Interplay of Human and AI Solvers on a Planning Problem

Afshin Ameri E.*, Branko Miloradović*, Baran Çürüklü*, Alessandro V. Papadopoulos*,
Mikael Ekström*, Johann Dreo†
*Mälardalen University, Västerås, Sweden. Email: {name.surname}@mdu.se
† Computational Biology dept., Université Paris Cité, Institut Pasteur, Paris, France. Email: johann.dreo@pasteur.fr

***Abstract*—With the rapidly growing use of Multi-Agent Systems (MASs), which can exponentially increase the system complexity, the problem of planning a mission for MASs became more intricate. In some MASs, human operators are still involved in various decision-making processes, including manual mission planning, which can be an ineffective approach for any non-trivial problem. Mission planning and re-planning can be represented as a combinatorial optimization problem. Computing a solution to these types of problems is notoriously difficult and not scalable, posing a challenge even to cutting-edge solvers. As time is usually considered an essential resource in MASs, automated solvers have a limited time to provide a solution. The downside of this approach is that it can take a substantial amount of time for the automated solver to provide a sub-optimal solution.**

In this work, we are interested in the interplay between a human operator and an automated solver and whether it is more efficient to let a human or an automated solver handle the planning and re-planning problems, or if the combination of the two is a better approach. We thus propose an experimental setup to evaluate the effect of having a human operator included in the mission planning and re-planning process. Our tests are performed on a series of instances with gradually increasing complexity and involve a group of human operators and a metaheuristic solver based on a genetic algorithm. We measure the effect of the interplay on both the quality and structure of the output solutions. Our results show that the best setup is to let the operator come up with a few solutions, before letting the solver improve them.

***Index Terms*—Mixed Human-AI Planning, Human-AI Collaboration, Multi-Agent Mission Planning.**

I. Introduction

In recent years, Multi-Agent Systems (MASs) have gained more popularity due to their ability to solve problems that are hard to solve by a monolithic system. The technological advancements, regarding both software and hardware aspects of the system, have sped up the adoption process of MAS. Consequently, systems that include more agents tend to be more complex to control and use. The aspect of MAS, that we are interested in, is mission planning for multi-robot systems.

The goal of mission planning is to produce a plan, i.e., a course of action for every agent, in order to achieve some given goal. What makes this part challenging is the potential complexity of the mission, which is usually correlated to the number of agents, the number of tasks, and the constraints included in the mission. In the general sense, the aim is to allocate tasks to agents with respect to given constraints, so that all the tasks are successfully completed. Finding a feasible plan may be sufficient in some cases, but in many other cases, one is interested in optimizing the plan with respect to some relevant metrics. For this purpose, the mission planning problem is usually formulated as an optimization problem, where the optimization function targets the minimization of a cost, like for example the mission's makespan. Depending on the mission requirements, different formal models exist that can be used in order to mathematically define such missions.

In practice, MASs usually come with Command & Control (C2) systems that require a human to operate them. The operator has an overview and control —through the C2— of information gathering, data processing, mission definition, planning, and plan execution. It is not uncommon for the operator to be directly in charge of the mission planning [1]. This option gives more flexibility to mission planning, as humans have the ability to adjust to unforeseen changes in the mission. However, while creating a plan, a human operator has no guarantee of the optimality of the produced plan. Moreover, humans can hardly deal with large problem instances, with several tasks and agents, and some of the mission constraints might be difficult to take into account.

To circumvent this issue, automated planners have been developed, that rely on Artificial Intelligence (AI) for optimization of the planning problem. AI solvers have the ability to handle complex missions, with a large number of constraints [2]. The downside of this approach is that sometimes it can take a lot of time to produce a solution that is sub-optimal. In order to overcome the aforementioned issues, we try to combine the best of both approaches in order to try and understand what is the interplay between the human operator and the automated planner. We provide a comprehensive analysis of how the two can coexist in the domain of mission planning. More specifically, we investigate how to include a human operator in the loop of the mission planning process. The contribution of this paper is twofold:

1) We conducted a series of experiments to assess the effect of a human using a metaheuristic optimization solver to improve their solution.
2) We measure this effect on both the *quality* and the

This work was supported by the Swedish Research Council (VR) with the PSI project (No. #2020-05094), by the Knowledge Foundation (KKS) with the FIESTA project (No. #20190034).

structure of the output solutions, along with how solvers transition from initial to final solutions.

II. Background and Motivation

The computation of the optimal solution of combinatorial optimization problems does not scale with the size of the problem. Even for small instances, the computation of the solution may be intractable. A typical example is the well-known Traveling Salesman Problem (TSP) [3]. TSP is an NP-Hard problem, meaning that no algorithm currently exists that can solve these kinds of problems in polynomial time. However, TSP has several very important applications in different domains, including manufacturing, transportation systems, and mission planning. The latter is the focus of our work.

Current-day mission command centers are operated by human operators. In some cases, a human operator is in charge of the mission planning. The alternative is to use automated planners that, given the mission inputs, can provide a solution in the form of allocation of tasks to agents, and ordering the execution of those tasks in such a way as to minimize the overall execution time.

On the other hand, automated planners may take a long time to produce a good solution. This has led some researchers to investigate if humans may be able to produce a good solution in a reasonable time. Dry *et al.* [4] showed that humans are capable of solving TSP problems up to the size of 120 nodes. The provided solutions are within 11% of the optimal solution. Moreover, they showed that the relationship between the number of cities and the solution quality is linear, meaning that the problems with the size of 10-20 cities were solved within 1% of the optimal solution. These findings are similar to the ones in the work of Vickets *et al.* [5], where even larger instances with about 40 cities were solved within 2.5% of the optimal solution. There are more studies conducted on the human ability to solve TSP problems. An overview of the literature on this subject is given by MacGregor and Chu [6]. It has been shown that spatial heuristics help humans to reduce the search space to a smaller size for similar planning problems [7], [8]. This human insight has been shown to be helpful in reducing planning time and plan quality in a collaborative setting by Kim. *et al.* [9]. However, the solution presented by Kim. *et al.* requires a domain expert to translate human planning strategies into computer code. Although TSP is a very hard problem to solve, it can be further generalized with additional constraints. These constraints are necessary to describe more complex missions. All of this led us to investigate how those performances translate to more complex problems, specifically a generalization of the TSP involving additional constraints like multiple salesmen with limited access to the cities, i.e., not every salesman can visit every city. This problem is known in the literature as Colored TSP (CTSP) described by Li. *et al.* [10].

In this study, we focus on the efficiency of a user *using* an AI as a support tool on the multi-agent mission planning problems set in the domain of underwater robotic mission planning without the need of employing domain experts to translate the human contribution.

III. Experiments Setup

The overall goal of the conducted experiments is to understand if including the human operator, at any stage of the planning process, can help improve the overall mission plan, especially considering that humans can use spatial cues in solving such planning problems [8]. In order to evaluate this, we set up a number of experiments. Firstly, the mission problem is solved with an automated planner. The planner is based on a meta-heuristic approach, specifically, the Genetic Algorithm (GA), and as such does not guarantee that the provided solution is optimal. More information on the planner implementation can be found in Sect. III-C1. Secondly, the mission problem is presented to the human subjects and they have a limited time to provide a solution, like in an operational setting. They are allowed to evaluate their plan as many times as they want within the given time span. Evaluation of the plan gives the human subject the idea of the quality of their solution compared to the best solution found by the automated planner. The quality difference is expressed in percentages after each plan submission. Thirdly, the tests are performed where the solutions provided by the human subjects are submitted to the automated planner as the initial solutions. The reasoning behind this is that those solutions might help the automated planner to avoid local minima pitfalls and produce overall better solutions.

A. Scenarios

The experiments are based on an underwater environment scenario using Autonomous Underwater Vehicles (AUVs). The aim is to plan missions involving a group of heterogeneous AUVs to perform a set of tasks. Each AUV might be equipped with one or more of the following sensors (i) a Camera capable of taking photos; (ii) a Sonar used for measuring acoustic characteristics of the target location; and (iii) a H_2S chemical sensor to measure the level of Hydrogen Sulfide in the water. In such scenarios, an agent[1] (in this case, an AUV) is expected to start its mission from a source depot, visit all necessary locations in order to complete allocated tasks and go to the destination depot. It is the planner's job to make sure the agent to which the tasks are allocated to has the necessary equipment to perform those tasks. The destination depot is represented as a surface point where AUV ends its mission, resurfaces, and awaits to be picked up by a surface vehicle.

B. Human Setup

A total of 16 participants were involved in the tests. The participants were in the age range of 24-41 (11 males, 5 females), and they volunteered to take part in tests. Throughout the tests, the only information gathered from the participants is the solutions that they provide to the planning problems. No personal data is stored in any form during the experiments, as no stratification is performed in the analysis.

[1]In the context of this paper, we use the terms robot, agent, vehicle, vessel and AUV interchangeably.

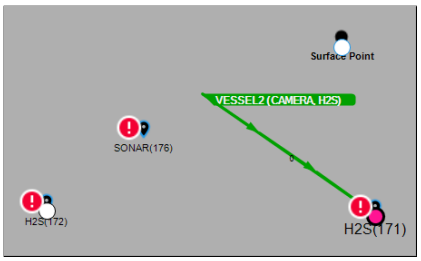


Fig. 1: MMT's User Interface while planning a mission. Each vehicle is color-coded and represented by a name and list of its available equipment. The tasks are also marked with the sensor required to perform the task. The green line represents the path that the vehicle takes to the next task (H2S(171)).

1) User Interface: The experiments are performed through the Mission Management Tool [11] (MMT), which provides a Graphical User Interface (GUI) for planning and supervising multi-agent missions (Fig. 2). The MMT's GUI represents mission tasks and agents as markers on a map. During mission planning, the operator's goal is to define a path that connects an agent with a set of tasks and ends in a *Surface Point* (Fig. 1). Each section of the path connecting two tasks, i.e., their locations, is called a transit. The vehicles and their paths are color-coded. When a vehicle visits a task, that task assumes the vehicle's color. For defining a new transit, the operator can drag the end point of a path, to the location of a new task. During this operation, the UI highlights the tasks that the selected vehicle can perform with white circles. These circles are shown on the map near the task marker location. This means that tasks which the vehicle cannot perform will not be highlighted as drop targets, hence preventing the user from erroneous task-vehicle assignments. For example, in Fig. 1, VESSEL2 (the green vehicle) does not have a sonar, therefore while creating a path for this vehicle, all sonar tasks are not recognized as drop points for it and do not show the white circular drop point. The exclamation marks on tasks, inform the user that the task is not included in the plan or one of the following conditions apply (i) The task is not visited by any vehicle; (ii) The task is visited by a vehicle, but the vehicle has not moved to another task or a surface point; and (iii) The task is visited by more than one vehicle. During mission (re)planning, the operators are allowed to edit their plans. The editing assumes two types of operations. The first one is the deletion of the path between two task locations by selecting the path (by clicking on one of the endpoints) and pressing the delete key. The second edit operation assumes the selection of one of the two connected task locations, disconnection of the path link from that task location, and the execution of a drag-and-drop operation to make a connection with a new task location. When all vehicles have a path that ends at a surface point and there are no tasks marked with an error icon, the planning process is finished.

2) User Tests: Each user is presented with 6 different scenarios. Each scenario involves a different number of tasks and vehicles. More information about the mission settings can be found in (Table I). The users were given the set of problems as a whole, and they had the liberty to choose their preferred order of solving them. For each test, the user has to plan or re-plan a mission within a given amount of time. The re-planning scenarios are different from the planning scenarios of the same size (i.e., P-S and RP-S involve two different missions). This choice has been made to prevent the experience gained by the human, while solving a planning problem, to be used in solving the re-planning problem of the same size.

TABLE I: Test missions provided to the users.

Test Name	No. Tasks	No. Vehicles	Time	Type
P-S	10	2	4 min.	Planning
P-M	23	3	7 min.	Planning
P-L	32	4	10 min.	Planning
RP-S	10	2	4 min.	Re-planning
RP-M	21	3	7 min.	Re-planning
RP-L	32	4	10 min.	Re-planning

Mission planning problems are presented to the user in the form of a map with markers denoting the location of tasks and vehicles, as well as, equipment availability. The user is then required to provide a valid plan by creating a path from the starting location of a vehicle through the locations of allocated tasks and ending the path at a surface point. The set of tasks to allocate to each vehicle is determined by the user. This is not the only step in the optimization process. The users also have to come up with the ordering of the tasks they allocate. Hence, the mission planning problem consists of two optimization sub-problems, task allocation, and task order. The same applies to re-planning problems.

In a re-planning scenario, the user is presented with an already generated plan, and they are asked to improve upon that plan by reducing the mission's overall execution time. Although the six missions are the same for all the users, the map is randomly rotated at the start of each test, to avoid any bias from the perception of the geometry of the scenario. This is inline with observations that show humans use spatial cues in planning scenarios [8]. Since the AI solver does not use spatial cues, rotating the map reduces the human bias. A plan is considered incomplete/infeasible if (i) not all tasks are included in the plan; (ii) a task is visited more than once; (iii) one or more tasks are allocated to vehicles that do not have appropriate equipment; and (iv) the end location of a vehicle is not a surface point. The User Interface has checks in place to avert these types of errors by either providing visual feedback to the user of potential issues (cases 1 and 2) or completely preventing the user to make such allocations (cases 3 and 4). Therefore user generated plans will be infeasible if the user ignores the warnings from the UI regarding task allocations. Similar checks are also implemented in the AI Solver, thus AI Solver results will not include infeasible solutions.

Before performing the experiments with users, we ran the AI solver on all test problems used in this paper. The AI solver was given 2 hours per problem to try and find the best

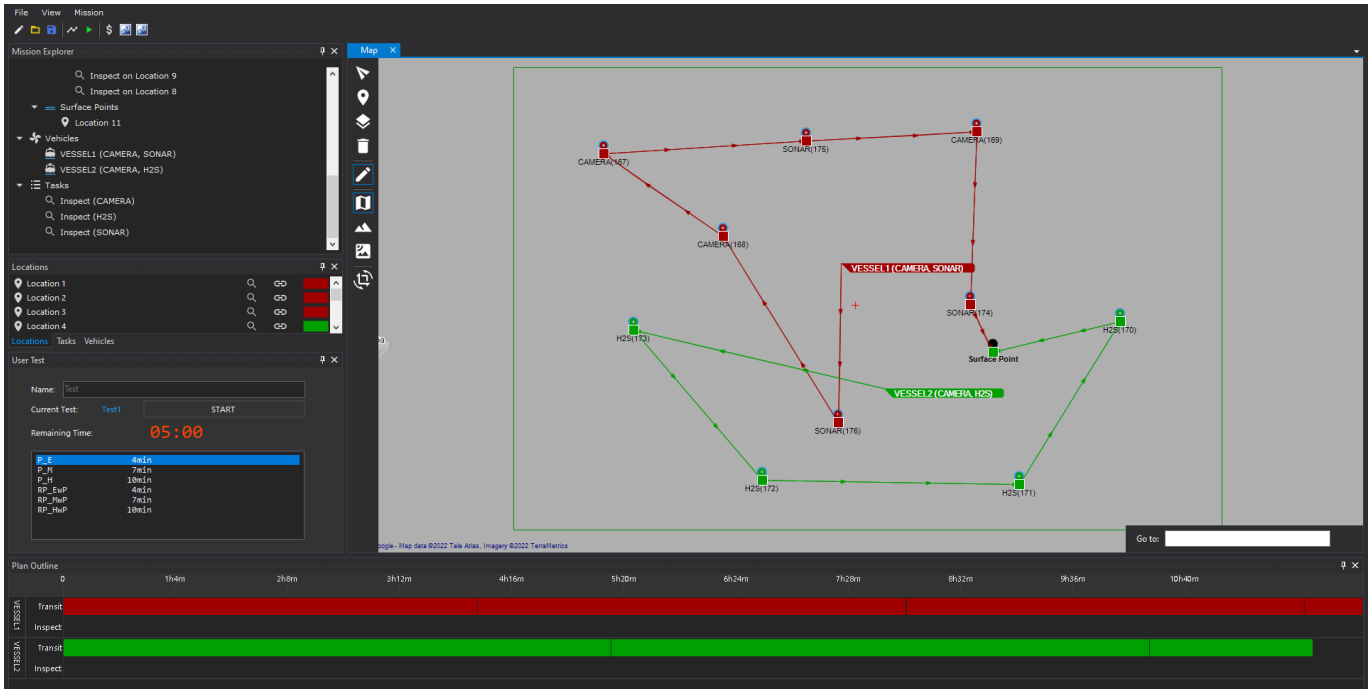


Fig. 2: A full mission planned in MMT with the color-coded Gantt chart showing the makespan of each agent's plan. The lines represent the vehicles' paths between the tasks.

solution possible for each of the instances used in the tests. The best-found solutions are then used to provide feedback to the user during the experiments. More specifically, a user is given the option to evaluate their current plan by pressing the evaluate button in the UI. This starts the process of evaluation by comparing the makespan of the submitted plan with the makespan of the best-found solution by the AI solver. The user is then informed of his performance, with a number showing the gap between the submitted solution and the best-found solver's solution. This gap is expressed in percentages, e.g. if the gap is 0% it means that the user's solution matches the best-found solver's solution. This "score" solely serves the purpose of motivating the test subjects to continue and improve the plan if they see there is room for improvement. In the realistic scenario the system will have no way of knowing how good the users solution is, since there would not exist an AI solution to compare with.

For each scenario, the user is allowed to edit their plan as many times as they wish during the session, as long as the deadline for that test has not reached. If the deadline for a test has reached, no more plan evaluations are allowed and the user has to move to the next test, or if all tests are finished, the user is done with the experiment. The test procedure for each participant is as follows: (i) A video tutorial, of approximately 10 minutes, is presented to the user on how to use the MMT's UI and plan a mission; (ii) The user is then given 15 minutes to practice on a sample mission using the UI. During this period they can ask the test supervisor to clarify any misunderstanding they might have about the UI or mission planning; (iii) The user starts the tests. Each test has a limited time during which the user can create their solutions to the given (re)planning problem (Table I) and evaluate them. During this period, the test supervisor is not present in the room; and (iv) Every user plan and its evaluation result are automatically saved for further analysis.

C. Solver Setup

The mission planning problem in this paper has a theoretical background in the CTSP. It was introduced by Li *et al.* [10]. In our scenario, a city corresponds to a task that must be completed, a salesman to a robot, and a color to an equipment type (e.g., Camera, Gripper, etc.). The colors associated with a robot represent the robot's equipment, while the colors associated with a task indicate the equipment required for its successful completion.

1) AI solver: The solver used in this setup is based on the Genetic Algorithm (GA) adapted to this specific planning problem. The GA solver is based on the solver presented by Miloradović *et al.* [12], with the difference being that the problems used in this paper do not have precedence constraints. However, for completeness, a short description of the solver will be presented here.

Chromosomes (candidate solutions) are encoded as two arrays of integers, where the first array consists of integers representing tasks and agents, and the second array represents task parameters (equipment requirements, task duration, and location). Each array begins with an agent gene, followed by the array of task genes that are allocated to that agent. Chromosome length varies from $n + 1$ to a maximum of $n + m$ genes, depending on the number of agents used. The

initial population is randomly created with respect to the given constraints, hence, initial solutions are all feasible. The crossover operator has not been used since it did not have positive effects on the convergence process.

The mutation is the only source of variability, as it allows genetic diversity in the population. Every individual has a low probability (10%) to be selected for mutation. Increasing the mutation probability beyond this value does not benefit the overall results. In this paper, two types of mutation schemes are used. One operates on the task genes by swapping tasks and inserting new genes, whereas the other mutates agent genes through growing (adding agents) and shrinking (removing agents) from the chromosome. A task swap mutation swaps two task genes in a chromosome, meaning that it can swap tasks within a single agent or between two agents. An insert mutation chooses a task and inserts it in a new location, similarly to the previously explained mutation, the insertion can be within the same agent or a different one. An agent shrink mutation removes one agent from a chromosome, reallocating its tasks to other agents. Growth agent mutation adds a new agent to the plan. The new agent gene is randomly inserted in the chromosome, acquiring tasks from that location in the chromosome up to the next agent gene or end of the chromosome. If there are conflicting tasks (a task not supported by an assigned agent), they are randomly reallocated to other agents that can successfully complete them. Both algorithms take into account given constraints, ensuring that the mutation process does not produce infeasible solutions.

The rest of the solver settings include the stopping criteria of 500 generations. One solver run consists of 10 restarts, meaning that after the stopping criterion is reached, the solver performs a restart and the population is initialized again from scratch. The best solution found is then sent back to the MMT. To keep the comparisons fair, for each test, AI gets the same amount of runs as the number of user solutions. Note that for each run, multiple independent restarts are necessary to assess the performance of our randomized solver. So having multiple independent restarts does not bias the performance, as the solver does not get any information from one restart to the other. These restarts are a feature of the solver, as reseeding (restarting) the population can lead to different overall performance, meaning they do not bring any advantage to the AI-only approach.

2) Test scenarios: The solver is given the same scenarios as humans and is limited by the number of generations, which always results in less run-time than the human's time budget (see Sect. III-B2). The solver either starts from random feasible initial solutions (AI-only), or from the solutions produced by the user (User+AI). In all cases, multiple runs are performed to correctly assess the solver's performance. One run of the solver is performed for each solution provided by the human, for each problem instance. In the AI planning setup, 41 independent runs are performed for the small-sized (P-S) problem instance, 52 independent runs for the medium-sized problem instance (P-M), and 83 independent runs for the large-sized problem instance (P-L). Similarly, in the User+AI setup, for each human plan, the AI runs once, hence the total number of runs equals the total number of human plans.

For assessing the performance, we consider a confirmatory analysis of the distributions of the costs of the best solutions produced by all runs. This corresponds to the fixed-budget benchmarking recommended by [13], considering central tendency and dispersion for confirmatory analysis, with a simple design of experiments, without tuning. In our case, the solving time is a more important constraint for the C2 system, than the quality of the plan or the ratio of fixed-target success. Additionally, users alone very often don't reach good solutions, and don't necessarily behave by incrementally improving the solutions, so that a fixed-target setting would recover less information. Finally, the users do not evaluate many solutions sufficiently to consider a cumulative approach, simulated restarts or feasibility rates [13].

IV. Results

In the result section, we analyze the performance of the three approaches based on solution quality expressed through cost and distance in the decision space. For simplicity, and without loss of generality, we chose the duration of every task to be 0. Hence, the overall cost is a reflection of travel times between tasks. We also made each vehicle's velocity the same. These choices simplify the solution of the problem on the human side, as it reduces the variables to consider while identifying a solution, but they can be easily handled by the automated planner. The distance between solutions is calculated using Hamming distance algorithm [14]. The Hamming distance between two strings of symbols (e.g., letters, numbers, etc.) is the measure of the number of places at which the corresponding symbols are different. This metric is useful as it shows us how many perturbations are needed to move from one solution to the other in the decision space.

A. End cost distributions

This first set of results considers the distribution of the best solutions found at the end of the three different approaches (Fig. 3). Results show that in all planning tests, the AI solver has produced plans with better costs than the human alone. For the P-S problem, all instances of the AI solver were able to find a solution with the same cost as the best human plan. In the P-M problem, the solver can either match or outperform the best human solution, while in the P-L problem, the solver has outperformed the humans (all solver solutions have lower costs). In re-planning tests, apart from the RP-S problem where the results are similar to the above, for both RP-M and RP-L problems, there have been instances where the human plans had better cost than the re-plans found by the AI solver.

The User+AI results do not show any improvement in the P-S and RP-S cases over the AI-only results, and it might be that the problem is solved to optimality by AI-only in both cases. For P-M and P-L problems, it can be observed that the User+AI results improve on the AI-only solutions as now there are more plans with better cost compared to AI-only solutions, although the value of the best cost remains the same in the

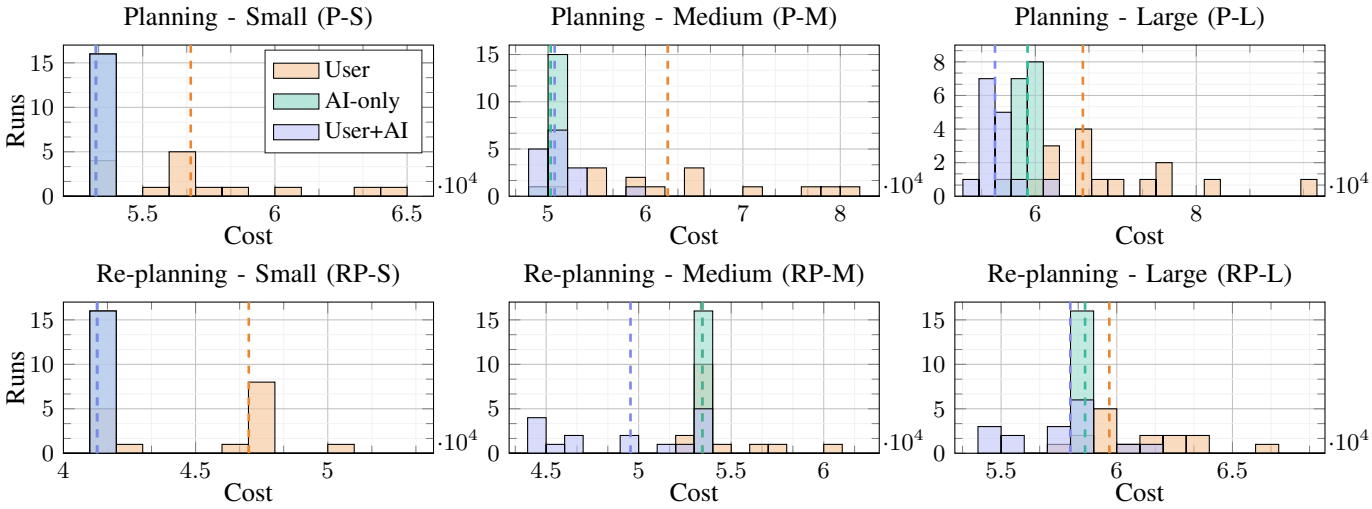
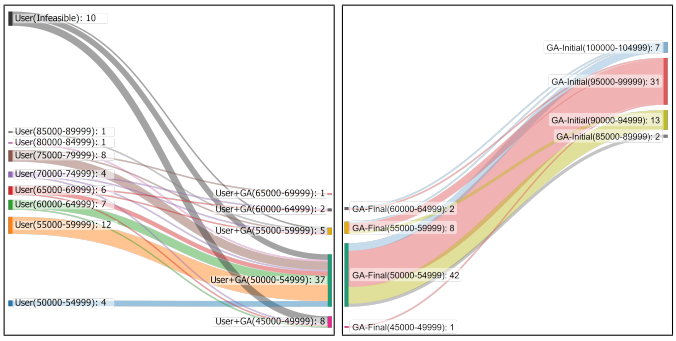


Fig. 3: The best solution distribution from the AI solver, User alone, and User+AI. The X-axis shows the cost and Y-axis is the number of runs ending with a solution of such cost. Vertical dashed lines in the figures represent the median of each of the 3 distributions, for each of the six different instances. Please note that in some cases bins overlap, e.g., in P-S case bins for AI-only and User+AI are on top of each other. There are other partial overlapping, e.g., in RP-M case, at the value of $5.3 \cdot 10^4$ User+AI has 5 results, User only 10 results, and AI-only 16 results.



(a) P-M User+AI (b) P-M AI-only

Fig. 4: Sankey diagrams comparing the medium-sized planning problem. (a) Cost results made by AI starting from user plans. (b) AI-only results starting from random solutions. The vertical axis shows the *ranked* histogram of costs. The left (and right) side displays the distribution of the initial costs, and the center part of the diagram shows the end costs. Lines linking initial and end buckets demonstrate the number of runs resulting in such transitions. Numbers in brackets are the buckets' ranges, the last number is the number of runs.

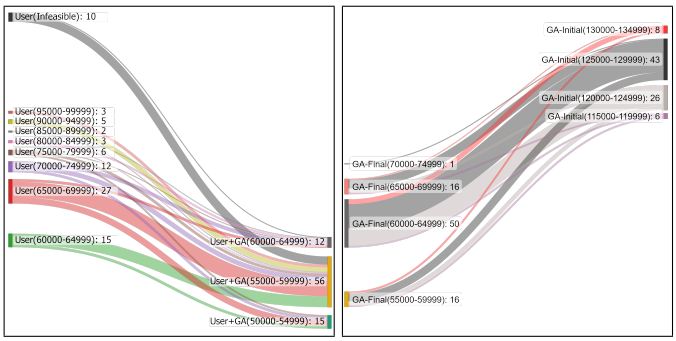
P-M case. In the P-L case, the User+AI approach is actually able to find new, better solutions at a lower cost. This is also the case for RP-M and RP-L problems, as in these cases, the User+AI approach has clearly produced plans that have much better cost than the solutions produced by the AI or humans alone. This is especially visible in the RP-M problem, where the median has reduced from 54 000 to 49 000. In addition, the User+AI approach managed to find new, better solutions, for both RP-M and RP-L cases, as can be seen in Fig. 3. This

TABLE II: p-values of the corrected statistical test between the three pairs of experiments, for all the problem instances.

Instance	P-S	P-M	P-L	RP-S	RP-M	RP-L
User / AI	$5.3 \cdot 10^{-5}$	$3.0 \cdot 10^{-6}$	$2.3 \cdot 10^{-6}$	$1.1 \cdot 10^{-5}$	0.81	$2.8 \cdot 10^{-5}$
User / UAI	$5.3 \cdot 10^{-5}$	$1.6 \cdot 10^{-5}$	$4.1 \cdot 10^{-6}$	$1.1 \cdot 10^{-5}$	$4.2 \cdot 10^{-4}$	$1.7 \cdot 10^{-4}$
AI / UAI	/	0.18	$1.3 \cdot 10^{-4}$	/	$7.9 \cdot 10^{-5}$	$3.7 \cdot 10^{-4}$

figure also shows that in all three re-planning scenarios, the results provided by the AI-only approach fall into a single bin, with little to no variance. This supports the assumption that the AI-only approach gravitated toward a local minimum point and exactly in these situations, using the approach of User+AI seems the most beneficial. User-produced plans have higher variance, thus using them as the input to the AI solver increases the chances of escaping the local point of attraction. They also help improve the overall solution quality.

Table II shows p-values computed by a non-parametric Mann-Whitney-Wilcoxon test, where three different cases are compared (i) User vs. AI; (ii) User vs. User+AI; and AI vs. User+AI. This test is used since sample data can be highly skewed and have extended tails, in which case the measure of the average is known to be non-robust. In order to circumvent this potential issue, the median value was used instead of the mean. For user results, the cost of the best plan (lowest cost) is considered. Costs distribution for the small planning problem (Fig. 3, top left) shows that the user alone is not able to out-compete the AI solver ($p < 10^{-4}$). Whenever AI is involved, it leads to the very same cost. For medium and large planning problems (Fig. 3, top right), the distributions of costs produced with the AI solver involved show some variance, but overall significantly outperform the user alone ($p < 10^{-4}$). For all planning problems, the involvement of



(a) P-L User+AI (b) P-L AI-only

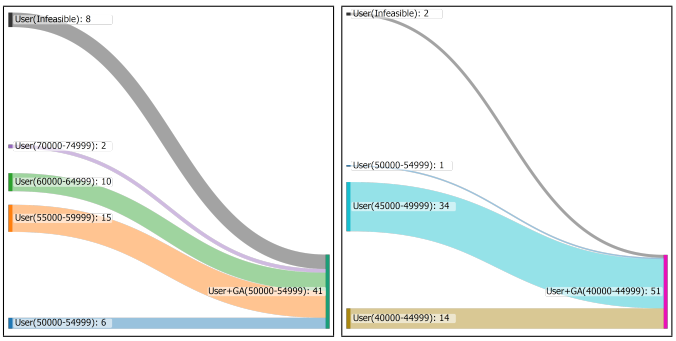
Fig. 5: Sankey diagrams comparing the large-sized planning problem. (a) Cost results made by AI starting from user plans. (b) AI-only results starting from random solutions.

the AI solver thus leads to better performance than those of the user. This observation holds for the small re-planning scenario as well (Fig. 3, bottom left, $p < 10^{-4}$ in Table II). For the medium re-planning scenario (Fig. 3, bottom middle), only the difference between the AI and User has no statistical significance, while the difference between AI and User+AI, and User and User+AI is statistically significant in favor of User+AI in both cases. However, there is no significant difference between the User and AI distributions. For the large re-planning problem (Fig. 3, bottom right), there is a significant difference between each category, i.e., User+AI outperforms both User and AI approaches. The AI approach outperforms the User only approach. Overall, the use of an AI solver allows for better performance in planning and simple re-planning scenarios, while in other problems the results show that the approach of combining User+AI yields the best results.

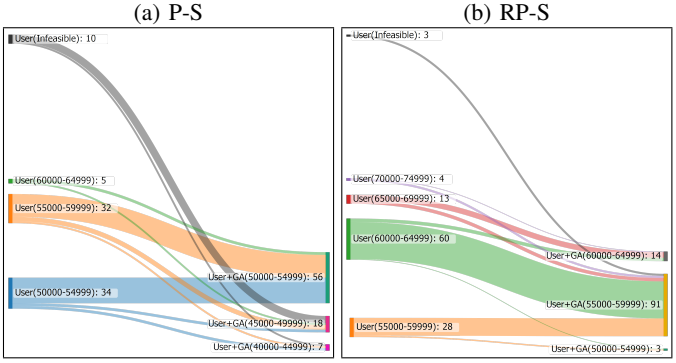
B. Optimization paths

This set of results considers improvements in solution costs while using different categories of solving methods. Figs. 4, 5, and 6 show the results in the form of Sankey diagrams[2], demonstrating the transition from an initial solution cost to a final solution cost. The sides of the diagrams contain histograms presenting plan costs ranked vertically on a relative scale (histograms to the bottom of the diagram have lower costs). The vertical position is not a function of the cost values, but of the rank of the bucket among other cost targets. The left side of the diagrams represents cost histograms for plans provided by the users, while the right side of the diagrams contains cost histograms for plans generated by the AI solver using the user plans as initial solutions. Figs. 4 and 5 present User+AI and AI-only diagrams for medium and large problems side by side. In both figures, the Sankey diagram for the AI-only solution is flipped. Note that the end of those diagrams (in the middle of the figures) shows the same data as Fig. 3. Flipped AI-only diagram for the small problem is not presented here as all the AI-only solutions for the small problem have the same initial and final costs. This is also observed for all re-planning missions, as the AI solver starts from the same initial plan/cost and the low variance in final costs (Fig. 3) means that they result in the same final cost. First, it can be observed that the AI-only initial (feasible random) solutions rank worse than most of the user solutions in Figs. 4, and 5. Only a small part of user solutions rank worse, and they are all infeasible solutions. For the medium-sized problem, a good proportion of those infeasible solutions actually lead to the best solutions found. This is not observed for the problem of large size. However, in all cases, several solutions produced by the user can actually lead the AI to find solutions of the best cost.

[2]Generated by SankeyMatic tool: https://sankeymatic.com/



(a) P-S (b) RP-S



(c) RP-M (d) RP-L

Fig. 6: User+AI Sankey diagrams for (a) small planning, (b) small re-planning, (c) medium re-planning and (d) large re-planning problems.

C. Decision Space

Solutions, as mentioned in Sect. III-C1, are represented as integer arrays. It is thus possible to measure a distance between them *in the decision space*. Fig. 7 shows the distribution of Hamming distances between all possible pairs of starting and final solutions, for the planning problems. Note that the small problem instances have smaller solutions, i.e., the strings being compared are of shorter lengths. Therefore, the maximum value for the distance metric is the length of the strings being compared, and the minimum one is 0 if the two solutions are exactly the same. In this work, we are comparing the distributions within each problem. In all cases, the median distance between pairs of solutions is smaller when the user is involved. Additionally, the variance is greater in those cases.

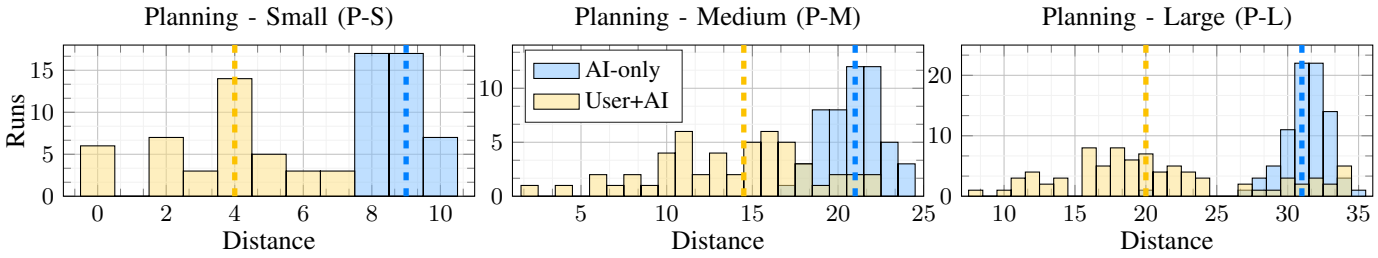


Fig. 7: Distribution of the distances between the starting and final solution. Vertical dashed lines show the median value.

D. Discussion

The general tendency is that the size of the problem is akin to its difficulty, for both the AI solver and the human operator, such a correlation fits the common knowledge of the planning domain. Based on the results, it can be argued that the inputs from the human operator have helped the AI solver to move outside of local minima and explore solutions that lead to lower costs (Fig. 3). Perhaps surprisingly, this ability can come even from infeasible solutions, albeit the random (but feasible) solutions produced by the initialization of the solver do not necessarily allow for the same improvement. This behavior has been observed in the evolutionary computation community [15], where it is sometimes advised to not over-constrain the problem model, and allow the search process to go through infeasible areas of the decision space.

We believe that this means that a human is able to capture some heuristics about what makes a good solution. Given the 2D embedding of the problem, this may be some geometric feature, that the solver, based on a combinatoric model, is not yet equipped to handle. This hypothesis finds some support in the fact that the solver has to search comparatively *less* (in the decision space) to find good solutions, if it starts from the user ones (Fig. 7). In any case, the use of the AI solver to improve user solutions is the best option. It seems still probable that allowing a larger computational budget to the solver would increase its performances to the point that no strong difference would be seen between AI and User+AI results. However, as soon as the complexity of the instance increases, the User+AI approaches may be interesting to reduce the planning time.

V. Conclusion

In our setting, the User+AI approach proved to be useful in planning/re-planning of larger instances. In addition, some of the best solutions produced by this approach came from the (sometimes infeasible) solutions created by the user.

There is always a compromise between the complexity of the problem instance and the time available to solve it. A well-known tendency in C2 systems is to target more complex problems (more agents, more tasks, etc.) and more reactive systems (hence shorter planning time). This paper has shown that the adoption of User+AI approaches would live at the edge of what a purely automated approach can solve, making it a possibly beneficial tool for C2 systems targeting challenging problems. For such challenging problems, there are no expert operators available for experiments, because the problem is new and has generally not been deployed on sufficiently many real use cases. However, whether people having more background knowledge in similar setups would be more efficient in solving the problem remains to be explored.

References

[1] C. Ramírez-Atencia, V. Rodríguez-Fernández, A. González-Pardo, and D. Camacho, "New artificial intelligence approaches for future uav ground control stations," *IEEE CEC*, pp. 2775–2782, 2017.

[2] S. A. Zanlongo, F. Abodo, P. Long, T. Padir, and L. Bobadilla, "Multi-robot scheduling and path-planning for non-overlapping operator attention," in *2018 Second IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2018, pp. 87–94.

[3] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *Journal of the Operations Research Society of America*, vol. 2, no. 4, pp. 393–410, 1954.

[4] M. Dry, M. D. Lee, D. Vickers, and P. Hughes, "Human performance on visually presented traveling salesperson problems with varying numbers of nodes," *The Journal of Problem Solving*, vol. 1, no. 1, p. 4, 2006.

[5] D. Vickers, M. Butavicius, M. Lee, and A. Medvedev, "Human performance on visually presented traveling salesman problems," *Psychological Research*, vol. 65, no. 1, pp. 34–45, 2001.

[6] J. N. MacGregor and Y. Chu, "Human performance on the traveling salesman and related problems: A review," *The Journal of Problem Solving*, vol. 3, no. 2, p. 2, 2011.

[7] T. T. Brunyé, S. B. Martis, and H. A. Taylor, "Cognitive load during route selection increases reliance on spatial heuristics," *Quarterly Journal of Experimental Psychology*, vol. 71, no. 5, pp. 1045–1056, 2018, pMID: 28326966.

[8] S. Rosenthal and L. M. Hiatt, "Human-centered decision support for agenda scheduling," *Proceedings of AAMAS*, vol. 2020-May, pp. 1161–1168, 2020.

[9] J. Kim, C. J. Banks, and J. A. Shah, "Collaborative planning with encoding of users' high-level strategies," in *Proceedings of the 31st AAAI Conference*, ser. AAAI'17. AAAI Press, 2017, p. 955–961.

[10] J. Li, M. Zhou, Q. Sun, X. Dai, and X. Yu, "Colored traveling salesman problem," *IEEE Trans. on Cyb.*, vol. 45, no. 11, pp. 2390–2401, 2015.

[11] E. A. Ameri, B. Cürüklü, B. Miloradovic, and M. Ekström, "Planning and supervising autonomous underwater vehicles through the mission management tool," in *Global Oceans 2020: Singapore*, 2020, pp. 1–7.

[12] B. Miloradović, B. Cürüklü, M. Ekström, and A. V. Papadopoulos, "A genetic algorithm approach to multi-agent mission planning problems," in *Operations Research and Enterprise Systems: ICORES 2019*. Springer, 2019, pp. 109–134.

[13] T. Bartz-Beielstein, C. Doerr, J. Bossek, S. Chandrasekaran, T. Eftimov, A. Fischbach, P. Kerschke, M. López-Ibáñez, K. M. Malan, J. H. Moore, B. Naujoks, P. Orzechowski, V. Volz, M. Wagner, and T. Weise, "Benchmarking in optimization: Best practice and open issues," *CoRR*, vol. abs/2007.03488, 2020.

[14] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[15] H. K. Singh, K. Alam, and T. Ray, "Use of infeasible solutions during constrained evolutionary search: A short survey," in *Artificial Life and Computational Intelligence*, T. Ray, R. Sarker, and X. Li, Eds. Cham: Springer International Publishing, 2016, pp. 193–205.