Neural Heuristics for Route Optimization in Service of a Search and Rescue Artificial Social Intelligence Agent

Yunzhe Wang,[1] **Nikolos Gurney,**[3]
Jincheng Zhou,[2, 3] **David V. Pynadath,**[3] **Volkan Ustun**[3]
[1]USC Dornsife College of Letters, Arts and Sciences
[2]USC Viterbi School of Engineering, Computer Science Department
[3]USC Institute for Creative Technologies
gurney@ict.usc.edu, ustun@ict.usc.edu

Abstract

The success and safety of a Search and Rescue (SAR) team hinge on routing, making it an integral part of any SAR mission. Consequently, an Artificial Social Intelligence (ASI) agent aware of the "good" available routes in a mission is a very desirable asset for a SAR team. Such awareness is contingent on having superior knowledge of the environment and understanding the dynamics of the SAR team. An ASI agent equipped with this capability can utilize it while reasoning about the mission, similar to how a human may use real-time GPS route suggestions from a navigation application. This feature was historically infeasible for real-time ASI agents because the problems were computationally intractable. However, recent advances in Graph Neural Networks, transformers, and attention models make them candidates to be leveraged as neural heuristics in routing problems to quickly generate near-optimal routes. This paper describes a sequential decision framework based on neural heuristics to devise such routes for participants in the DARPA ASIST Minecraft SAR Task and reports our initial findings.

Introduction

The success and safety of a SAR team hinge on routing, making access to a trusted advisor an indispensable asset. Although many factors impact trust in advisors, expertise is fundamental (French, Raven, and Cartwright 1959; Bonaccio and Dalal 2006). Thus, a SAR ASI agent must have demonstrable expertise in the real-time routing of teams.

A trusted SAR ASI that is equipped with routing technology is similar to a driver with a real-time GPS navigation application. The recommendations that an app provides are based on critical information which it can easily access and process, such as road conditions, traffic jams, accidents, etc. Obviously, this is not something a driver can readily do while traveling—but drivers are also privy to their own private information, such as preferences and physiological states. A driver with a car full of hungry kids, for example, may choose to stop and eat dinner early after the app recommends a re-route rather than following the recommendation. Critically, this decision does not make the re-routing recommendation any less valuable. The driver *trusted* the app, but used private information to make a personal and rational decision. Just like the app performs real-time path planning and provides options to the user based on the current information state, the SAR ASI agent ingests routing suggestions and combines them with other information that it has for the mission and the SAR team it is assisting. Then, the SAR ASI agent makes informed recommendations to the SAR team.

SAR missions burden rescuers with continual decisions about how to traverse an environment. These decision are frequently made under extreme conditions and often from very limited information, both of which place significant stress on humans and impact their decision making capacity (Starcke and Brand 2016). One of these stressers is enough to warrant technological help, but more troublesome is the NP Hard nature of routing in SAR missions. SAR routing is akin to the family of Traveling Salesman Problems (TSPs) which are also NP Hard and studied extensively in Operations Research. However, the most advanced Operations Research methodologies cannot generate real-time guaranteed optimal solutions for TSP problems (save for small networks). There are a number of heuristics available to find candidate solutions (Boussaïd, Lepagnot, and Siarry 2013), and many practical applications utilize such heuristics.

An insight from studying TSPs is that the problem instances often share key characteristics or patterns. Cappart et al. (2021) illustrate this with a trucking company routing problem in which the company needs to generate daily routes for the same city, but with slight differences due to traffic conditions. Such similarities bring opportunities for the data-dependent machine learning approaches that exploit these patterns (Bengio, Lodi, and Prouvost 2021). Recently, Graph Neural Networks (GNNs) with attention mechanisms have become strong heuristic alternatives for combinatorial optimization problems (Cappart et al. 2021). We leverage such an approach as a neural heuristic to quickly generate good paths that exploit the similarities in routing requirements of a Minecraft SAR task. A real-time ASI agent prototype can utilize this capability to explore the routing options available to the SAR team under different conditions.

Background

Combinatorial optimization (CO) is a well-established interdisciplinary area with many critical real-world applications

including routing (Korte and Vygen 2012). For a given problem, CO tries to optimize a cost (or objective) function by selecting a subset from a finite set which is subject to constraints on the selection. The goal of CO is to find a unique, optimal solution to each problem. This is not always practical, however, due to the complexity of problems. Practitioners turn to problem-specific heuristic methodologies in such instances (Boussaïd, Lepagnot, and Siarry 2013).

Graph Neural Networks (GNNs) are a powerful machine learning architecture that leverages structural, relational, and compositional biases to facilitate geometric deep learning (Gilmer et al. 2017). GNNs aggregate information into simpler representations of nodes and edges from structural and feature-based (e.g. node or edge type) graph data. By parameterizing this aggregation, they can be trained in an end-to-end fashion against a loss function. What differentiates GNNs from their predecessors, convolutional and recurrent neural networks, is the ability to operate on higher complexity data than what can be represented in regular Euclidean structures, e.g., a picture (2D) or text (1D). GNNs accomplish this by being order-invariant—they propagate on each node in the graph independently and ignore the input order—and by using the graph structure to guide propagation. These innovations empower GNN models to "reason" about a graph, that is, draw general inferences, and then use those inferences to successfully make predictions and classifications (Zhou et al. 2020). Recently, GNNs have also been utilized as neural heuristics to generate solutions for CO problems (Vesselinova et al. 2020). The central promise of GNNs in this role is that the learned vector representations encode crucial graph structures to help solve a CO problem more efficiently (Cappart et al. 2021).

Kool et al. (2018) proposed a transformer-like encoder-decoder architecture based on Graph Attention Networks (Veličković et al. 2017), a well-known GNN architecture with attention, for general routing problems. In this framework, an encoder-decoder neural network is trained on randomly generated routing problems with network sizes similar to a target problem. The training leverages an actor-critic reinforcement learning approach that circumvents the need for the optimal solutions to the training instances. Although the training needs to be done in advance, with a trained model it is possible to quickly generate very good solutions to SAR routing problems. The work depicted in this paper leveraged the Kool et al.'s codebase (2018) (ALSR codebase) and augmented it for our task.

DARPA ASIST Minecraft SAR Task

The DARPA ASIST Minecraft SAR task ("our" task) is a game-based simulated training environment designed with the explicit purpose of developing ASI agents. The task places participants in a Minecraft environment where they attempt to save victims of an urban disaster and earn points for doing so. Victims are either non-critical or critical, the latter of which are worth more points but take a team effort to save. The environment includes risks, which are locations where a player is frozen until a teammate rescues them. There is also rubble, which can be cleared, that may block access to victims. Teams consist of three participants, and each participant can select one of three roles: medical specialist (medic), hazardous material specialist (engineer), and search specialist (transporter). Medics can triage victims and rescue frozen teammates, engineers can clear rubble, and transporters can transport victims. Each role is associated with a tool that expires after so many uses and must be renewed. Further details about the environment are available in the preregistration (Huang et al. 2021).

Solution Approach

Participants in a Minecraft SAR experiment face routing decisions akin to a Vehicle Routing Problem (VRP), which is a type of TSP in the Operations Research literature. A VRP is a combinatorial optimization problem with an objective to identify optimal routes for each vehicle in a delivery fleet. For example, in the Minecraft SAR environment, triaging victims, clearing rubble, deactivating freezing plates, and converging at critical victims are all demand points (customer locations) that the participants need to visit to complete the SAR task. Consequently, VRP models can provide routing decisions for planning a participant's path. Different VRP models can assist in capturing different aspects of the Minecraft SAR task. For example, the capacitated VRP (CVRP) model, which defines vehicle capacities, can model tool durability, whereas the CVRP with Profits (CVRPP) model can assign unique rewards to different demand points.

We address routing for the Minecraft SAR teams in a sequential decision-making framework via defining separate VRP models for each team member. Each team composition, e.g., one medic - two engineers or one medic - one engineer - one transporter, defines a different sequence of VRP models. Suggested routes for one model informs the definition of the next model. For example, suppose that we generated paths for the medic with the CVRP model under the assumption that there is no rubble. An engineer must clear the rubble blocking a victim before the medic arrives at this location for the paths to be valid. This forms the basis for the engineer routing model definition, which is akin to the CVRPP. In this definition, we set up rewards if the engineer can clear path rubble before it is needed or if the engineer can be at a critical victim location near the same time as the medic so that triage can start.

Each team composition defines a different sequential decision-making problem. When we have completed the model definitions for all team compositions, we can compare the results across teams. Here, we only define a (medic, engineer, engineer) team for demonstration purposes. One of the main assumptions made with this team composition is that the second engineer switches to the medic role if time permits and starts triaging victims in reverse order from the medic routing solution. Figure 1 depicts the routing problem definition process for this team composition in detail.

Our sequential decision-making framework can start at any arbitrary point during an experiment. Thus, it is possible to run the framework and generate solutions for any combination of victims, rubble, and freezing plates. Additionally, the framework can also be initialized with the information that participants have (or should have), yielding solutions

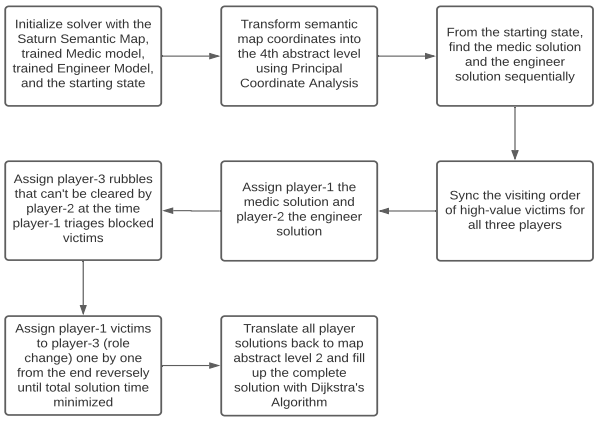

Figure 1: The sample process to generate a complete routing solution for a (medic, engineer, engineer) team

representative of participant knowledge states. Such a capability also allows the ASI agent to compose what-if type of questions to develop a better intuition on the routing options available to the team under different assumptions.

Solution Pipeline

Our framework adapts the ALSR codebase. We trained separate models for the medic and engineer roles using various network sizes so that the solution generation process can utilize the appropriate model to produce routes. Save a few modifications to set up the routes correctly, we directly utilized a CVRP model native to the ALSR codebase for training medic routes. We had to update the ALSR codebase for training CVRPP models for the engineer role. For instance, we added timing-based penalties to the objective function to make the engineer route better aligned with the medic route. We defined two types of penalties: (1) Being late, e.g., there is a penalty if the engineer clears rubble later than the medic needs it; and (2) Being off-sync, e.g., there is a penalty if the engineer visits critical victims or freezing plates before or after the medic's visits. Algorithm 1 presents the CVRPP model generation in detail. In this model, parameters control the weights for each type of penalty and the length of the proposed route in the objective function.

The ALSR codebase strictly requires using 2D coordinates (between 0 and 1), not distances between nodes, as input. We addressed this requirement by setting up four abstraction levels of the Minecraft SAR task maps. The lowest abstraction is the original task map (Saturn) (Figure 2a). Our second abstraction level is a semantic graph that captures all the main map features and structures (rooms, victims, rubble, and the connection of them) needed for navigation decisions (Figure 2b). Each role has objective nodes, e.g., the medic role needs only the victim nodes, whereas the engineer role requires the rubble nodes directly blocking victims, the critical victims, and the freeze plates. The third abstraction level is a distance matrix capturing pairwise distances for all objective nodes (Figure 2c), where Dijkstra's algorithm (Dijkstra 1959) is utilized to calculate the distances between objective nodes directly based on the semantic graph representation (Figure 2b). The fourth abstraction level is a set of 2D coordinates between 0 and 1 dictated by the ALSR codebase. Because the distance matrix generated in the third abstraction level is non-euclidean, we performed Metric Multidimensional Scaling (mMDS) (Kruskal 1978; Cox and Cox 2008), also known as Principal Coordinate Analysis (PCoA), to transform the distance matrix into coordinates in high-dimensional space ($\sim$36 dim). We then used the John-Lindenstrauss Transform (JLT) (Johnson and Lindenstrauss 1984) to generate 2D coordinates from those in the high-dimensional space while preserving distance information from the initial matrix (Indyk, Matoušek, and Sidiropoulos 2017). As the last step, the output coordinates from JLT are scaled to $[0, 1]$ to finalize the fourth abstraction level of the map (Figure 2d). We perform route planning via neural heuristics on the fourth abstraction level and convert the generated routes back to the semantic map representation for visualization, analysis, and interfacing purposes.

Algorithm 1: Training Pipeline for the Engineer's Routing Model

Input: trained medic model $model_m$, normal victims V_n, high-value victims V_h, rubbles R, freezing plates F, medic tool life T_m, engineer tool life T_e, mismatch penalty coefficient α, rubble penalty coefficient β
Output: Trained Engineer model $model_e$
Initialize Engineer model $model_e$
Initialize Medic Graph G_m and Engineer Graph G_e
repeat
  $G_m \leftarrow$ `RandUnif2D(`$|V_n| + |V_h| + 1$`)`
  Medic Route $R_m \leftarrow model_m(G_m, T_m)$
  foreach *node* $u \in G_m$ **do**
    if $u \in V_h \cup R \cup F$ **then**
      $G_e \leftarrow G_e \cup \{u\}$
    end
  end
  Engineer Route $R_e \leftarrow model_e(G_e, T_e)$
  Mismatch Penalty $P_m \leftarrow 0$
  Rubble Penalty $P_r \leftarrow 0$
  foreach *node* $u \in G_e$ **do**
    $t_m^u \leftarrow$ `time(`R_m, u`)`, $t_e^u \leftarrow$ `time(`R_e, u`)`
    if $u \in V_h \cup F$ **then**
      $P_m \leftarrow P_m + l_2(t_m^u, t_e^u)$
    end
    if $u \in R$ **and** $t_e^u > t_m^u$ **then**
      $P_r \leftarrow P_r + 1$
    end
  end
  $cost \leftarrow |R_e| + \alpha \cdot P_m + \beta \cdot P_r$
  Update $model_e$ parameters based on $cost$
until *convergence*

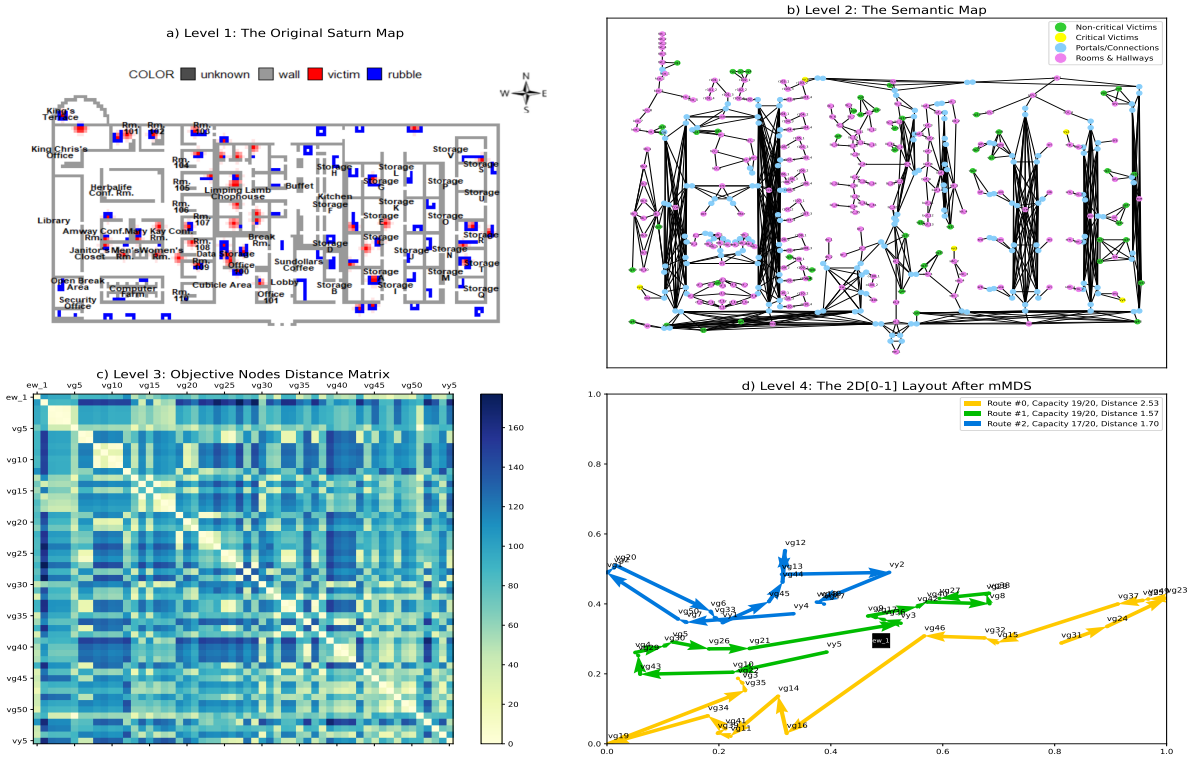

Figure 2: Four Abstract Levels of the Saturn Map. a) Plan view of a representative Saturn map in its original form. b) Semantic map with victims/rooms/connections extracted. Note: the map structures are consistent with that in (a), but victim distribution is different. c) Distance matrix (in meters) of objective nodes extracted. d) An euclidean 2-dimensional layout transformed from the distance matrix using mMDS and JLT. Coordinates are scaled to [0-1]. A sample routing solution with three routes is shown.

Exploratory Results

We ran two experiments with our task on the Saturn_B map (see Huang et al. 2021). Both assumed the team composition described in the Solution Approach section. The first experiment focused on generating a complete solution for the task. Our sequential decision-making framework generated a solution with a perfect score of 750, an average distance traveled per participant of 1646 blocks, and completed the mission in 802 seconds. In comparison, the four teams from the Human Subjects Research (HSR) trials which started with the same team composition—trials 416, 450, 508, and 523—averaged 392.5 points and a distance of 2974 blocks per participant during their 900 second missions. The second experiment focused on selecting an arbitrary point in a trial and running our pipeline from that point onward. For this experiment, we selected trial 416, and we ran our pipeline given the scenario state at time 10:44 (when a role change from an engineer to medic happens). Our pipeline suggests a solution for the remaining 4:16 of the mission that improves the total score from 480 to 620 points.

Discussions and Future Work

Our pipeline generated superior routes to those followed by HSR teams. This is expected since the framework had access to the maps and victim locations, meaning it had the required information to generate near-optimal routes. Nevertheless, these results showed that our framework has the potential to be a *trusted* tool to service a SAR ASI agent. The ability to quickly generate routing solutions under different conditions would allow the ASI agent to better reason about what routes are available to the SAR team while intervening.

We plan to complete our framework for all possible team compositions and run experiments to compare them in the short term while setting up an interface for interaction with the real-time ASI agent prototype. We will also exploit the decoding stage of our framework by augmenting the masking component in the ALSR codebase to better handle CO model constraints and resolve potential deadlocks, e.g., a medic waits for the engineer to clear rubble, where the engineer waits for the medic at a critical victim. In the medium term, we will explore avenues to generate solutions concurrently for all team members directly on the third abstraction level—the distance matrices based on the semantic map.

Conclusions

We depicted a sequential decision-making framework leveraging neural heuristics to quickly inform a real-time ASI agent on routing options available to a SAR team under different conditions given the state of the SAR mission. Such an agent could leverage this framework as a trusted tool for routing when reasoning about how to assist the SAR teams better in Minecraft SAR experiments.

Acknowledgements

Part of the effort depicted is sponsored by the U.S. Army Research Laboratory (ARL) under contract number W911NF-14-D-0005 and by the Defense Advanced Research Projects Agency (DARPA) under contract number W911NF2010011, and that the content of the information does not necessarily reflect the position or the policy of the Government or the Defense Advanced Research Projects Agency, and no official endorsements should be inferred.

References

Bengio, Y.; Lodi, A.; and Prouvost, A. 2021. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research* 290(2): 405–421.

Bonaccio, S.; and Dalal, R. S. 2006. Advice taking and decision-making: An integrative literature review, and implications for the organizational sciences. *Organizational behavior and human decision processes* 101(2): 127–151.

Boussaïd, I.; Lepagnot, J.; and Siarry, P. 2013. A survey on optimization metaheuristics. *Information sciences* 237: 82–117.

Cappart, Q.; Chételat, D.; Khalil, E.; Lodi, A.; Morris, C.; and Veličković, P. 2021. Combinatorial optimization and reasoning with graph neural networks. *arXiv preprint arXiv:2102.09544* .

Cox, M. A.; and Cox, T. F. 2008. Multidimensional scaling. In *Handbook of data visualization*, 315–347. Springer.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1(1): 269–271.

French, J. R.; Raven, B.; and Cartwright, D. 1959. The bases of social power. *Classics of organization theory* 7: 311–320.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, 1263–1272. PMLR.

Huang, L.; Freeman, J.; Cooke, N.; Dubrow, S.; Colonna-Romano, J.; Wood, M.; Buchanan, V.; and Caufman, S. 2021. ASIST Study 2 June 2021 Exercises for Artificial Social Intelligence in Minecraft Search and Rescue for Teams. URL https://doi.org/10.17605/OSF.IO/GXPQ5.

Indyk, P.; Matoušek, J.; and Sidiropoulos, A. 2017. 8: low-distortion embeddings of finite metric spaces. In *Handbook of discrete and computational geometry*, 211–231. Chapman and Hall/CRC.

Johnson, W. B.; and Lindenstrauss, J. 1984. Extensions of Lipschitz mappings into a Hilbert space 26. *Contemporary mathematics* 26.

Kool, W.; Van Hoof, H.; and Welling, M. 2018. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475* .

Korte, B.; and Vygen, J. 2012. The traveling salesman problem. In *Combinatorial Optimization*, 557–592. Springer.

Kruskal, J. B. 1978. *Multidimensional scaling*. 11. Sage.

Starcke, K.; and Brand, M. 2016. Effects of stress on decisions under uncertainty: A meta-analysis. *Psychological bulletin* 142(9): 909.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* .

Vesselinova, N.; Steinert, R.; Perez-Ramirez, D. F.; and Boman, M. 2020. Learning combinatorial optimization on graphs: A survey with applications to networking. *IEEE Access* 8: 120388–120416.

Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1: 57–81.