stored compactly in a structure referred to as the *control set* which contains a preselected sampling of states and actions describing the transitions from each node. An advantage of the SL is that nodes do not need to be allocated until their parent node is *expanded* (placed on the closed list) during the graph search. This feature reduces the memory requirements to be predominantly the storage of instantiated nodes. Further memory reduction can be obtained while maintaining optimality by utilizing an admissible heuristic in a search algorithm. SLs are resolution complete, meaning that all possible solutions will be considered within the sampling density of the state-action space.

3 Technical Approach

One of the primary limitations of the SL is the rigidity imposed by the regular sampling of the control set, where nodes are chosen irrespective of their associated cost in the robot's environment, resulting in many unused or unexpanded nodes sampled in high cost regions. To alleviate this limitation, an approach called the Adaptive State Lattice (ASL) was developed to optimize the sampled values of the state space before nodes are represented in an open list during heuristic search [2]. Adaptation of the representation within search has been shown to improve the performance of heuristic search in a State Lattice (shorter trajectories, faster runtime, lower memory utilization) over fixed search with a finer resolution SL in sufficiently complex environments. However, the ASL applies optimization to all nodes regardless of the potential for improvement. This can result in wasteful computations spent attempting to optimize regions where little gain is obtainable. An outline of the general search

process is shown in Algorithm 1 for the PASL and the three variants explored later in Sect. 4. Similarly to traditional graph search, expansion is performed on the top node in the open list generating child (L1) nodes. Instead of directly adding these nodes to the open list, a function is evaluated to predict which L1 nodes may benefit from adaptation. For each of the L1 nodes that exceed a particular threshold, their children (L2) nodes are generated and optimization is performed over that L1 parent. The prediction process for the SL always returns a *false* and conversely a *true* for the ASL. For completeness, the prediction step for the heuristic based SASL is outlined in Algorithm 2. The PASL prediction step is shown in Algorithm 3.

Algorithm 1: Predictive Graph Search

Input : Start node $\mathbf{x}_s$, goal node $\mathbf{x}_g$, step length α, line search parameter β, finite difference step Δ, predictive threshold (if applicable) c_{thresh}, predictive model (if applicable) net, training data mean (if applicable) $\hat{\mathbf{X}}$, training data standard deviation (if applicable) σ

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* $\mathbf{X}_{L1}$ **do**
      if $\mathbf{x}_{L1}$ *is not in OPEN* **then**
        if $predict(\mathbf{x}_p, \mathbf{x}_{1:N}, m, c_{thresh}, net, \hat{\mathbf{X}}, \sigma)$ **then**
          $\mathbf{X}_{L2} \leftarrow$ EXPAND($\mathbf{x}_{L1}$)
          adapt($\mathbf{x}_{L1}, \mathbf{X}_{L2}, \alpha, \beta, h$)
        end
        OPEN $\leftarrow \mathbf{x}_{L1}$
      end
    end
    closed $\leftarrow \mathbf{x}_{next}$
    sort(OPEN)
  end
end

3.1 Local Optimization in State Lattices

The ASL algorithm applies optimization of every instantiated node's state vector [2], where the values of the sampled state space are optimized but trajectories subject to

boundary state constraints of other nodes expressed in the state lattice. This procedure is described mathematically in Eq. 1, where $\mathbf{x_p}$ represents the current (later referred to as the parent) node's state vector containing position, heading, curvature, linear velocity, and/or other quantities of interest. The objective being minimized, denoted as $J_{agg}(\mathbf{x}_p)$ is the aggregate control set cost of the current node expansion.

$$\underset{\mathbf{x_p}}{\text{minimize}} \quad \mathrm{J}_{agg}(\mathbf{x_p}) \tag{1}$$

The aggregate control set cost is the sum of each of the edges connecting the current (parent) node to each of its child nodes as shown in Eq. 2 where $\mathbf{x}_n$ denotes the nth child node's state vector and $J(\mathbf{x}_p, \mathbf{x}_n)$ represents the cost of the edge between the parent and the nth child node.

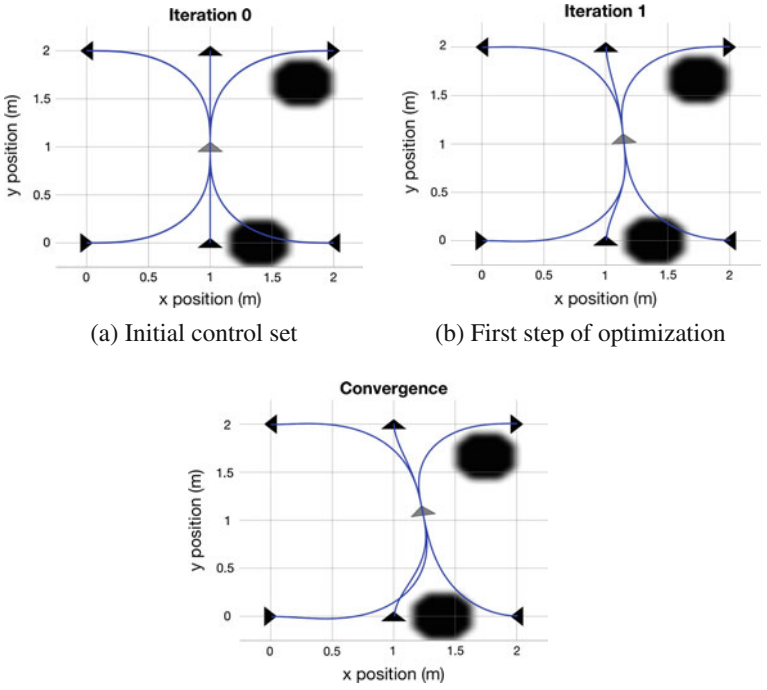
$$J_{agg}(\mathbf{x}_p) = \sum_{n=1}^{N} J(\mathbf{x}_p, \mathbf{x}_n) \tag{2}$$

The cost of a single edge is the sum of the nth child node's total edge path traversal time $s_{f,n}$ and integrated path cost denoted as $\mathscr{L}(\mathbf{x}_p, \mathbf{x}_n, s)$, as shown in Eq. 3 where $s_{0,n}$ and $s_{f,n}$ represent the nth child node's starting and final time respectively. The integrated path cost represents the cost of traversing a particular region in the robot's environment. In practice, the environment representation is usually defined using a discrete pixel approximation such as a cost map. Consequently, the integration in Eq. 3 is approximated numerically using the cost of pixels intersected by the edge.

$$J(\mathbf{x}_p, \mathbf{x}_n) = s_{f,n} + \int_{s_{0,n}}^{s_{f,n}} \mathscr{L}(\mathbf{x}_p, \mathbf{x}_s, s) ds \tag{3}$$

The optimization of the state vector adapts the parent node's edges to adjust the shape of the control set to conform to features in the local environment, which (in terms of the graph search) reduces the edge's cost and reflects a more optimal set of routes that would be represented by a finer representation of the search space. A visualization of the iterative optimization process applied to a simple cost map is shown in Fig. 2, highlighting how edges no longer cross high risk regions and are therefore better suited to the proximate environment.

The optimization technique in Fig. 2 is gradient descent which uses forward differencing to numerically estimate the aggregate control set cost objective gradient. Numerical estimates of the gradient are performed to enable evaluation of vehicle models that may contain complex models of dynamics and wheel-terrain interaction. When the state-action space is densely sampled, the computational overhead required to compute the gradient is consequential resulting in increased runtime of large searches. For further details about implementation and performance of the ASL method is presented in [2].



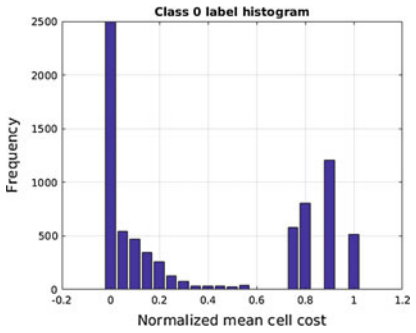
(a) Initial control set (b) First step of optimization

(c) Adapted state vector

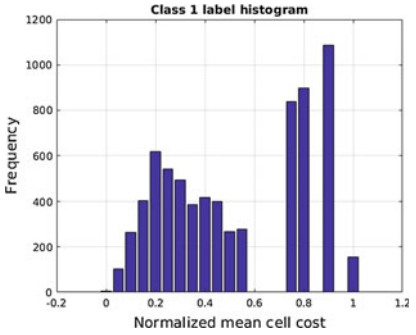
Fig. 2 State vector adaptation using gradient descent. The edges from the center parent node intersect high cost (dark) regions in the cost map. Iterative optimization adjusts the parent node's state vector until all edges traverse safer low cost regions

3.2 Heuristic Based Selective Adaptation

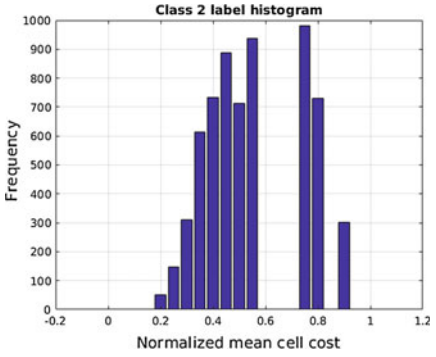
The application of local optimization allows the ASL to achieve higher relative optimality in many cases over the SL [2]. However, a major limitation is the indiscriminate optimization of nodes. In a statistical study, it was shown that a heuristic could be applied to selectively perform adaptation resulting in reduced runtime while providing solutions in the same homotopic class as the ASL [3]. The proposed algorithm, known as the SASL, computes the Normalized Mean Cell Cost (NMCC) of the environment patch spanned by the current node's expansion as a heuristic. This approach is outlined in Algorithm 2 and is included in the experiments discussed in Sect. 4. A threshold was chosen based on statistics collected over training data shown in Fig. 3.



(a) Histogram of negligible improvement



(b) Histogram of moderate improvement



(c) Histogram of significant improvement

Fig. 3 Histograms of improvement vs NMCC for threshold selection. A large amount of the improvement in class 2 (significant) occurs below NMCC = 0.6. Additionally, this is the case for moderate and negligible (class 1 and 0 respectively) improvement. A threshold of NMCC = 0.6 was chosen for the SASL based on these statistics

3.3 Predictive Adaptation of Search Space Representations

The SASL approach attempts to alleviate some of the computations wasted by the application of optimization to all nodes with the ASL, however experiments demonstrate that it is difficult to design an heuristic that is both accurate and efficient. Instead, we propose the use of a predictive model learned over a selection of training data to serve as the arbiter for predictive adaptation of state vectors in the search space. The model utilized in our proposed PASL method is an artificial neural network [16] where the input layer consists of the vectorized local region pixel values augmented with the parent node's orientation (θ_p) and the free function parameters used to describe all exiting edge curvature profiles. For consistency of the evaluation presented later in Sect. 4, the pixel region is identical to the patch utilized by SASL in Sect. 3.2. An example of the feature vector is shown in Eq. 4 where m_i represents the ith cost map patch index, $k_{j,n}$ the jth parameter of the curvature function for

Algorithm 2: Prediction for Selectively Adaptive State Lattice

Input : Parent node $\mathbf{x}_p$, child nodes $\mathbf{x}_{1:N}$, costmap m, heuristic threshold h
Output: Boolean prediction

Predict $(\mathbf{x}_p, \mathbf{x}_{1:N}, m, h)$
  $c \leftarrow 0$
  for i *in patch* $m(\mathbf{x}_p)$ **do**
    $c \leftarrow c + m_i$
  end
  $c \leftarrow \frac{c}{N_{cells} * MAXCOST}$
  if $c \leq h$ **then**
    return *true*
  end
  else
    return *false*
  end
end

nth child node, and $s_{f,n}$ the path length of the curvature profile for the nth child node. The local regions are 41×41 pixel patches which, when augmented with the additional features, results in an input layer of 1724 features. For every node during planning, the PASL collects all appropriate features and performs adaptation only if the model output is above a chosen threshold. Algorithm 3 outlines the PASL prediction procedure.

Algorithm 3: Prediction for Predictively Adapted State Lattice

Input : Parent node $\mathbf{x}_p$, child nodes $\mathbf{x}_{1:N}$, costmap m, improvement threshold h, trained artificial neural network net, mean of training data $\hat{X}$, standard deviation of training data σ
Output: Boolean prediction

Predict $(\mathbf{x}_p, \mathbf{x}_{1:N}, m, h)$
  $\theta_p \leftarrow orientation(\mathbf{x}_p)$
  $U \leftarrow parameters(\mathbf{x}_p, \mathbf{x}_{1:N})$
  $\mathbf{X} \leftarrow \begin{bmatrix} m & \theta_p & U \end{bmatrix}$
  $X \leftarrow X - \hat{X}$
  $X \leftarrow \frac{X}{\sigma}$
  $\hat{y} \leftarrow forward(net, X)$
  if $\hat{y} \geq h$ **then**
    return *true*
  end
  else
    return *false*
  end
end

$$\underline{\mathbf{x}} = \left[m_1\ m_2\ \ldots m_n\ \theta_p\ k_{1,1}\ \ldots\ s_{1,f}\ k_{2,1}\ \ldots\ s_{n,f}\right]^T \tag{4}$$

All inputs are zero centered and normalized to unit variance (calculated exclusively over training data) prior to applying the network with an architecture containing two hidden layers of 50 and 200 nodes respectively and hyperbolic tangent sigmoid activation functions. An output layer consisting of a single node and a linear activation function is used to represent the predicted improvement in objective. All training data was collected over a series of worlds generated using the random process discussed in Sect. 4 and the resulting data was binned into improvement intervals of 50. Subsequently, the data was sub-selected for training using stratified random sampling to prevent overfitting the model. All training targets were scaled by a factor of 10 as it was empirically determined that the model was overfitting to outputs of smaller magnitude.

Prior to employing the model in planning, a five fold cross validation experiment was performed using 70, 15, 15% for training, validation, and testing data respectively to evaluate the chosen network architecture. A series of threshold values were selected and binary classification (adaptation/no adaptation) was performed on each data point using the model's predicted improvement. The True Positive Rate (TPR) of correct classifications for each threshold value was averaged over five folds and are shown in Fig. 4. The results illustrate the model's classification performance of approximately 93–100% over the spanned threshold range.

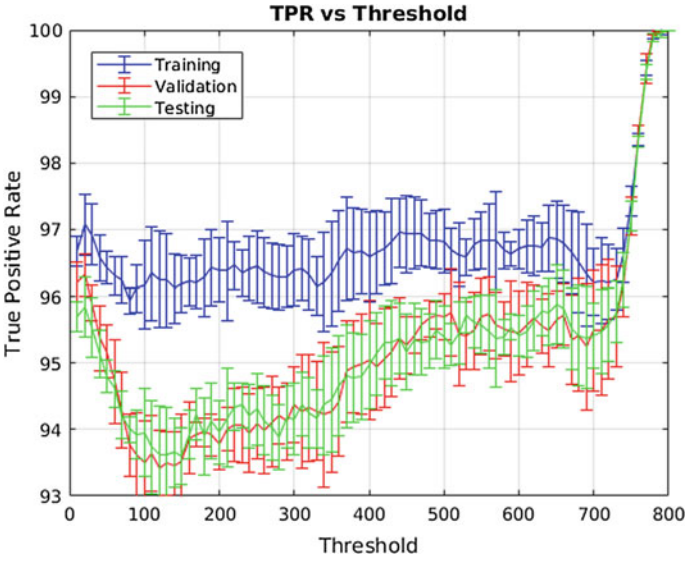
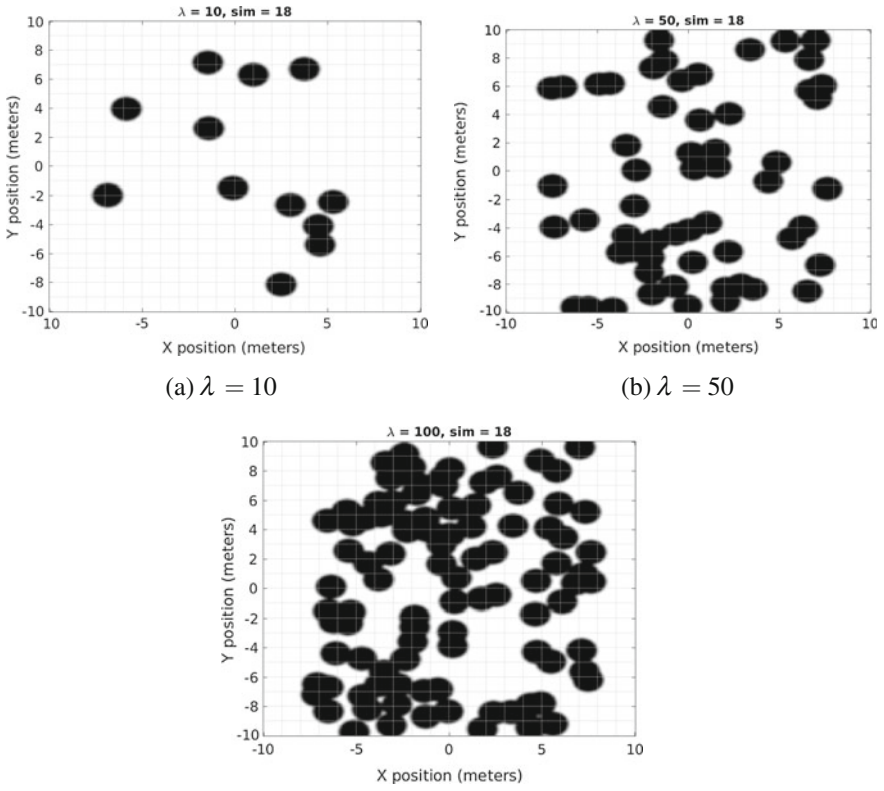


Fig. 4 Five-fold cross validation evaluation of neural network with 95% confidence

4 Experimental Design and Evaluation

Our proposed approach is focused on improving upon the performance of SL and as such, we restrict the scope of our subsequent analysis to variants of this search space representation. A statistical study was performed to evaluate the PASL and compare its performance with the SL, ASL, and SASL. All four algorithms were assessed over simulated random worlds with varying degrees of entropy, representing a mobile robot's environment. Each world contained specified regions where start and goal points were selected. These areas were common for all worlds and each algorithm was tasked with planning for every combination of world, start and goal point. The total number of plans solved for each algorithm was 32100. The simulations used in this evaluation do not include the separate set of random worlds generated strictly for model training from Sect. 3.3.



(a) $\lambda = 10$ (b) $\lambda = 50$

(c) $\lambda = 100$

Fig. 5 Sample of random worlds generated for simulation experiments. The number of obstacles (dark regions) increases with the Poisson expected rate of occurrence (λ). The start region for the planners is in the unoccupied space on the left, whereas the goal regions are on the right

Random worlds were created using a Poisson forest procedure similar to the approach described in [17]. The number obstacles were chosen using a Poisson distribution for ten levels of expected rate of occurrence (λ). These ranged from $\lambda = 10 - 100$ and also the free space world $\lambda = 0$. To model a penalty function of proximity to obstacles, the map was blurred using a Gaussian kernel and cropped by half a meter on all sides to avoid edge effects. Each map ranged from $(-10, 10)$ meters in x, y and was sampled at a resolution of 5 centimeters. The search algorithm was forbidden from expanding into regions with maximum cost. A consequence of this is that some worlds do not have solutions in the continuum and the planning algorithms will fail. A small selection of random worlds are shown in Fig. 5 to provide a qualitative representation of the planning difficulty for a range of λ.

4.1 Statistical Results

The relative optimality is defined as the ratio of the free space solution cost obtained with SL and the solution cost obtained by a planner for a world with a particular λ. Relative optimality and runtime results for all algorithms are shown in Fig. 6. Runtime results were obtained using an Intel Xeon(R) CPU E5-2520 v3 2.40 GHz processor.

The trends in Fig. 6 indicate that the ASL tends to outperform the other algorithms in terms of relative optimality, but also requires the most runtime. A notable exception to this is for SASL with a heuristic threshold of 0.7. Since the adaptation only considers local regions, there is no guarantee that it will improve the global objective. Therefore, performing optimization does not always result in a better solution. For this data point, it is believed that the SASL actually benefited from not performing optimizations in some instances.

The PASL performance is relatively consistent. As the improvement threshold increases, the quality of the solution degrades, however a decrease in runtime is obtained. With an improvement threshold of 200, the PASL runtime is consistently lower than SASL with a heuristic of 0.7. Furthermore, the relative optimalities of the two algorithms are comparable in the more cluttered obstacle fields. At less cluttered obstacle densities, the SASL tends to outperform the pasl in terms of relative optimality, however at significantly increased runtime. The increased runtime for the SASL algorithm is likely due to the difficulty of setting a good threshold when using a simple heuristic. An interesting comparison is between the SASL with a threshold of 0.6 and the PASL with a threshold of 300. Although the runtime is comparable between the two algorithms in the higher obstacle density worlds, the PASL is able to maintain higher relative optimality. In this domain, it appears that the PASL predictive model outperforms the SASL at selecting nodes to optimize given the higher relative optimality. Due to the challenging nature of the planning problems for higher λ, in many domains it is worthwhile to spend computational resources to improve the solution.

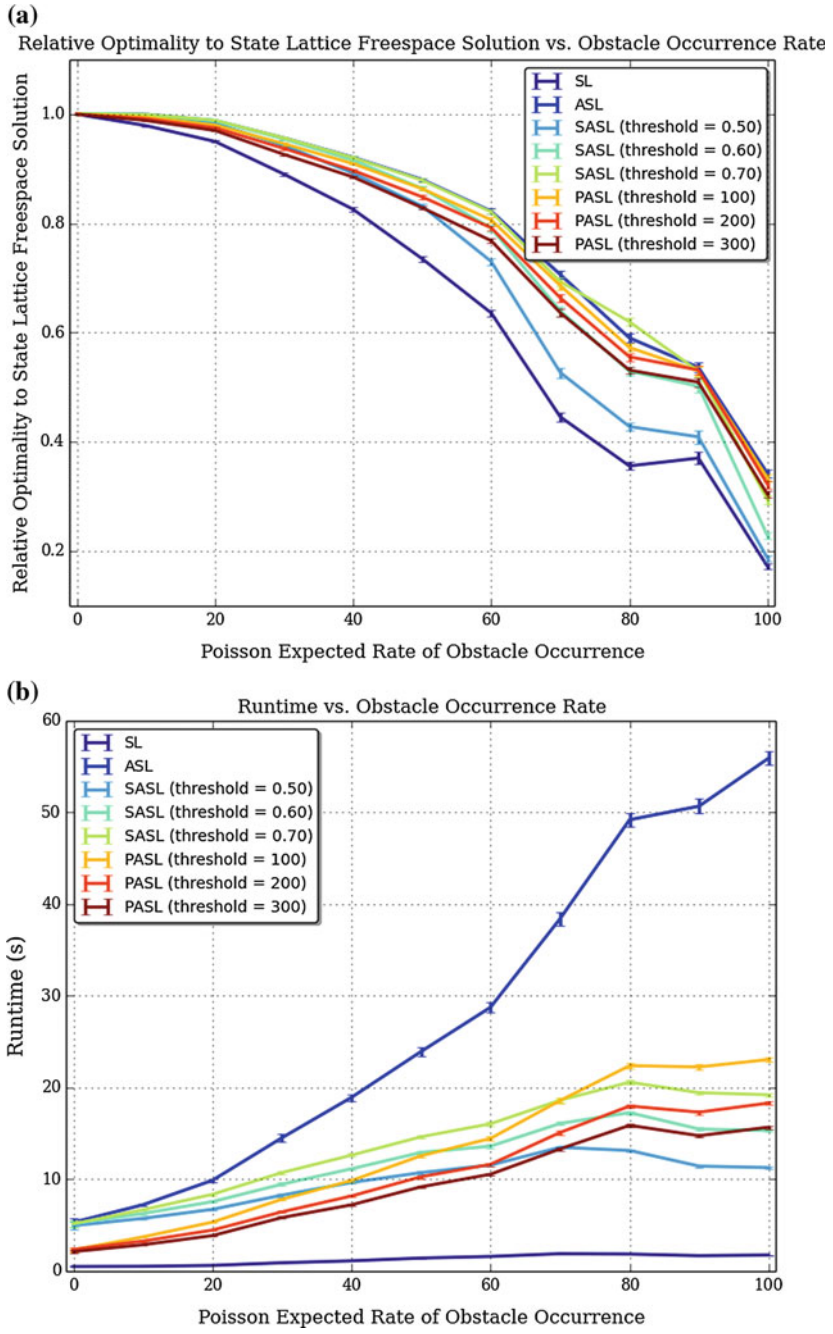
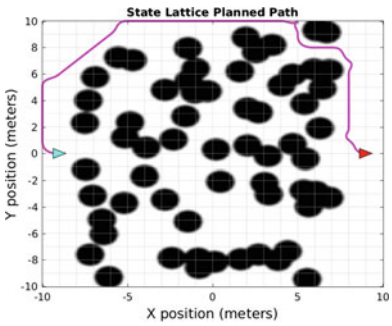


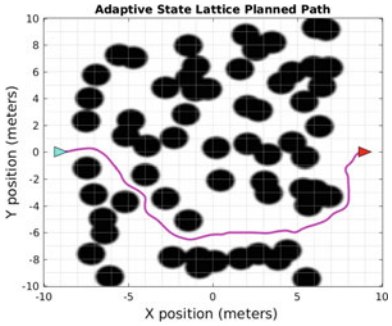
Fig. 6 Simulated random world study results for relative optimality **a** and runtime **b** versus λ presented with 95% confidence intervals

4.2 Comparative Results

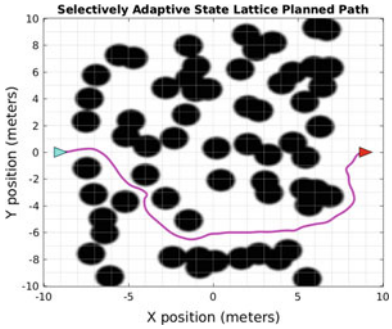
To examine the qualitative differences between the four algorithms a sample world is chosen with a nominal amount of clutter and the path representations are visualized in Fig. 7. The map in Fig. 7 is hand selected to represent planning in a moderately complex environment. The solution obtained with the SL is the fastest with a total runtime of 1.07 s, however it has the highest path cost at $J = 37.63$. Due to the regular sampling resolution, the unadapted search is unable to cut through the clutter to reach the goal. The ASL obtains the lowest path cost at $J = 28.00$, but with the highest runtime at 30.08 s. This search is able to optimize to the cost map and weave through obstacles allowing it to achieve a lower cost solution. Similarly to the ASL, the SASL is also able to apply some amount of optimization and achieves a path cost of $J = 33.33$ with a runtime of 15.69 s. The PASL performs similarly but with a lower path cost at $J = 28.54$ and faster runtime at 11.67 s. For this sample, the PASL



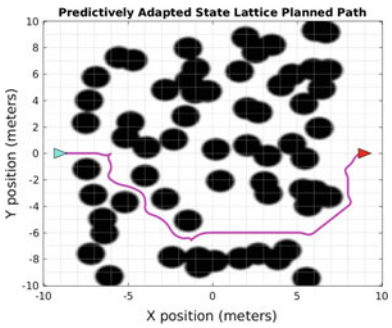
(a) State Lattice Path Visualization (cost = 37.63, runtime = 1.07 seconds)



(b) Adaptive State Lattice Path Visualization (cost = 28.00, runtime = 30.08 seconds)



(c) Selectively Adaptive State Lattice Path Visualization (cost = 33.33, runtime = 15.69 seconds, threshold = 0.7)



(d) Predictively Adapted State Lattice Path Visualization (cost = 28.54, runtime = 11.67 seconds, threshold = 200)

Fig. 7 Qualitative comparison between the solutions obtained by each algorithm tasked with planning from the start node (cyan) to the goal (red) in a randomly generated world with $\lambda = 60$

was able to produce a result comparable to the ASL, with a significantly reduced runtime. An interesting note here is that the ASL, SASL, and PASL solutions all belong to the same homotopic class whereas the SL solution does not. This seems to indicate that the application of predictive adaption can result in solutions of similar quality to fully adapted search spaces, but with large reductions in runtime.

5 Conclusions and Future Work

As the prevalence of UGVs increases, computationally efficient and safe motion planning algorithms become ever more crucial. For applications where memory resources are limited and risk mitigation is paramount, the SL and its extensions are well suited due to their ability to obtain deterministic, resolution optimal solutions that inherently satisfy nonholonomic constraints. Improvements over resolution optimality of the SL is shown to be possible by applying local optimization over samples in the graph. In this paper, we have shown that a learned predictive model can achieve nearly the same optimality as the ASL with significantly reduced runtime requirements and outperform simple hand-coded thresholds for selective adaptation. Statistically significant results are obtained using simulations in random worlds which show an improvement over the SASL and the SL in relative optimality and the SASL and ASL in runtime.

Future work involving the presented algorithm includes optimizations for improving the runtime performance, field experiments in partially observed environments, and adaptation of richer spatial-semantic models of the underlying representation. Although thorough assessment of the algorithm requires examining the performance over many planning scenarios, implementation and validation of these experiments using a physical platform is valuable. The scope of this paper is to improve the performance this particular class of motion planning algorithms, however future work involves comparisons between probabilistic sampling approaches such as RRTss and PRMs.

Acknowledgements This work was supported in part by the National Science Foundation under grant IIS-1637813.

References

1. Pivtoraiko, M., Knepper, R.A., Kelly, A.: Differentially constrained mobile robot motion planning in state lattices. J. Field Robot. **26**, 308–333 (2009)
2. Howard, T.: Adaptive Model-Predictive Motion Planning for Navigation in Complex Environments. Ph.D. thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Aug 2009
3. Napoli, M., Biggie, H., Howard, T.M.: On the performance of selective adaptation in state lattices for mobile robot motion planning. In: Proceedings of the IEEE/RSJ International Con-

ference on Intelligent Robots and Systems, Sept 2017
4. Kavraki, L.E., Svestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **12**, 566–580 (1996)
5. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. In: Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C), vol. 1, pp. 473–479 (1999)
6. Urmson, C., Simmons, R.: Approaches for heuristically biasing rrt growth. IEEE/RSJ Int. Conf. Intell. Robot. Syst. **2**, 1178–1183 (2003)
7. Jaillet, L., Cortes, J., Simeon, T.: Transition-based rrt for path planning in continuous cost spaces. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2145–2150, Sept 2008
8. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. **30**, 846–894 (2011)
9. Yi, D., Goodrich, M.A., Seppi, K.D.: Homotopy-aware rrt*: Toward human-robot topological path-planning. In: 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pp. 279–286. IEEE (2016)
10. Starek, J., Schmerling, E., Janson, L., Pavone, M.: Bidirectional fast marching trees: an optimal sampling-based algorithm for bidirectional motion planning. In: Workshop on Algorithmic Foundations of Robotics (2014)
11. Choudhury, S., Gammell, J.D., Barfoot, T.D., Srinivasa, S., Scherer, S.: Regionally accelerated batch informed trees (rabit*): a framework to integrate local information into optimal path planning. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), May 2016
12. Likhachev, M., Ferguson, D.: Planning long dynamically feasible maneuvers for autonomous vehicles. Int. J. Robot. Res. **28**, 933–935 (2009)
13. Pivtoraiko, M., Kelly, A.: Differentially constrained motion replanning using state lattices with graduated fidelity. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2611–2616 (2008)
14. Kelly, A., Nagy, B.: Reactive nonholonomic trajectory generation via parametric optimal control. Int. J. Robot. Res. **22**, 583–601 (2003)
15. Howard, T.M., Kelly, A.: Optimal rough terrain trajectory generation for wheeled mobile robots. Int. J. Robot. Res. **26**(2), 141–166 (2007)
16. Lippmann, R.P.: An introduction to computing with neural nets. SIGARCH Comput. Archit. News **16**, 7–25 (1988)
17. Karaman, S., Frazzoli, E.: High-speed flight in an ergodic forest. CoRR (2012). arXiv:1202.0253