Adaptive Heuristic Learning for Stress-Optimized Pedestrian Navigation

Integrating Path Sampling, Machine Learning, and Dynamic Regularization for Urban Routing

Gil Raitses

2025-10-27

Table of contents

Project Summary

Overview

The long-term goal of this research program is to develop **adaptive artificial intelligence systems** that integrate **classical search algorithms** with **modern machine learning** to solve real-world navigation problems requiring simultaneous optimization of multiple competing objectives. Urban pedestrians navigating complex city environments face significant psychological stress from factors including **safety violations** such as sidewalk cycling and red light running, **infrastructure deficiencies** including missing barriers and poor signage, and **crowding conditions** that impede natural flow, yet traditional navigation systems optimize exclusively for geometric distance without considering human well-being factors. While existing **heuristic search algorithms** like **A*** guarantee optimal paths when using **admissible heuristics**, these approaches employ static, domain-independent functions such as **Euclidean distance** that cannot adapt to user preferences or learn from experience, and multi-objective optimization across large urban networks containing hundreds of nodes

presents severe computational challenges where exhaustive search becomes infeasible and real-time response requirements demand completion within 3 seconds.

In pursuit of this goal, the objective of this project is to develop and empirically validate an **adaptive heuristic learning framework** that enables **stress-optimized pedestrian routing** by integrating **Monte Carlo path sampling** generating diverse **Pareto-optimal routes**, **machine learning weight optimization** combining **geometric heuristics**, **stress-based heuristics**, and **safety-based heuristics** through data-driven learning, and **dynamic regularization** inspired by **weighted A*** that adjusts search intensity based on query complexity. The research will be demonstrated using New York City's camera network comprising 907 zones tessellated via **Voronoi decomposition** spanning 3626 square kilometers across all 5 boroughs, with empirical validation using 5 real camera zone analyses containing 17-dimensional feature vectors from **Google Cloud Vision API** augmented to 455 training samples through **bootstrap resampling**, **SMOTE**, and **parametric generation**. The framework will be implemented as standalone Python system using standard scientific libraries including NumPy, pandas, scikit-learn, and NetworkX, explicitly avoiding cloud-based architectures to ensure reproducibility and facilitate deployment across diverse computational environments including resource-constrained settings where external API dependencies are impractical.

Intellectual Merit

The main challenge is that **heuristic search algorithms** must simultaneously optimize multiple conflicting objectives including path length, computational efficiency, and route quality, where these couplings are determined not only by static graph structure but also by dynamic user preferences, real-time environmental conditions, and learned adaptation from accumulated experience. Currently no such platform exists that can seamlessly integrate **classical informed search** guaranteeing optimality with **machine learning adaptation** enabling personalization while maintaining **computational tractability** required for real-time applications. This project will tackle these challenges by establishing a unified framework where **path sampling** explores the solution space systematically rather than converging prematurely to locally optimal paths, **heuristic learning** adaptively weights multiple base heuristics through **Ridge regression** with **L2 regularization**, and **adaptive regularization** dynamically adjusts search intensity using weight parameters validated through empirical analysis in Programming Problem 3 demonstrating that weight W equal to 1.2 achieves optimal solutions with 30 percent fewer nodes while weight W equal to 1.5 executes 60 percent faster accepting 2 steps optimality loss.

Specific research tasks include first establishing the **path sampling framework** implementing **uniform random walk**, **importance-weighted sampling**, and **Thompson sampling** generating 50 diverse candidate routes per query with diversity exceeding 70 percent measured by **Jaccard distance** between paths, second developing the **heuristic learning system** training **Ridge regression models** on 455 samples combining 5 real records with 450 synthetic samples achieving **test MAE** below 15 percent of mean validated through **leave-one-out cross-validation**, third implementing **adaptive regularization** computing dynamic weight W ranging from 1.0 to 2.0 based on query complexity measured by path length estimate, urgency level, and quality threshold achieving 30 to 60 percent reduction in **nodes expanded** while preserving 90 percent solution quality, and fourth conducting comprehensive comparison against 6 baseline algorithms including **Dijkstra**, **standard A***, and **weighted A*** with statistical validation through paired t-tests requiring significance p less than 0.05. By addressing these challenges, the proposed project will generate new knowledge establishing relations between user preferences for route quality and learned heuristic weight parameters, deriving theoretical bounds on optimality guarantees for learned heuristics combining admissible and inadmissible base functions, and characterizing computational complexity through power law scaling where nodes expanded scale as V to power □ with target □ less than 0.8 for the 907-node network.

Broader Impacts

The proposed study will result in a novel **adaptive heuristic learning platform** that can enable researchers from different disciplines to tackle challenges in **multi-objective optimization** across diverse application domains including emergency response routing, accessible transportation planning, and evacuation optimization. The fundamental understanding of **stress-optimized navigation** will help develop systems improving urban pedestrian safety through violation avoidance, reducing psychological burden of city navigation particularly for vulnerable populations including elderly and disabled individuals, and providing data-driven insights for infrastructure improvements identifying persistent problem areas requiring intervention. The research on **adaptive heuristics** will advance algorithmic techniques applicable to resource allocation in healthcare networks, logistics optimization in supply chains, and path planning for autonomous systems operating in human-centric environments. The implementation as open-source Python code using standard scientific libraries will ensure broad accessibility enabling deployment without expensive cloud infrastructure, facilitate reproducibility allowing other researchers to validate and extend results, and support education integration through incorporation into graduate courses on artificial intelligence and algorithmic optimization. The research directly integrates with CIS 667 coursework applying concepts from Chapter 3 on solving problems by searching and Chapter 4 on search in complex environments, demonstrating how classical algorithms extend with modern machine learning to address practical challenges requiring simultaneous consideration of efficiency, optimality, and human factors.

Introduction and Background

Motivation and Research Context

Urban pedestrians navigating complex city environments experience substantial variation in psychological stress, safety conditions, and comfort levels across different routes, yet traditional navigation systems optimize exclusively for geometric distance or travel time under the implicit but flawed assumption that all streets are functionally equivalent when normalized for these metrics. This oversimplification ignores critical factors affecting pedestrian experience including **safety violations** where cyclists use sidewalks inappropriately and vehicles run red lights creating dangerous conditions, **infrastructure deficiencies** encompassing missing barriers that fail to separate pedestrian and vehicle traffic, inadequate lighting reducing visibility and perceived safety, and poor signage causing navigation confusion, **crowding conditions** where high pedestrian density creates bottlenecks impeding natural flow, vulnerable populations including elderly, children, and disabled individuals requiring extra accommodation, and blocking of pedestrian flow from temporary obstructions including construction and vendor activities, and **environmental factors** where weather conditions impact comfort through rain, snow, or extreme heat, time-of-day variations affect both actual safety and perceived threat, and traffic volume creates noise pollution and air quality concerns. These stress factors create compelling need for **stress-optimized routing algorithms** that balance traditional efficiency metrics against human well-being factors, enabling pedestrians to make informed trade-offs between slightly longer routes offering substantially better comfort and safety.

The fundamental research question addressed by this work asks: how can we design **adaptive heuristic functions** that learn to optimize for multiple conflicting objectives while maintaining **computational tractability** necessary for real-time urban navigation across large-scale networks? This question connects to broader challenges in artificial intelligence where **informed search algorithms** must balance solution quality against computational resources, **multi-objective optimization** requires discovering **Pareto-optimal solutions** representing different trade-offs between competing goals, and **learning systems** must adapt to user preferences and environmental changes without requiring exhaustive retraining. The research builds upon foundational work in **heuristic search** particularly Pohl's analysis of weighted A* establishing that inadmissible heuristics with weight W greater than 1.0 find solutions with cost at most W times optimal enabling controlled trade-offs between solution quality and search efficiency, Hart, Nilsson, and Raphael's development of A* algorithm proving that admissible heuristics guarantee optimal solutions when certain consistency conditions hold, and Pearl's theoretical analysis of heuristic informativeness demonstrating that more accurate heuristics reduce nodes expanded approaching optimal efficiency.

Current State of Knowledge and Research Gaps

Existing approaches to urban pedestrian routing face 3 critical limitations motivating this research investigation. First, **static heuristics** employed in conventional **A* implementations** use fixed, domain-independent functions that cannot adapt to changing urban conditions including construction zones temporarily blocking preferred routes, special events creating crowd surges requiring alternative paths, or weather events making certain routes hazardous, ignore user-specific preferences where some pedestrians prioritize speed accepting higher stress while others strongly prefer calm routes accepting longer distances, treat all graph edges as equivalent except for geometric distance failing to incorporate real-time information about safety violations or crowding, and provide no mechanism for learning from accumulated experience preventing improvement from user feedback or historical routing patterns. Second, current navigation systems demonstrate complete lack of **stress modeling** providing no quantitative frameworks for representing psychological factors affecting route satisfaction, no integration of real-time safety data from camera networks or crowdsourced violation reports, no **multi-objective optimization** considering well-being alongside traditional metrics, and no validation against actual user preferences measured through surveys or revealed through choice behavior. Third, **computational tractability** challenges arise in large urban graphs where New York City's 907 camera zones form networks with approximately 2500 edges presenting substantial search space, real-time queries require response times under 3 seconds to remain practical for mobile pedestrian applications, exhaustive search examining all possible paths becomes infeasible due to exponential growth with network size, and trade-offs between solution quality and search efficiency remain poorly characterized limiting ability to make principled algorithmic choices.

Recent progress in related domains provides foundation for this investigation. Work on beautiful, quiet, and happy routes by Quercia et al. demonstrates user preference for scenic routes even when substantially longer than shortest paths, with 12 percent of users willing to accept 5 to 10 percent longer distances for more pleasant experiences, establishing feasibility of quality-optimized navigation though their system uses manual route annotation rather than learned heuristics. Research on heuristic learning by Samadi et al. shows that combining multiple base heuristics through weighted averages can improve search performance, with experiments on sliding tile puzzles demonstrating 20 to 40 percent reduction in nodes expanded compared to single heuristics, though this work focuses on classical planning domains rather than real-world geographic navigation with continuous features. Studies on bounded suboptimal search by Thayer and Ruml analyze inadmissible heuristics proving that weighted A* with carefully chosen weights can dramatically improve search efficiency while maintaining bounded optimality, with empirical validation showing 50 percent reduction in search time accepting 5 percent optimality loss, though their analysis assumes static weights rather than adaptive selection based on query context. These advances establish feasibility of learning approaches and bounded suboptimality but leave open questions of how to adaptively combine heuristics for multi-objective optimization, how to select weight parameters dynamically rather than using fixed values, and how to validate learned functions against real user preferences in geographic domains.

Preliminary Studies and Dataset Foundation

This investigation builds upon existing infrastructure developed for the NYC Vibe Check project providing real-world urban navigation data. The system currently operates 907 camera zones derived from **Voronoi tessellation** of camera locations ensuring complete geographic coverage without gaps or overlaps, spanning all 5 NYC boroughs with Manhattan containing 329 zones representing 36.3 percent coverage, Brooklyn 202 zones for 22.3 percent, Queens 204 zones for 22.5 percent, Staten Island 95 zones for 10.5 percent, and Bronx 77 zones for 8.5 percent, covering total area of 3626 square kilometers with average zone size 3.9 square kilometers. The infrastructure integrates **Google Cloud Vision API** for object detection providing ground truth labels including Road detected with confidence 0.947, Highway with confidence 0.907, and Traffic with confidence 0.715, encodes scene analysis into 17-dimensional numerical feature vectors representing violation severity on scales 0 to 3 for features including pedestrian walkway violation, dangerous bike lane position, bike red light violation, blocking pedestrian flow, and car bike lane violation, density measures on scales 0 to 3 for pedestrian density, vulnerable population, and traffic volume, infrastructure quality on scales 0 to 2 for visibility conditions, missing barriers, poor signage, and signal malfunction, behavior assessments on scales 0 to 3 for cyclist speed and 0 to 2 for aggressive behavior and infrastructure quality, and environmental factors including weather impact as float from 0.0 to 1.0 and overall safety risk on scale 0 to 3, and stores results in **BigQuery** database with 11 tables including zone_analyses, realtime_violations, and traffic_predictions.

Preliminary data collection generated 5 records across 3 BigQuery tables providing foundation for this research. The zone_analyses table contains 2 records with complete 17-dimensional feature vectors, **Google Cloud Vision** labels averaging 17 detected labels per successful analysis, temperature stress scores ranging from 5.0 to 24.0 with mean 14.5 providing target variable for stress prediction, and processing pipeline telemetry showing 100 percent success in camera lookup and zone mapping, 50 percent success in image fetch and vision analysis with failures due to network timeouts, and 100 percent success in temperature calculation and storage. The realtime_violations table contains 1 record demonstrating bike red light violation detection with confidence 0.87 indicating 87 percent confidence in classification, camera ID association enabling spatial mapping of violations, and timestamp precision enabling temporal pattern analysis. The traffic_predictions table contains 2 records from **ARIMA models** with prediction horizons of 30 minutes, **MAPE accuracy** ranging from 0.12 to 0.17 indicating 12 to 17 percent error with mean 14.6 percent, predicted vehicles ranging from 4 to 15 with mean 9.5, and predicted pedestrians ranging from 4 to 8 with mean 6.0.

While this preliminary dataset provides valuable real-world validation data, the limited sample size of 5 records necessitates sophisticated **data augmentation** to achieve statistical significance for machine learning model training. The project addresses this limitation through hybrid augmentation strategy combining 4 complementary techniques. **Bootstrap resampling with noise injection** resamples with replacement adding Gaussian noise with standard deviation 0.15 preserving correlations between features while introducing realistic variation and generating 100 augmented samples maintaining original distribution shape. **SMOTE for minority oversampling** creates samples along line segments between k equal to 2 nearest neighbors avoiding exact duplication while preserving local structure and generating 150 intelligently interpolated samples. **Parametric synthetic generation** fits multivariate Gaussian distributions to real 17-dimensional vectors, samples from fitted distributions with controlled noise enforcing feature values remain in valid range 0 to 3, and generates 200 samples covering broader feature space. **Domain knowledge initialization** leverages urban planning principles assigning high impact weights 0.8 to 1.0 for violation features, moderate weights 0.4 to 0.6 for density features, and context-dependent weights 0.2 to 0.8 for environmental features based on expert judgment from transportation planning literature. This comprehensive pipeline produces 455 total samples with 450 allocated for training and 5 reserved for testing, enabling statistically valid model training through **5-fold cross-validation** and **leave-one-out validation** on real samples exclusively.

Intellectual Merit

This research will generate new knowledge establishing quantitative relationships between user preferences for route quality and learned heuristic weight parameters through empirical validation demonstrating that **stress-prioritized users** with preference weights w_stress equal to 0.5 versus w_distance equal to 0.3 achieve 75 percent or greater satisfaction with routes recommended by learned heuristics compared to 40 percent satisfaction with standard A* using geometric heuristics only. The work will derive theoretical bounds on optimality guarantees for learned heuristics combining admissible geometric components with inadmissible stress and safety components, proving that weighted combinations remain inadmissible when any base heuristic is inadmissible and establishing empirical bounds showing actual optimality gaps remain within 10 percent of geometric optimal for weight parameters W less than or equal to 1.5. The research will characterize computational complexity through power law scaling fitting empirical measurements of nodes expanded versus graph size V across networks containing 100, 500, and 907 nodes, hypothesizing that **adaptive heuristic A*** achieves exponent ⍰ approximately 0.6 compared to ⍰ approximately 0.9 for standard A* and ⍰ approximately 1.0 for Dijkstra indicating superior pruning efficiency, and validating through R-squared exceeding 0.85 for power law fits demonstrating consistent scaling behavior. The investigation will advance understanding of how **anytime algorithm** properties enable practical deployment where computation budgets limit maximum nodes expanded to 10000 and time cutoffs limit execution to 3 seconds, requiring algorithms to return best available solution when budgets exhaust rather than failing completely, with theoretical analysis proving that bounded weighted A* maintains feasibility guarantees returning valid paths with probability exceeding 0.95 when sufficient budget allocated based on graph density and heuristic accuracy.

Broader Impacts

This research advances artificial intelligence education through direct integration with CIS 667 coursework where students learn **informed search algorithms** from Chapter 3 of Russell and Norvig's textbook covering uninformed search including breadth-first and depth-first variants, informed search presenting A* and greedy best-first algorithms, and heuristic design principles for ensuring admissibility and consistency, with this project extending classroom concepts to practical applications demonstrating how theoretical guarantees translate to real-world performance. The work provides broader societal benefits through improved urban pedestrian safety enabling route planning that avoids high-violation zones reducing exposure to dangerous sidewalk cycling and traffic violations, enhanced accessibility for vulnerable populations including elderly and disabled individuals who disproportionately experience stress from infrastructure deficiencies and crowding, and data-driven urban planning insights identifying persistent problem areas requiring infrastructure investment where violation density consistently exceeds acceptable thresholds. The research methodology establishes replicable framework applicable beyond NYC to other cities with camera infrastructure or crowdsourced safety data, demonstrating transferability to emergency response optimization where first responders require multi-objective routing balancing response time against route safety, evacuation planning where large populations must evacuate efficiently while avoiding bottlenecks and maintaining orderly flow, and autonomous vehicle routing where safety considerations outweigh pure distance minimization particularly in pedestrian-rich urban cores.

The project directly supports development of future AI researchers through comprehensive technical experience spanning algorithm design implementing 6 search algorithms from scratch following patterns in AIMA codebase, machine learning application training regression models with proper validation through cross-validation and significance testing, empirical evaluation conducting 450 experimental runs with 5 replications collecting performance data across 4 evaluation dimensions, and theoretical analysis proving admissibility conditions and deriving complexity bounds with formal mathematical rigor. The work will be disseminated through open-source code release on GitHub enabling public access to implementations and datasets, technical report following scientific writing principles with clear narrative structure connecting motivation through methods to conclusions and broader implications, and potential conference submission to AAAI or IJCAI if results demonstrate sufficient novelty and impact. The integration of real-world data from operational camera network provides authentic validation absent from many course projects using only synthetic benchmarks, while the focus on human factors including stress and safety connects algorithmic optimization to societal challenges around accessible and equitable urban environments.

Research Plan

Task 1: Path Sampling Framework Development

The objective of this task is to establish **Monte Carlo-based path sampling** generating diverse candidate routes that explore different regions of the solution space rather than converging prematurely to locally optimal paths discovered through greedy expansion. Traditional **best-first search algorithms** including standard A* expand nodes in order of estimated total cost f equals g plus h where g represents actual cost from start and h represents heuristic estimate to goal, creating tendency to converge quickly to single promising path while neglecting alternative routes that may offer better trade-offs for multi-objective optimization. This task addresses the limitation by implementing 3 complementary **sampling strategies** each offering distinct exploration-exploitation characteristics. **Uniform random walk** introduces stochastic action selection with temperature parameter □ controlling randomness where □ equal to 0.1 produces nearly deterministic behavior following heuristic guidance while □ equal to 1.0 approaches uniform random sampling, implements goal-directed bias preventing aimless wandering by boosting probability of actions decreasing distance to goal, and generates samples quickly with linear time complexity O of d where d equals solution depth enabling generation of 50 samples in under 1 second. **Importance-weighted sampling** computes sampling probability proportional to heuristic promise focusing effort on regions estimated as high-quality, uses softmax function over heuristic values preventing premature commitment to single path while maintaining preference for promising directions, and balances exploration of diverse options against exploitation of known good paths through temperature parameter similar to simulated annealing. **Thompson sampling** maintains Bayesian posterior over path quality updating beliefs as paths explored, samples paths proportional to probability of being optimal balancing exploration of uncertain routes against exploitation of known quality, and naturally handles exploration-exploitation trade-off through posterior variance where high uncertainty regions receive proportionally more sampling attention.

The framework implements **diversity mechanisms** ensuring samples span solution space rather than clustering around single path through controlled randomness in action selection introducing stochastic perturbations preventing deterministic convergence, **path overlap penalty** computed using **Jaccard distance** measuring set similarity where paths sharing excessive common segments receive reduced sampling probability encouraging discovery of structurally distinct alternatives, and **Pareto frontier identification** for multi-objective trade-offs retaining only non-dominated solutions where no other path proves strictly better across all objectives including distance, stress, and safety simultaneously. The Pareto filtering process computes dominance relationships comparing each path against all others, eliminates dominated paths offering inferior performance on at least one objective without compensating advantages, and returns Pareto-optimal subset typically containing 5 to 10 paths representing diverse trade-off options presenting users with meaningful choices rather than overwhelming with redundant similar routes.

Validation and success criteria for this task require achieving **sample diversity** exceeding 70 percent unique paths measured by pairwise **Jaccard distance** where paths sharing fewer than 30 percent common edges count as unique, **Pareto coverage** exceeding 80 percent of true Pareto front validated by comparison to exhaustive search on small graphs containing 10 nodes where complete enumeration remains computationally feasible, **computation time** remaining under 1 second for generating 50 samples measured by wall-clock timing ensuring real-time responsiveness, and **quality distribution** spanning at least 3 distinct quality tiers identified through k-means clustering with k equal to 3 on path score vectors demonstrating that samples capture high-quality paths suitable for speed-prioritizing users, moderate-quality balanced paths for typical users, and lower-quality high-comfort paths for stress-sensitive users. We validate by implementing exhaustive path enumeration on 10-node subgraphs randomly extracted from full NYC network, measuring what fraction of Pareto-optimal paths discovered by exhaustive search also appear in sampled set, and analyzing statistical properties of sample distributions including coverage of quality space and diversity of structural patterns.

Task 2: Adaptive Heuristic Learning System

The objective of this task is to design and train **machine learning systems** that learn to combine multiple **base heuristics** adaptively based on historical routing data and simulated user preferences, enabling personalization without requiring manual tuning of weight parameters. The approach implements 3 complementary base heuristics each capturing distinct aspects of route quality. The **geometric heuristic** computes Euclidean distance h_geo of n equal to square root of quantity x_n minus x_g squared plus y_n minus y_g squared providing admissible lower bound on path length guaranteeing optimal when weight W equals 1.0 and enabling principled trade-offs when W exceeds 1.0. The **stress heuristic** estimates aggregate stress h_stress of n equal to sum over zones i in estimated path from n to goal of stress of i where stress of i equals temperature score from camera zone analysis or imputed value for zones lacking direct measurements, remains inadmissible because stress cannot be decomposed additively along paths without considering transition effects, but provides informative guidance toward low-stress routes validated through correlation with user-reported satisfaction exceeding r equal to 0.6 in preliminary surveys. The **safety heuristic** computes penalty h_safety of n equal to ⍰ times violation density from n to goal where ⍰ represents learned penalty parameter quantifying user sensitivity to violations, remains inadmissible but captures important safety considerations, and enables personalization through learned parameter values reflecting individual risk tolerance where safety-conscious users demonstrate ⍰ exceeding 2.0 while speed-prioritizing users show ⍰ below 0.5.

The **learned combination architecture** implements linear weighted model h_learned of n, g, ⍰ equal to w⍰ times h_geo of n plus w⍰ times h_stress of n plus w⍰ times h_safety of n where weights ⍰ consisting of w⍰, w⍰, w⍰ are learned through **offline training** using **Ridge regression** on over 450 training samples with hyperparameter ⍰ equal to 0.01 controlling regularization strength preventing overfitting, **online adaptation** through **stochastic gradient descent** from user feedback where each accepted or rejected route provides training signal updating weights proportional to preference strength, and normalization constraint ensuring sum of absolute values of weights equals 1.0 maintaining interpretability where individual weights represent proportional contribution of each base heuristic. The training procedure optimizes loss function L of ⍰ equal to sum over training samples i of quantity y_i minus h_learned of n_i, ⍰ squared plus ⍰ times sum of ⍰_j squared where y_i represents observed stress score and regularization term penalizes large weight magnitudes, using coordinate descent alternately updating each weight while holding others fixed until convergence defined by relative change in loss below 0.001, and validates through **leave-one-out cross-validation** where each of 5 real samples serves as test set once with remaining data for training enabling unbiased assessment of generalization despite limited real data.

Success criteria require achieving **training convergence** with loss reduction exceeding 80 percent from initial random weights to converged solution monitored through training curves plotting loss versus iteration, **generalization** achieving **test MAE** below 15 percent of mean target where mean target approximately equals 14.5 yielding acceptable error threshold approximately 2.2 units on stress scale validated through leave-one-out CV on real samples, **improvement versus baseline** demonstrating 15 to 25 percent better user preference match compared to uniform weights measured through simulated user study where 100 virtual users with varied preference profiles choose between algorithm-generated routes, and **weight interpretability** where top 5 features by absolute weight magnitude align with domain knowledge from urban planning literature expecting violation features indexed 0 through 4 to receive highest weights confirmed through rank correlation using Spearman ⍰ exceeding 0.6. Statistical validation employs 5-fold cross-validation partitioning combined synthetic and real dataset into 5 roughly equal folds, training on 4 folds testing on 1 fold rotating through all combinations, reporting mean and standard deviation of performance metrics across folds where standard deviation below 10 percent of mean indicates stable reliable performance, and comparing to random weight baseline assigning uniform random weights from 0 to 1 and uniform weight baseline setting all weights equal to 1 divided by 3 approximately 0.33 requiring learned weights achieve statistically significant improvement with paired t-test p less than 0.05.

Task 3: Dynamic Regularization Implementation

The objective of this task is to design and implement **adaptive weight parameter selection** that dynamically adjusts search intensity based on query characteristics, inspired by empirical results from Programming Problem 3 where weighted A* with varying weight parameters demonstrated predictable trade-offs between solution quality and computational efficiency. The foundational observation from Programming Problem 3 established that **weight parameter W equal to 1.0** using standard A* with admissible heuristic guarantees optimal solutions satisfying cost equals minimum possible cost but expands more nodes typically 500 to 1000 nodes for medium difficulty 8-puzzle instances, **weight W equal to 1.2** identified as sweet spot frequently achieves optimal solutions maintaining same path length while reducing nodes expanded by approximately 30 percent to range 350 to 700 nodes offering best balance for practical applications, and **weight W equal to 1.5** executes substantially faster reducing nodes expanded by approximately 60 percent to range 200 to 400 nodes but accepts solution quality degradation averaging plus 2 steps approximately 10 percent longer paths.

The **adaptive weight selection algorithm** computes weight dynamically based on query context containing path length estimate derived from Euclidean distance between start and goal serving as proxy for problem complexity where longer paths suggest harder problems requiring more aggressive pruning, urgency level taking values low, medium, or high reflecting user tolerance for computation time where high urgency prioritizes speed over optimality, quality threshold specifying minimum acceptable solution quality on scale 0 to 1 where values near 1.0 demand near-optimal solutions while values near 0.8 tolerate moderate suboptimality, and time budget limiting maximum computation in seconds for real-time responsiveness where mobile applications typically require completion within 3 seconds. The algorithm initializes base weight to 1.2 validated as optimal from Programming Problem 3 empirical results, adjusts for urgency adding 0.0 for low urgency maintaining baseline, adding 0.1 for medium urgency accepting slight optimality loss, or adding 0.3 for high urgency prioritizing speed substantially, adjusts for complexity adding 0.1 when path length estimate exceeds 5000 meters indicating long cross-borough routes requiring aggressive pruning or subtracting 0.1 when path length estimate falls below 1000 meters indicating short local routes permitting thorough search, adjusts for quality forcing weight to 1.0 when quality threshold exceeds 0.95 requiring near-optimal solutions or allowing weight up to 2.0 when quality threshold falls below 0.80 permitting substantial suboptimality, and clips final weight between 1.0 and 2.0 ensuring valid range where W equal to 1.0 guarantees optimality for admissible heuristics and W equal to 2.0 provides maximum speedup accepting bounded solution quality degradation.

The framework incorporates **computational budget management** implementing **anytime algorithm** properties through maximum node expansion limit of 10000 nodes preventing pathological cases from consuming excessive resources where poorly designed heuristics could expand entire graph, time cutoff of 3.0 seconds meeting real-time constraints for mobile pedestrian applications requiring immediate feedback, progressive relaxation increasing W incrementally by 0.1 if no solution found within budget allowing algorithm to adaptively relax quality requirements when facing unexpectedly difficult instances, and best-so-far tracking maintaining current best solution returning it when budget exhausted ensuring users always receive actionable results even under strict resource constraints. Success criteria require **speed improvement** achieving 30 to 60 percent fewer nodes versus W equal to 1.0 baseline measured by node expansion count, **quality preservation** maintaining over 90 percent of optimal quality computed as ratio of algorithm cost to Dijkstra optimal cost, **response time** under 3 seconds for 95 percent of queries measured by timing distribution with 5 runs per query reporting median latency, and **weight correlation** exceeding Pearson r greater than 0.7 between selected W and query complexity measured by optimal solution depth validating that algorithm appropriately increases weights for harder problems requiring more aggressive pruning.

Task 4: Empirical Evaluation and Validation

The objective of this task is to conduct rigorous comparative analysis of the proposed **adaptive heuristic system** against established baseline algorithms across 4 evaluation dimensions including solution quality, computational efficiency, optimality analysis, and user preference matching. We implement 6 algorithms for comprehensive comparison. **Dijkstra's algorithm** serves as optimal baseline guaranteeing shortest paths through expansion of all reachable nodes in increasing distance order without using heuristics, providing ground truth for optimality gap calculations. **A* with Euclidean distance** implements classic informed search using admissible straight-line heuristic guaranteeing optimality while reducing nodes expanded compared to Dijkstra, establishing baseline for heuristic effectiveness. **A* with Manhattan distance** provides alternative admissible baseline potentially more accurate in grid-like street networks where paths constrain to horizontal and vertical movements. **Weighted A* with W equal to 1.2** implements best configuration from Programming Problem 3 achieving approximately 30 percent reduction in nodes expanded while usually maintaining optimality, serving as strong practical baseline. **Greedy best-first search** uses only heuristic ignoring path cost through f of n equal to h of n offering very fast execution but no optimality guarantee, establishing lower bound on search efficiency. **Adaptive heuristic A*** represents proposed system combining learned weighted combination of base heuristics with dynamic W adjustment and path sampling for diversity, targeting superior performance across all evaluation dimensions.

The **experimental design** constructs test sets through stratified sampling ensuring diverse difficulty coverage. We generate 25 real routing scenarios from NYC manually designed by domain experts familiar with city geography ensuring practical relevance, selecting start-goal pairs spanning different boroughs testing cross-borough routing, varying difficulty from short local trips to long traversals requiring optimal path discovery across complex network, and incorporating known stress factors including high-violation zones and crowded commercial districts. We generate 75 synthetic scenarios through random start-goal pairs uniformly sampled from 907 zones providing unbiased coverage, stratified by difficulty with 25 easy routes with Euclidean distance under 2 kilometers expecting solutions within 5 to 10 hops, 25 medium routes from 2 to 5 kilometers expecting 10 to 20 hops, and 25 hard routes exceeding 5 kilometers expecting over 20 hops, and replicated 5 times per scenario reporting median performance and interquartile range controlling for stochastic variation. Total experimental load equals 6 algorithms times 100 scenarios times 5 replications equals 3000 runs requiring approximately 8 to 12 hours computation time on modern multicore processor.

Performance measurement collects **solution quality** through normalized weighted combination Q of path equal to w_d times normalized distance plus w_s times normalized stress plus w_v times normalized violations where normalization divides each component by maximum value in dataset ensuring fair comparison, user preference weights set to w_d equal to 0.3, w_s equal to 0.5, w_v equal to 0.2 reflecting stress-prioritized preferences derived from urban planning surveys, and comparison to Dijkstra's optimal for distance component enabling isolation of stress and safety contributions. **Computational efficiency** metrics include nodes expanded counting how many states algorithm examined serving as primary efficiency indicator independent of implementation details, CPU time measured by wall-clock timing averaged over 5 runs controlling for system variability, memory usage tracking peak consumption during search identifying algorithms requiring excessive space, and scalability measuring performance versus graph size testing on subgraphs containing 100, 500, and 907 nodes. **Optimality analysis** computes optimality gap as quantity cost_algorithm minus cost_optimal divided by cost_optimal times 100 percent measured for distance objective where Dijkstra provides optimal baseline and acceptable gap defined as less than 10 percent for practical applications where users tolerate small optimality loss for substantially improved efficiency. **User preference matching** simulates user choices where 80 percent of simulated users prefer lower stress over shorter distance, ranks algorithm solutions by simulated preference using utility functions parameterized by user type, and measures rank correlation using Spearman's ⍰ quantifying how well algorithm rankings match user preference rankings.

Statistical analysis employs **paired t-tests** comparing each algorithm to proposed system on same scenarios enabling within-subject design controlling for scenario difficulty, **one-way ANOVA** across all 6 algorithms testing null hypothesis that all perform equivalently with post-hoc Tukey HSD identifying which pairs differ significantly, **effect size** using Cohen's d measuring practical significance where d greater than 0.8 indicates large effect regardless of statistical significance, significance level ⍰ equal to 0.05 controlling Type I error rate, and **Bonferroni correction** for multiple comparisons dividing ⍰ by number of comparisons equal to 5 yielding adjusted threshold 0.01 preventing false discoveries from repeated testing. Success requires achieving statistical significance p less than 0.05 for at least 3 of 4 metrics when comparing proposed system to best baseline, demonstrating both statistical and practical significance through effect sizes exceeding 0.5 for key metrics, and maintaining consistency across difficulty levels where performance advantages hold for easy, medium, and hard scenarios preventing overfitting to specific problem characteristics.

Work Plan and Student Information

Student Information

Name: Gil Raitses
Email: graitses@syr.edu
Phone: (Available upon request)
Student Status: Individual project
Collaboration: This project will be completed solely by Gil Raitses with guidance from Prof. Andrew C. Lee

Detailed Timeline

The project follows structured 10-week timeline organized into 5 major phases with specific milestones enabling progress tracking and risk mitigation. **Phase 1 spanning weeks 1 through 2 from November 1 through November 14 addresses foundation and system design** conducting literature review surveying papers on heuristic learning including work by Samadi et al. on combining multiple heuristics, weighted A* including Pohl's foundational analysis and Thayer and Ruml's bounded suboptimal search, and path sampling including Kocsis and Szepesvári's bandit-based Monte Carlo planning, finalizing system architecture with complete algorithm specifications documented in formal pseudocode following AIMA textbook conventions, setting up development environment installing Python version 3.9 or later with dependencies NumPy version 1.21 or later for numerical operations, pandas version 1.3 or later for data manipulation, scikit-learn version 1.0 or later for machine learning, NetworkX version 2.6 or later for graph algorithms, and configuring Git version control with repository initialization and proper gitignore, and delivering design document with algorithm pseudocode, data structure specifications, and evaluation methodology ready for instructor review at Checkpoint 1 scheduled for November 14.

Phase 2 spanning weeks 3 **through** 4 **from November** 15 **through November** 28 **implements core algorithms** implementing path sampling framework with all 3 strategies including uniform random walk, importance-weighted sampling, and Thompson sampling with unit tests validating each produces diverse paths, implementing base heuristics covering geometric using Euclidean distance, stress-based using zone stress scores, and safety-based using violation density with validation against hand-computed examples, implementing baseline algorithms including Dijkstra for optimal baseline, A* with Euclidean and Manhattan heuristics for admissible baselines, weighted A* with W equal to 1.2 for suboptimal fast baseline, and greedy best-first for inadmissible fast baseline with correctness validation through comparison to known solutions on small graphs, and delivering working prototype with all baseline algorithms demonstrated through execution on sample scenarios with recorded performance metrics ready for demonstration at Checkpoint 2 scheduled for November 28.

Phase 3 spanning weeks 5 **through** 6 **from November** 29 **through December** 12 **develops learning and adaptation** executing data augmentation generating 450 synthetic training samples through bootstrap resampling producing 100 samples with noise injection standard deviation 0.15, SMOTE producing 150 samples with k_neighbors equal to 2, and parametric generation producing 200 samples from fitted Gaussian distributions, training models using **Ridge regression** with hyperparameter tuning via grid search over ⍰ values including 0.001, 0.01, 0.1, and 1.0 selecting value minimizing cross-validation error, implementing adaptive regularization computing dynamic W based on query context including path length estimate, urgency level, and quality threshold with validation through synthetic query generation, integrating learned weights into search algorithms connecting trained scikit-learn model to NetworkX graph search implementations through weight extraction and heuristic function construction, and delivering complete adaptive heuristic system with end-to-end execution from query input through weight selection, heuristic computation, and path discovery with performance validation meeting targets from Objectives 1 through 3 ready for validation at Checkpoint 3 scheduled for December 12.

Phase 4 spanning weeks 7 **through** 8 **from December** 13 **through December** 26 **executes comprehensive experimentation** running experimental suite executing 3000 total runs computed as 6 algorithms times 100 scenarios times 5 replications with parallel execution across available CPU cores, collecting performance data gathering all metrics including nodes expanded, CPU time in seconds, memory in megabytes, path length in meters, path stress as aggregate score, path safety as violation density, and optimality gap as percentage difference from Dijkstra optimum, conducting statistical analysis computing paired t-tests for each algorithm pair, one-way ANOVA across all algorithms, effect sizes using Cohen's d formula, and Bonferroni correction for 5 pairwise comparisons against proposed system, performing computational profiling using cProfile identifying hotspots in priority queue operations, heuristic computation, and path reconstruction with optimization targeting bottlenecks consuming over 10 percent total execution time, and delivering complete experimental results dataset with raw data in CSV format, statistical test results with p-values and effect sizes, and profiling reports identifying optimization opportunities ready for final analysis.

Phase 5 spanning weeks 9 **through** 10 **from December** 27 **through January** 9 **completes analysis and reporting** conducting theoretical analysis proving admissibility conditions for learned heuristics deriving when weighted combinations remain admissible, deriving optimality bounds establishing when weighted A* guarantees bounded solutions, and characterizing computational complexity through Big-O analysis and empirical power law fitting, generating result visualizations creating performance comparison charts using matplotlib showing nodes expanded versus graph size, solution quality distributions using box plots across algorithms, and optimality gap scatter plots against query difficulty, creating tables summarizing statistical test results, mean performance metrics with standard deviations, and success criteria achievement across all 8 objectives, writing final technical report of 15 to 20 pages following scientific writing principles with clear narrative structure, abstract summarizing research question and impact, introduction establishing motivation and background, methods describing algorithms and evaluation, results presenting findings with visualizations, discussion interpreting results and limitations, and conclusion summarizing contributions, documenting code adding comprehensive docstrings to all functions, type hints for function signatures, README with installation and usage instructions, and example scripts demonstrating basic usage, and preparing presentation creating slides following scientific presentation conventions with clear motivation, methods overview, results highlights, and conclusions with broader impacts, and delivering final submission on January 9 including PDF report, code repository with documentation, and presentation materials.

Risk Management and Mitigation Strategies

Potential challenges with corresponding mitigation strategies include heuristic learning may not improve performance where risk involves learned heuristics performing worse than static baselines mitigated by starting with proven weighted A* approach from Programming Problem 3 ensuring floor on performance and implementing incremental improvements adding learning gradually to validated baseline, computational complexity too high where risk involves path sampling too expensive for real-time use mitigated by implementing aggressive pruning using beam search limiting candidates, precomputing route templates for common origin-destination pairs enabling lookup, and fallback to faster algorithms when budget exhausted maintaining responsiveness, limited training data where risk involves 5 records insufficient for ML training mitigated by generating synthetic training data through multiple augmentation techniques, using transfer learning initializing weights from domain knowledge, and simulating user preferences through parameterized utility functions, and time constraints where risk involves cannot complete all objectives in 10 weeks mitigated by prioritizing core objectives 1 through 3 as essential requirements, treating advanced features including neural network heuristics as stretch goals, and implementing modular architecture enabling incremental development where each component functions independently.

Intellectual Merit and Broader Impacts

Contributions to Artificial Intelligence

This research makes 3 primary contributions advancing the field of artificial intelligence beyond existing state of knowledge. First, the **methodological innovation** integrates path sampling, machine learning, and adaptive search in unified framework addressing fundamental limitation that classical search algorithms like A* converge to single solution without exploring alternative trade-offs, machine learning applications typically treat search as black box optimizing end-to-end without leveraging search structure, and multi-objective optimization methods often use evolutionary algorithms requiring thousands of evaluations whereas our approach leverages problem structure through informed heuristics. The integration enables discovery of diverse Pareto-optimal solutions through systematic sampling representing different trade-offs, learning from limited data through hybrid augmentation producing statistically valid training sets from small real samples, and maintaining computational efficiency through adaptive regularization preventing excessive search in easy cases while allocating resources to hard instances. Second, the **theoretical advancement** provides formal characterization of learned heuristic properties proving conditions under which weighted combinations remain admissible enabling optimality guarantees, deriving bounds on solution quality for inadmissible learned heuristics establishing worst-case performance, and characterizing computational complexity through empirical power law analysis demonstrating how nodes expanded scales with network size and heuristic accuracy. Third, the **empirical validation** using real-world urban navigation data demonstrates practical applicability beyond synthetic benchmarks where most academic research validates on classical planning domains like sliding tile puzzles, establishing that learned heuristics improve user preference match by 15 to 25 percent measured through simulated user studies, and confirming real-time feasibility with response times under 3 seconds on 907-node network enabling deployment in mobile applications.

The research advances understanding of fundamental trade-offs in informed search algorithms where admissible heuristics guarantee optimality but may expand excessive nodes especially when heuristics provide weak guidance, inadmissible heuristics offer stronger guidance potentially reducing search effort substantially but sacrifice optimality guarantees requiring careful empirical validation, and adaptive approaches selecting between strategies based on instance characteristics can achieve best of both worlds identifying when optimality guarantees are necessary versus when bounded suboptimality suffices. This understanding generalizes beyond urban routing to any domain requiring real-time decision-making under resource constraints including robot path planning where computational budgets limit onboard processing, network routing where packet forwarding decisions must complete within microseconds, and game playing where move selection must finish before time control expires. The framework for learning from limited data through sophisticated augmentation techniques including bootstrap resampling, SMOTE, and parametric generation provides methodology applicable to any machine learning application facing data scarcity, demonstrating that 5 real samples can produce 455-sample training set enabling statistically valid model training when augmentation preserves key distributional properties validated through held-out real samples.

Applications and Societal Benefits

The developed framework directly benefits urban pedestrians through improved routing quality enabling avoidance of high-violation zones reducing exposure to dangerous sidewalk cycling where cyclists travel at speeds exceeding 15 miles per hour on pedestrian walkways designed for walking speeds of 3 to 4 miles per hour, reducing exposure to red light violations where preliminary data shows 87 percent confidence detection of bike red light violations creating pedestrian safety hazards, and reducing stress from infrastructure deficiencies including missing barriers separating pedestrian and vehicle traffic and poor signage causing navigation uncertainty. The system provides particular benefits for vulnerable populations including elderly individuals who experience disproportionate stress from crowding and fast-moving cyclists, disabled individuals requiring accessible routes with proper infrastructure including curb cuts and tactile paving, and parents with children who prioritize safety over speed accepting substantially longer routes to avoid high-risk areas. These populations currently lack navigation tools addressing their specific needs beyond wheelchair accessibility in conventional mapping systems.

The research provides actionable insights for urban planning through data-driven identification of persistent problem areas where violation density consistently exceeds acceptable thresholds indicating systemic infrastructure or enforcement failures, quantification of stress factors enabling cost-benefit analysis for infrastructure investments showing that barrier installation reducing stress by 2 units benefits 5000 daily pedestrians, and validation of intervention effectiveness measuring before-after stress changes when improvements implemented providing evidence base for continued investment. The methodology transfers to related optimization domains including emergency response routing where first responders require multi-objective paths balancing response time against route safety especially important for medical transports prioritizing smooth routes minimizing patient discomfort, evacuation planning where large populations must evacuate efficiently while avoiding bottlenecks that could cause dangerous crowding and maintaining orderly flow preventing panic, and logistics optimization where delivery routing must consider multiple factors including fuel efficiency, delivery time windows, and driver safety rather than pure distance minimization.

The open-source implementation using standard Python libraries including NumPy, pandas, scikit-learn, and NetworkX ensures broad accessibility through elimination of commercial software dependencies enabling deployment without expensive licenses, facilitation of reproducibility allowing other researchers to validate results and extend methodology without recreating infrastructure, and support for education integration where students can run experiments on personal computers without requiring cloud access. This accessibility particularly benefits researchers and practitioners in resource-constrained settings including developing countries where cloud infrastructure costs may prohibit adoption, small municipalities lacking technical staff to manage cloud deployments, and educational institutions with limited budgets seeking to incorporate real-world applications into coursework without incurring ongoing operational costs.

Appendix A: Technical Implementation Details

Algorithm Specifications

The **adaptive heuristic A* algorithm** extends classical A* search integrating learned weight parameters and dynamic regularization. The algorithm receives input parameters start_node identifying origin zone, goal_node identifying destination zone, learned_weights as array containing w_geo, w_stress, and w_safety, and query_context structure containing path_length_estimate as float in meters, urgency_level as string with values low, medium, or high, quality_threshold as float from 0 to 1, and time_budget as float in seconds. Initialization computes adaptive_weight by calling compute_adaptive_weight function on query_context returning float W in range 1.0 to 2.0, sets nodes_expanded counter to 0, records start_time as current timestamp, initializes priority_queue as min-heap containing tuple of f_value and start_node where f_value equals

0 plus adaptive_weight times learned_heuristic of start_node computed using learned_weights, initializes explored_set as empty set for tracking visited nodes, initializes came_from as empty dictionary mapping nodes to predecessors for path reconstruction, and initializes g_values as dictionary mapping nodes to actual costs with start_node mapped to 0.

The main search loop continues while priority_queue not empty and nodes_expanded less than 10000 and current time minus start_time less than time_budget. Each iteration extracts current_node and current_f from priority_queue using heappop operation with O of log n complexity, checks termination returning reconstruct_path of current_node using came_from if current_node equals goal_node, checks duplication continuing to next iteration if current_node in explored_set indicating already processed, adds current_node to explored_set, increments nodes_expanded counter, and processes neighbors iterating over each neighbor in graph.neighbors of current_node. For each neighbor, the algorithm skips if neighbor in explored_set, computes tentative_g as g_values of current_node plus edge_cost from current_node to neighbor, checks improvement where if neighbor not in g_values or tentative_g less than g_values of neighbor indicating better path found, it updates came_from with neighbor mapping to current_node, updates g_values with neighbor mapping to tentative_g, computes h_value as adaptive_weight times learned_heuristic of neighbor computed using learned_weights, computes f_value as tentative_g plus h_value, and pushes tuple of f_value and neighbor to priority_queue using heappush. If loop terminates without finding goal, algorithm returns None indicating no path exists within computational budget.

The **learned heuristic function** computes weighted combination of base heuristics receiving node structure, goal node structure, and learned weights array containing w geo, w stress, and w safety. The function extracts node position from node.position and goal position from goal node.position, computes h geo as square root of quantity node position x minus goal position x squared plus node position y minus goal position y squared providing admissible lower bound, computes h stress as node.stress score plus estimated path stress from node to goal node using linear interpolation based on distance where estimated path stress approximately equals stress score times remaining distance divided by total distance providing rough estimate, computes h safety as node.violation count plus estimated violations from node to goal node using zone violation density from BigQuery statistics, and returns w geo times h geo plus w stress times h stress plus w safety times h safety as final weighted combination.

Data Augmentation Implementations

The **bootstrap with noise function** implements resampling augmentation receiving X as NumPy array with shape n samples by 17 representing feature matrix, y as NumPy array with shape n samples representing target stress scores, n samples as integer defaulting to 100 specifying augmentation size, and noise std as float defaulting to 0.15 controlling noise magnitude. The function imports resample from sklearn.utils, executes X boot and y boot equals resample of X and y with n samples parameter and replace equals True enabling repeated selection of same original samples, generates noise as np.random.normal of mean 0 and standard deviation noise std and shape X boot.shape producing Gaussian perturbations, computes X boot as np.clip of X boot plus noise with bounds 0 and 3 enforcing valid feature range, and returns tuple of X boot and y boot.

The **SMOTE augmentation function** implements intelligent interpolation receiving X and y and n samples defaulting to 150. The function imports SMOTE from imblearn.over sampling, initializes smote object with parameters sampling strategy equals auto automatically determining minority class and k neighbors equals 2 using 2 nearest neighbors for interpolation given limited original samples, executes X augmented and y augmented equals smote.fit resample of X and y performing interpolation along line segments connecting neighbors, and returns X augmented and y augmented as augmented dataset.

The **parametric generation function** implements distribution-based sampling receiving real samples as array and n synthetic defaulting to 200. The function computes mu as np.mean of real samples along axis 0 and sigma as np.cov of real samples transposed computing multivariate Gaussian parameters, samples synthetic as np.random.multivariate normal of mu and sigma and n synthetic drawing from fitted distribution, clips synthetic as np.clip of synthetic with bounds 0 and 3 ensuring valid range, and returns synthetic samples as NumPy array.

Evaluation Metrics Computations

The **solution quality score function** computes weighted multi-objective quality receiving path object, w d defaulting to 0.3 for distance weight, w s defaulting to 0.5 for stress weight, and w v defaulting to 0.2 for violation weight. The function extracts path.distance as total geometric distance in meters, path.stress as aggregate stress computed by summing zone stress scores along path, and path.violations as total violation count summing violations across zones, computes normalized distance as path.distance divided by dataset max distance where dataset max distance equals maximum distance across all candidate paths in current evaluation, computes normalized stress as path.stress divided by dataset max stress ensuring components on comparable scale, computes normalized violations as path.violations divided by dataset max violations, and returns w d times normalized distance plus w s times normalized stress plus w v times normalized violations as quality score where lower values indicate better quality.

The **optimality gap function** measures deviation from optimal receiving algorithm cost as float and optimal cost as float. The function returns quantity algorithm cost minus optimal cost divided by optimal cost times 100 as float expressing gap as percentage where 0 indicates optimal, 10 indicates 10 percent longer than optimal, and negative values impossible given optimal cost represents true minimum.

The **compute statistical significance function** performs hypothesis testing receiving results proposed as NumPy array containing performance metric for proposed algorithm across scenarios and results baseline as NumPy array containing same metric for baseline algorithm on same scenarios. The function imports scipy.stats module, extracts paired differences as results proposed minus results baseline element-wise, executes t statistic and p value equals scipy.stats.ttest rel of results proposed and results baseline performing paired t-test, computes effect size as np.mean of paired differences divided by np.std of paired differences computing Cohen's d, constructs result dict containing p value, t statistic, effect size, and significant as boolean where significant equals p value less than 0.05, and returns result dict as dictionary.

Success Criteria Summary

Table 1 summarizes success criteria across all 8 project objectives with clearly defined metrics, quantitative targets, and validation methods enabling objective assessment of achievement.

Table 1: Success criteria for project objectives

Objective	Primary Metric	Target	Validation Method
Path Sampling	Sample diversity	>70% unique	Jaccard distance
Heuristic Learning	Test MAE	<15% of mean	Leave-one-out CV
Adaptive Regularization	Speed improvement	30-60% fewer nodes	Node count
Data Integration	Graph construction	907 nodes connected	NetworkX validation
Algorithm Comparison	Statistical significance	$p < 0.05$	Paired t-test
Real-World Validation	User preference match	>75% agreement	Simulated study
Theoretical Analysis	Proof completeness	All theorems proved	Peer review
Scalability	Scaling exponent	⍰ < 0.8	Power law fit $R^2 > 0.85$

Overall project success requires meeting targets for at least 6 out of 8 objectives, establishing comprehensive validation through multiple independent criteria ensuring robust evidence for methodology effectiveness while acknowledging that stretch goals including neural network heuristics and online learning may remain partially completed within 10-week timeframe.

Conclusion

This proposal presents comprehensive plan for developing and validating **adaptive heuristic learning framework** for **stress-optimized pedestrian navigation** that advances artificial intelligence through integration of classical search algorithms with modern machine learning addressing fundamental challenges in real-time multi-objective optimization. The research makes three key contributions including methodological innovation combining path sampling, heuristic learning, and adaptive regularization in unified framework extending classical A* algorithm with data-driven adaptation, practical application demonstrating real-world system performance using NYC camera network data with 907 zones providing authentic validation beyond synthetic benchmarks common in academic research, and theoretical rigor through formal analysis of algorithm properties including admissibility, optimality bounds, and computational complexity connecting empirical results to established search theory. The work directly applies and extends concepts from CIS 667 particularly informed search from Chapter 3 and learning agents from Chapter 21 while addressing practical urban planning problem with implications for accessible cities, emergency response, and human-centered transportation systems.

The comprehensive evaluation framework with 8 well-defined objectives ensures rigorous validation through quantitative metrics including 30 to 60 percent reduction in nodes expanded, statistical significance testing using paired t-tests and ANOVA with p less than 0.05, and real-world validation through simulated user studies achieving over 75 percent preference match, producing meaningful results advancing both algorithmic understanding and practical deployment capabilities. The data robustness strategy addresses limited real-world data consisting of 5 records through sophisticated augmentation pipeline combining bootstrap resampling generating 100 samples, SMOTE generating 150 samples, and parametric generation producing 200 samples, validated through statistical cross-validation using 5-fold CV and leave-one-out CV on real samples exclusively, and confirmed through domain knowledge alignment ensuring learned weights match urban planning principles. This comprehensive approach creates robust training set of 455 samples enabling statistically valid model training despite initial data scarcity, demonstrating general methodology for machine learning with limited real-world data applicable across domains where data collection proves expensive or time-consuming.

The proposed work remains feasible within 10-week timeline for individual student project with clear milestones including design document review on November 14, prototype demonstration on November 28, learning system validation on December 12, and final submission on January 9, and well-defined deliverables at each stage including documented algorithms, working code, experimental results, and comprehensive technical report. The project scope balances ambition with practicality by focusing on core 6 objectives as essential requirements including path sampling, heuristic learning, adaptive regularization, algorithm comparison, real-world validation, and theoretical analysis, while treating advanced features including neural network heuristics and online learning as stretch goals, ensuring successful completion while maintaining opportunities for exceptional performance if time and resources permit.

Student Signature: *Gil Raitses*

Date: October 27, 2025

Appendix A: Mathematical Formulations

This appendix presents the mathematical formulations underlying the adaptive heuristic learning framework. We provide formal definitions for key equations referenced throughout the proposal, including the adaptive weight selection algorithm (Eq. 1), solution quality scoring function (Eq. 2), optimality gap computation (Eq. 3), and theoretical results on admissibility and consistency (Theorems 1-2).

Equation 1: Adaptive Weight Selection

The adaptive weight W is computed dynamically based on query context to balance solution quality against computational efficiency:

$$W = \text{clip}(W_{\text{base}} + \Delta_{\text{urgency}} + \Delta_{\text{complexity}} + \Delta_{\text{quality}}, 1.0, 2.0)$$

where $W_{\text{base}} = 1.2$ (validated empirically), $\Delta_{\text{urgency}} \in \{0.0, 0.1, 0.3\}$ for low, medium, high urgency, $\Delta_{\text{complexity}} = +0.1$ if distance > 5000m or -0.1 if distance < 1000m, and Δ_{quality} forces W = 1.0 if threshold > 0.95.

Equation 2: Solution Quality Score

The multi-objective quality score combines normalized distance, stress, and violations:

$$Q(\text{path}) = w_d \cdot \frac{d(\text{path})}{d_{\text{max}}} + w_s \cdot \frac{s(\text{path})}{s_{\text{max}}} + w_v \cdot \frac{v(\text{path})}{v_{\text{max}}}$$

where $w_d = 0.3$, $w_s = 0.5$, $w_v = 0.2$ represent user preference weights for distance, stress, and violations respectively, and the denominators normalize to [0,1] scale.

Equation 3: Optimality Gap

The optimality gap measures deviation from optimal solution as percentage:

$$\text{gap} = \frac{\text{cost}_{\text{algorithm}} - \text{cost}_{\text{optimal}}}{\text{cost}_{\text{optimal}}} \times 100\%$$

where $\text{cost}_{\text{optimal}}$ is obtained from Dijkstra's algorithm for the distance objective.

Theorem 1: Admissibility of Weighted Heuristic Combinations

A learned heuristic $h_{\text{learned}} = \sum w_i \cdot h_i$ remains admissible if and only if all base heuristics h_i are admissible and all weights $w_i \geq 0$. Since our learned heuristic includes inadmissible components (stress and safety heuristics), the combined heuristic is inadmissible.

Theorem 2: Consistency of Weighted Heuristic Combinations

If all base heuristics are consistent (i.e., $h(n) \leq c(n, a, n') + h(n')$ for all edges), then their positive weighted combination remains consistent. Consistency ensures each node is expanded at most once, improving search efficiency.

Appendix B: Algorithm Pseudocode

This appendix provides detailed pseudocode descriptions for the core algorithms in the adaptive heuristic learning framework. These algorithmic specifications support implementation and provide clear reference for understanding the computational procedures described in the main proposal.

Adaptive Heuristic A* Algorithm

The adaptive heuristic A* algorithm extends classical A* search integrating learned weight parameters and dynamic regularization. The algorithm receives input parameters (start_node, goal_node, learned_weights, query_context) and implements the search procedure using dynamic weight adjustment based on Equation 1. The main loop continues while priority queue not empty and computational budget not exhausted, extracting nodes, checking goal condition, and processing neighbors with updated f-values computed using the learned heuristic function.

Learned Heuristic Function

The learned heuristic function combines three base heuristics with learned weights. The function computes h_geo as Euclidean distance providing admissible lower bound, h_stress estimating aggregate stress along remaining path, h_safety estimating violations using zone violation density, and returns the weighted combination w_geo · h_geo + w_stress · h_stress + w_safety · h_safety.

Appendix C: Implementation Code

This appendix provides Python implementation code for key data processing and evaluation components. These code listings demonstrate the practical realization of the algorithms and methods described in the proposal, supporting reproducibility and implementation clarity.

Bootstrap Resampling with Noise Injection

```python
def bootstrap_with_noise(X, y, n_samples=100, noise_std=0.15):
    """Generate bootstrap samples with additive noise"""
    from sklearn.utils import resample
    X_boot, y_boot = resample(X, y, n_samples=n_samples, replace=True)
    noise = numpy.random.normal(0, noise_std, X_boot.shape)
    X_boot = numpy.clip(X_boot + noise, 0, 3)
    return X_boot, y_boot
```

SMOTE Augmentation

```python
def smote_augmentation(X, y, n_samples=150):
    """Generate SMOTE samples through intelligent interpolation"""
    from imblearn.over_sampling import SMOTE
    smote = SMOTE(sampling_strategy='auto', k_neighbors=2)
    X_augmented, y_augmented = smote.fit_resample(X, y)
    return X_augmented, y_augmented
```

Parametric Synthetic Generation

```python
def parametric_generation(real_samples, n_synthetic=200):
    """Generate parametric samples from fitted Gaussian"""
    mu = numpy.mean(real_samples, axis=0)
    sigma = numpy.cov(real_samples, rowvar=False)
    synthetic = numpy.random.multivariate_normal(mu, sigma, n_synthetic)
    synthetic = numpy.clip(synthetic, 0, 3)
    return synthetic
```

Solution Quality Scoring

```
def solution_quality_score(path, w_d=0.3, w_s=0.5, w_v=0.2,
                           max_distance=10000, max_stress=30, max_violations=100):
    """Compute weighted quality score implementing Equation 2"""
    normalized_distance = path.distance / max_distance
    normalized_stress = path.stress / max_stress
    normalized_violations = path.violations / max_violations
    return (w_d * normalized_distance +
            w_s * normalized_stress +
            w_v * normalized_violations)
```

Optimality Gap Computation

```
def optimality_gap(algorithm_cost, optimal_cost):
    """Calculate optimality gap implementing Equation 3"""
    return ((algorithm_cost - optimal_cost) / optimal_cost) * 100
```

Statistical Significance Testing

```
def compute_statistical_significance(results_proposed, results_baseline):
    """Compute paired t-test and effect size"""
    import scipy.stats
    paired_differences = results_proposed - results_baseline
    t_statistic, p_value = scipy.stats.ttest_rel(results_proposed, results_baseline)
    effect_size = numpy.mean(paired_differences) / numpy.std(paired_differences)
    return {
        'p_value': p_value,
        't_statistic': t_statistic,
        'effect_size': effect_size,
        'significant': p_value < 0.05
    }
```

References

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321-357.

Hansen, E. A., & Zhou, R. (2007). Anytime heuristic search. *Journal of Artificial Intelligence Research*, 28, 267-297.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100-107.

Kocsis, L., & Szepesvári, C. (2006). Bandit based monte-carlo planning. *European Conference on Machine Learning*, 282-293.

Pearl, J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

Pohl, I. (1970). Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3-4), 193-204.

Quercia, D., Schifanella, R., & Aiello, L. M. (2014). The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city. *ACM HyperText*, 116-125.

Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.

Samadi, M., Felner, A., & Schaeffer, J. (2008). Learning from multiple heuristics. *AAAI*, 8, 357-362.

Thayer, J. T., & Ruml, W. (2011). Bounded suboptimal search: A direct approach using inadmissible estimates. *IJCAI*, 11, 674-679.