Nudging Automated Planners with Learned User Preferences

Fusun Yaman[1] [a], Thomas C Eskridge[2] [b] Ron Scott[1], Li Lin[1], Jeff Miller[1], Daniel Carpenter[3]
[1]*BBN Technologies, Cambridge, MA, USA*
[2]*L3Harris Institute for Assured Information, Florida Institute of Technology, Melbourne, FL, USA*
[3]*Air Force Research Laboratory*
fusun.yaman@rtx.com

Keywords: Planning, Scheduling, User Preference

Abstract: Automated planning tools play a large and expanding role in the function of many parts of our lives. The complex nature of the planning problem and the increasing amount of information the human planner must synthesize indicate that assistive automation must soon become the norm. Despite this, many existing automated planners are incapable of producing plans that reflect the desires and expertise of their operators. They do not have the direct ability to consider the operators' priorities, nor can they exploit expert operational knowledge that comes from human experience and not data systems. In this paper we present methods to learn operator planning preferences and then nudge our automated logistic planner to produce plans that are better aligned with operator preferences without changing the code of the planner.

1. INTRODUCTION

AI planner and schedulers are the earliest adopted AI systems (c.f. Sacerdoti, 1977, Zweben and Fox, 1994). While planning and scheduling systems can successfully search through complex and large search spaces to identify valid plans, their outputs are rarely adequate for the human planners. This is because the AI captures hard constraints of the problem, ensuring viability of the solution but oftentimes ignoring the human preferences and perspective into what constitutes a good solution. The complicated nature of the implemented systems and the often life-critical applications they serve makes it impractical or cost prohibitive to modify the code to include the ever-changing human perspective and situationally specific mission priorities.

The objective of Learning Expertise from Air-Mission Planners (LEAP) is to improve the usability and performance of legacy mission planning systems by identifying operator preferences with a minimum number of queries and then externally affecting the output of the mission planning system to produce plans that are aligned with learned preferences without changing the original planning algorithm's code base.

LEAP augments the automated planner using expert, contextualized criteria gathered from interactive learning sessions with subject matter experts (SME)s, with a minimal number of questions. In this paper we demonstrate that LEAP could improve plan quality produced by a black box mission planner through a small number of interactions with human SMEs. Our contributions are following:

1. Identified classes of preferences that could be learned with minimal querying.
2. Developed a general Question & Answer (Q&A) framework with a question generation strategy that was based on reducing the uncertainty while allowing both direct and indirect queries. Direct queries explicitly queried knowledge, indirect queries showed the user options and asked which one was better.
3. Designed and implemented control nudges to affect the learned preferences in the planner. These nudges worked by modifying either configuration parameters of the planner or the

[a] https://orcid.org/0009-0005-4683-5235
[b] https://orcid.org/0000-0002-2117-5294

input to the planner, never modifying the source code of the planner.

4. Demonstrated significant performance improvement in quality of plans with respect to planner preferences in both automated experiments and in-person SME evaluations. LEAP plans were always at least as good as the legacy planner, most of the time even better.

2. BACKGROUND

Figure 1 shows a high-level view of LEAP components. The central oval represents the Learning Controller, which is responsible for the run-time management of the Interactive Learning process. The Learning Controller chooses from one or more scenarios produced off-line, with the help of our SME, or can learn online from operator inputs. As a session proceeds, Preference Learning and Query Formulator are responsible for maintaining the model of learned operator preferences and posing new queries to the SME to refine the learned model. The Query Formulator minimizes the number of queries needed using different strategies. LEAP's automated planner is Global Mission Scheduler (GMS) to automate cargo mission planning (Scott *et al.,* 2009*)*. The Problem Formulator provides the Learning Controller a mechanism to communicate preferences and control nudges to the GMS Planner for a specific planning problem. The SME from whom we are learning preferences is pictured in the upper-left quadrant and is queried by the Learning Controller with the assistance of GUIs built to support an interactive learning session. Next, we present the background on the cargo mission planning domain, GMS planning system, and an example use case.

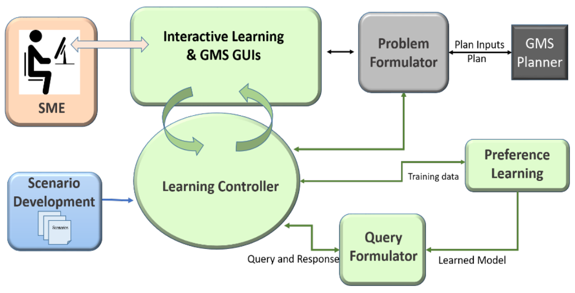


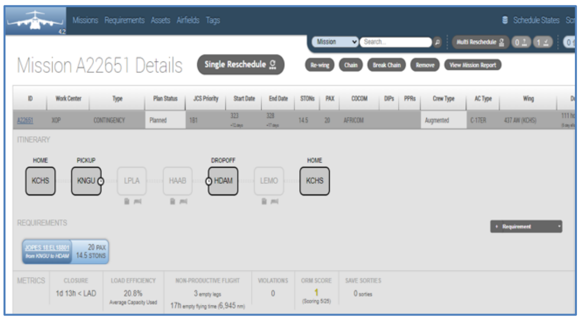
Figure 1: High-level LEAP Components with Automated Planner in gray/black, Learning Model in green, Scenario Development in blue, and Human Planning SME in orange.

2.1 Cargo Mission Planning Domain & Planner

A cargo mission starts as a response to a set of requirements, each specifying an amount of cargo (and/or passengers) to be moved from a given POE (point of embarkation) to a given POD (point of debarkation), with the load ready to be picked up on ALD (available to load date) and needing to be delivered by LAD (latest arrival date). In the plans generated, there is one mission per requirement, i.e., we are not considering aggregating cargo in a single plane. In this domain the aircraft and personnel have home stations to return after completing a mission. Each hub in the domain is an airfield identified with an ICAO code, a four-letter code designating aerodromes around the world.

In this paper, we will use the GMS planner to create the schedule and itinerary for a given mission. Itinerary generation includes planning for all the necessary refueling and crew rest stops, considering the cargo loads on each leg (to ensure the validity of the itinerary, since fuel is burned faster for heavier loads, resulting in a shorter maximum range). Examples of additional considerations are airfield capacities, windows of operation, and crew duty day limitations. GMS utilizes complex physics models to compute realistic flight times and routes based on the load size, fuel amount, fuel burn rate and runway lengths etc.

The GMS Mission Detail Display shown in Figure 2 is a visualization of a planned mission. This mission – mission A22651 (using an arbitrary naming scheme) – has a pickup of cargo at Norfolk (KNGU) followed by a drop-off of cargo at Djibouti (HDAM). The rough itinerary shows what roles each stop plays in the mission. The mission is currently planned to be operated out of Charleston (KCHS), transiting to Norfolk (KNGU) to pick up cargo, followed by stops to both refuel and rest crew (denoted by the gas pump icon and bed icon beneath the ports) in Lajes (LPLA) and Bole (HAAB) before arriving at Djibouti. A refuel and crew rest stop at Moron, Spain (LEMO) is made on the way back to Charleston. The stops that are required for this mission - the stops where cargo is picked up or dropped off (or the aircraft/aircrew originate or terminate) - are highlighted with darker borders, while the remaining stops (with lighter borders) can be replaced at will by the planner or the automated scheduler.



ID	Work Center	Type	Plan Status	JCS Priority	Start Date	End Date	STONs	PAX	COCOM	DIPs	PPRs	Crew Type	AC Type	Wing
A22651	XOP	CONTINGENCY	Planned	1B1	323	328	14.5	20	AFRICOM			Augmented	C-17ER	437 AW (KCHS)

METRICS	CLOSURE	LOAD EFFICIENCY	NON-PRODUCTIVE FLIGHT	VIOLATIONS	ORM SCORE	SAVE SORTIES
	1d 13h < LAD	20.8% Average Capacity Used	3 empty legs 17h empty flying time (6,945 nm)	0	1 (Scoring 5/25)	0 sorties

Figure 2: GMS Mission Detail Display.

2.2 Example Cargo Mission Planning Problem

We expand the example presented to illustrate the choices a planner needs to make to plan a mission, to describe how these choices might map to preferences in a simple priority-based preference model and how a few questions let us learn an expert's preferences in planning this mission.

The mission tasks a cargo plane to travel from its home base to Norfolk Naval Air Station to pick cargo, deliver it to Djibouti, and return home empty. Using GMS, we computed four alternative plans. Each one utilizes different intermediate stops for crew rest and refueling. Three mission plans use augmented crew (allowing longer non-stop flights). Table 1 summarizes some characteristics of the statistics for each plan: "Lateness" (measured in hours) is the difference between the actual delivery time and required delivery time. Lateness is zero if the cargo was delivered before or on time. "Crew Type" is either basic or augmented), "Rest Quality" (desirability rating), and finally "Flying Hours" the time in flight which is used to estimate cost.

The "best" option for this mission depends on the operator's preferences. If the SME has the preference F1 > F2 > F3 > F4, then he chooses option 3 as the best plan. If the SME has the preference F2 > F1 > F3 > F4, then he chooses option 4, with the basic crew, even though the aircraft will be out for an extra day. If the SME is more concerned about crew rest locations, then he might have a preference such as F3 > F1 > F2 > F4, which ranks option 2 as highest. Even from one example, it is possible to learn, at least partially, what the SME's preference might be. For example, if the SME chooses option 3 over all others, assuming less cost is preferred, LEAP can deduce that the most important features are F1 and F2. Using this partial model when planning a similar mission, LEAP tasks the GMS planner to produce a plan with the qualities of an augmented crew and the shortest time away for the aircraft and crew.

3. PREFERENCES AND LEARNING

Initially, we represented preferences using lexicographic preference models (LPMs), which enforced a total order on the attribute space (Colman and Stirk, 1991, Ford *et al.,* 1989, Westenberg and Koele, 1994). LPMs also generalizes to continuous valued variables by taking advantage of monotonic properties. For example, we could represent that shorter flight time is preferred over longer flight time. Priority-based preference examples discussed in previous section are all LPMs. Later on, we extend our preference models with a trade-off representation by combining the existing LPMs with a decision-tree-like conditional representation.

Learning Algorithm: We utilized the CHARM algorithm (Yaman *et al.* 2011) to learn the LPMs. CHARM kept track of a set of LPMs consistent with all the training data, which was a set of pairwise comparisons on plan features. For example, Table 1 represents six pairwise plan comparisons for CHARM.

Initially, CHARM considers all features as equally important (rank of 1). At every iteration and for every pair, CHARM predicts a winner as follows: Among the features that were different in the comparisons, the ones that have the smallest rank (the most salient) votes to choose the preferred option. If the CHARM prediction was correct (the option with the most votes wins), then the ranks stay the same. Otherwise, the ranks of the features that voted for the wrong option were incremented, thus reducing their importance. CHARM loops over the set of pairwise comparisons until the ranks converge. Correctness and convergence of the algorithm is guaranteed if the

Table 1: Mission Plan Options with Respect to Defined Features of a Learned Model.

Plans	F1: Lateness	F2: Crew Type	F3: Rest Quality	F4: Flying Hours
#1	1	Augmented	Reasonable	41
#2	2	Augmented	Preferred	42
#3	0	Augmented	Reasonable	43
#4	4	Basic	Undesirable	44

training data is noise and tie free. For LEAP, we assumed that the SME choices were consistent, thus the input to preference learning was noise free.

3.1 Trade-off Trees

SME feedback highlighted the inflexibility of the LPMs, which required we extend our preference model representation. For example, if our SME wanted to express: "Even if flight-time is more important than lateness, if less than 0.5 hour gain in flight time is causing more than 24 hours delay then flight-time should be ignored in favor of lateness." In achieving this extension, we aimed for computationally feasible and explainable representations.

Our trade-off-based models use a hybrid representation of decision trees and LPMs. Such hybrid representations represent both conditional preferences and trade-off concepts. Figure 3 is an example trade-off tree where original LPM is in green and a truncated LPM which ignores the flight-time is in purple. Depending on the maximum lateness when comparing a pair of plans, and the difference in flight time and lateness, we might have ended up using either the original LPM or the truncated one. In a trade-off tree for two attributes A and B, where A is declared more important than B in general, the intermediate nodes could have branching conditions on one of the following:

- A(+): Best value for the attribute A when comparing two individuals, in this case two plans.
- B(+): Best value for the attribute B.
- diff_A: Absolute value of difference in A values in compared individuals.
- diff_B: Absolute value of difference in B values in compared individuals.

While in theory we could learn tradeoffs between every consecutive attribute in an LPM, our interactions with the SME led us to believe that the most important tradeoff relationship was between the first and the second attributes in an LPM. Thus, we focused on demonstrating the learning and representation of tradeoff trees for the most important two attributes only. Learning of tradeoff trees was triggered when the planner picked a seemingly inferior plan with respect to the most important attribute as learned by the LPM, but superior with respect to the second most important attribute (see Figure 4). Whenever the SME made such a choice,

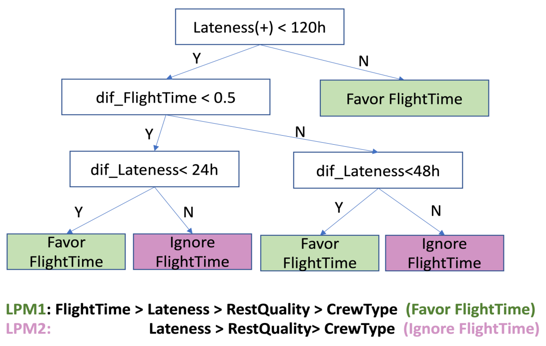
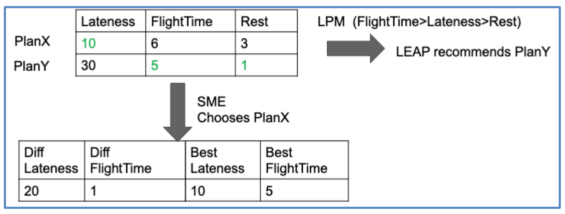


Figure 3: Trade-off tree representation.

we created a vector with four values computed from the original pair of plan attributes, and labeled it as an example of ignoring the most important attribute. The next section explains the algorithm that took these data points and built a trade-off tree.



	Lateness	FlightTime	Rest
PlanX	10	6	3
PlanY	30	5	1

Diff Lateness	Diff FlightTime	Best Lateness	Best FlightTime
20	1	10	5

Figure 4: Generating trade-off tree attributes from LPM.

3.1.1 Learning trade-off trees

Our strategy in learning trade-off trees was based on tracking the SME interactions with the LEAP ranking GUI. Whenever the SME reordered the ranked mission plans, LEAP recorded this exception and generated data points (as explained in the previous section) to feed into the well-known decision tree building algorithm, C4.5. Using C4.5, we built a tree incrementally as the user kept adding more exceptions. As the exceptions updated in the trade-off model, LEAP used the new model to rank the next set of plans, thus, providing a life-long learning set up. For our experiments, we limited the exception recording and learning sessions to three sets of missions. This was mostly to ensure consistent interaction with the SME.

One of the weaknesses of the C4.5 algorithm was the boundary conditions computed for numeric nodes being limited to the values that are seen in the training data set. While we could have employed a search-based method to pinpoint the exact threshold value for these nodes, we decided, for the sake of not bombarding the user with a series of implicit questions, to ask the user directly to set the boundary conditions. From our experience with SMEs, the user would already have a trade-off value set in their mind. A binary search over the value space just to prove that we could eventually learn the correct value would only frustrate the user and violate one of our main goals. Thus, after the learning step was completed, we presented the user an interactive trade-off tree visualization where they could adjust the node conditions. As the user adjusts the values (overwriting previous tree), we re-compute the trade-off tree and updated the visualization.

Sorting with trade-off trees could be tricky due to loss of transitivity between pairwise comparisons when pairs follow different branches. Loss of transitivity could lead to circular ordering graph which meant individual schedules could not be sorted. To overcome this challenge, we implemented a scoring function that counted the pairwise wins when soring a set of plans. The tie breaker if there were equal scores was the original LPM.

4. CONTROL: NUDGING ALGORITHMS

We started with two kinds of nudges to make GMS produce user preference aligned solutions: 1) modify inputs (ΔS for modifications in scenario), and 2) set configuration parameters (ΔC for modifications in

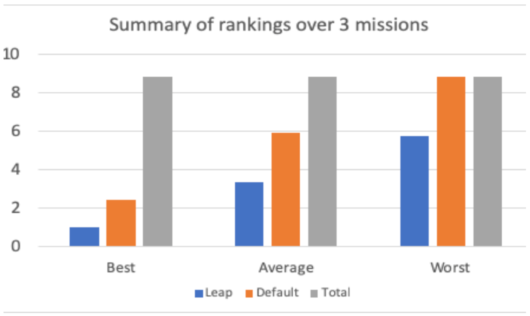


Figure 6: Summary of rankings over three missions.

configuration) that would bias GMS towards desired mission plans. Before discussing methods for nudging (i.e., computing ΔS and ΔC), we will summarize the GMS algorithm, which has two-steps. In the first step, GMS computes the best stop over locations for a pair of pick-up and drop-off locations. This step uses an A-* algorithm where the guiding metric is the total time spent on the route. The amount of time spent on a rest stop is penalized if the rest stop is not a favored base. Next, the landing and takeoff of each leg in the route is computed, taking into account availability of resources on the bases. The ingrained biases in GMS algorithm order might not necessarily align with preferences of the user creating challenges in aligning GMS output with the user's preferences. The following is a list of nudging methods for single attributes:

Lateness: Add a landing constraint for destination: *LandingTime<= dueDate.* If there is no solution, then relax the constraint by 6 hours until a solution is found (**ΔS**).

Rest Quality: Reduce the time-delay in in the first-tier bases to further distinguish them from lower tiers (**ΔC**).

FlightTime: Set the time-delay in all stop over bases to zero so the flight time dominates the total time spent (**ΔC**).

CrewType: GMS has the option to choose the crew type as input. If there is no solution for the preferred type, then switch the value (**ΔS**).

We used the following generic algorithm shown in Figure 5 for biasing GMS results. In this domain, the Lateness and CrewType nudge could be combined with every attribute because they added more constraints. However, the FlightTime and RestQuality nudges would be mutually exclusive because they were part of the same optimization function. Thus, we could only nudge the most important one with respect to the LPM, while the other could be nudged individually. This meant that given an LPM with the four attributes listed above, we could at most combine 3 of them.

Algorithm 1 Algorithm to nudge GMS for LPM $A_1 > ... > A_n$

```
1:  Solutions S = $\epsilon$
2:  Results R = Run GMS with $A_1$ nudge
3:  S = S + R
4:  for i = 2 to n do
5:      Nudges N = $A_i$ nudge
6:      for j = 1 to i-1 do
7:          if nudge $A_j$ is not mutually exclusive with any in N then
8:              Nudges N = N + $A_j$
9:          end if
10:     end for
11:     Results R = Run GMS with nudges N
12:     S = S + R
13:     Nudges N = $\epsilon$
14: end for
15: sort(S)
16: Return highest ranking K solutions
```

Figure 5: Algorithm for composing nudges to compute solutions with arbitrary LPMs

To extend the algorithm for use with trade-off trees, we keep the nudging algorithm the same, but utilize the trade-off tree for sorting final set of schedules.

5. EXPERIMENTS AND RESULTS

This section provides the experiment methodology and results evaluating LEAP performance. These experiments ranged from fully automated to human-in-the-loop, and also from domain-specific to abstract, to demonstrate the generalizability of the results. In this paper we will present results of our fully automated experiments due to space constraints.

To evaluate the plan quality, we compared the rankings of two planners; where the Default plan generator was unaltered GMS software and the other planner was LEAP with ability to nudge GMS to produce solutions more aligned with a given LPM. For any given mission requirement and an LPM we ran both planners to get the best ten (or the size of the smallest set of plans returned by any planner) plans according to target LPM. We then combined the best solutions into one set and ranked them, keeping track of the pedigree of the solutions. Our hypothesis was the LEAP plans, on average, would have better rankings than default GMS plans.

5.1 Single Attribute Nudge Experiments

Initially, we only performed nudging on the most important attribute of an LPM. We also performed the experiments for 5 different missions and all possible LPMs with our four attributes to demonstrate some generalizability of the results. Figure 6 shows the average of the best, average, and worst rankings over three of the 5 missions only. This is because for two of the five missions, LEAP did not produce a different set of solutions than GMS. Further investigation revealed that these two missions had a very tight solution space and not much flexibility in producing viable mission plans. For these two missions, different solution sets were produced only when FlightTime was the most important attribute. This was because the nudging method for FlightTime actually relaxed the constraints.

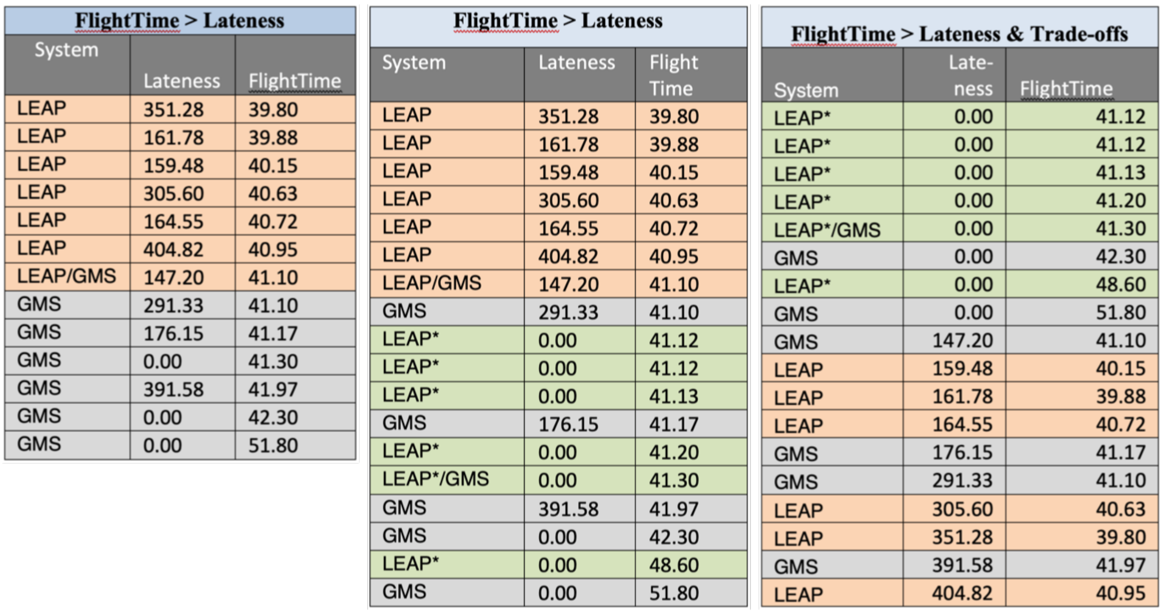
For the other three missions, LEAP produced consistently diverse solutions and the best schedule was always LEAP (Rank #1). Furthermore, LEAP schedules were never ranked worse than GMS.

When we analyzed the difference in plan quality per LPM, we discovered the following results: The difference in plan quality was most evident when the FlightTime attribute was the most important attribute in an LPM. In such cases, LEAP plans always outranked the GMS plans. However, this was not the case when RestQuality was the most important attribute. Often, GMS and LEAP shared the top ranking and sometimes even more than just the top slot. This is because GMS was biased to produce plans with best RestQuality.

5.2 Multi-Attribute Nudge Experiments

We ran experiments, where we combined multiple nudges using mission A22669, which was chosen due to the high diversity in solutions. Several cargo planes across many wings are available for this mission in the input scenario allowing many types of solutions. We computed schedules for all LPMs with multiple nudging. In most but FlightTime dominant (i.e., FlightTime being the most important attribute) LPMs, the multi-nudging further improved the quality of the plans. However, that was not the case for FlightTime dominant LPMs. In Figure 7 we compare the plan quality for LEAP with single-nudge and multi-nudge using FlightTime>Lateness>RestQuality>CrewType LPM. The left table in Figure 7 shows the ranking of GMS plans and plans produced by LEAP with FlightTime nudge only. Clearly the LEAP plans have better flight time, however they are also extremely late. The middle table compares new plans found using multiple nudges (LEAP*) in addition to single-nudge lateness, As seen in the middle table LEAP* plans are ranked worse than LEAP (single nudge). Using only strict preference models, to gain 5 minutes of flight time, the cargo could be delayed for days. Normally, the user would prefer to minimize flight time over lateness, but if the time difference between two missions was small, but the savings in Lateness was large, then switching preferences to favor Lateness was justified.

To reflect that conditional preference, we employed the trade-off tree in Figure 3 (obtained from a learning session with our SME) to re-rank the schedules. The right table in Figure 7 shows the re-ranked elements clearly demonstrating a more balanced lateness and flight-time in the top ranked elements which are products of multiple nudges.



FlightTime > Lateness

System	Lateness	FlightTime
LEAP	351.28	39.80
LEAP	161.78	39.88
LEAP	159.48	40.15
LEAP	305.60	40.63
LEAP	164.55	40.72
LEAP	404.82	40.95
LEAP/GMS	147.20	41.10
GMS	291.33	41.10
GMS	176.15	41.17
GMS	0.00	41.30
GMS	391.58	41.97
GMS	0.00	42.30
GMS	0.00	51.80

FlightTime > Lateness

System	Lateness	Flight Time
LEAP	351.28	39.80
LEAP	161.78	39.88
LEAP	159.48	40.15
LEAP	305.60	40.63
LEAP	164.55	40.72
LEAP	404.82	40.95
LEAP/GMS	147.20	41.10
GMS	291.33	41.10
LEAP*	0.00	41.12
LEAP*	0.00	41.12
LEAP*	0.00	41.13
GMS	176.15	41.17
LEAP*	0.00	41.20
LEAP*/GMS	0.00	41.30
GMS	391.58	41.97
GMS	0.00	42.30
LEAP*	0.00	48.60
GMS	0.00	51.80

FlightTime > Lateness & Trade-offs

System	Late-ness	FlightTime
LEAP*	0.00	41.12
LEAP*	0.00	41.12
LEAP*	0.00	41.13
LEAP*	0.00	41.20
LEAP*/GMS	0.00	41.30
GMS	0.00	42.30
LEAP*	0.00	48.60
GMS	0.00	51.80
GMS	147.20	41.10
LEAP	159.48	40.15
LEAP	161.78	39.88
LEAP	164.55	40.72
GMS	176.15	41.17
GMS	291.33	41.10
LEAP	305.60	40.63
LEAP	351.28	39.80
GMS	391.58	41.97
LEAP	404.82	40.95

Figure 7: GMS, single-nudge (LEAP), multi-nudge (LEAP*) comparisons with and without the and trade-off.

6. CONCLUSIONS

In this paper we presented the LEAP system that learns and then applies user preferences to an automated logistic planner to produce plans that are more acceptable to users. The goal of the LEAP project was to demonstrate control over a fixed legacy mission planning system to produce plans that were better aligned with planners' preferences, which were elicited through minimal querying to ensure continued use and applicability of existing systems by assuring they could be adapted to current needs and priorities of the stakeholders.

The key contribution of this LEAP paper is to demonstrate the ability to nudge the automated planner without changing its code, through externally operating nudges. While implementations of nudges were specific to the GMS planner, we identified generic classes of nudges for controlling automated planners such as global configuration parameter setting and input modification. Designing nudges to influence the GMS planner to produce a plan in accordance with those preferences was the most challenging part of the project. In this effort, nudges for each preference attribute were designed manually. We succeeded in designing an algorithm to combine arbitrarily complex trade-off and priority-based preferences. In LEAP, we could rationally design nudges because we had the domain and planning algorithm expertise. If the planner is truly a black box, a machine learning approach to analyze the interaction space is recommended.

In our experiments, we demonstrated that the best LEAP plans were always as good as the best GMS plans, and most of the time better. Due to space constraints, we demonstrated the efficacy of LEAP through automated experiments.

Finally, we believe LEAP control approaches are applicable to beyond GMS planner. There are many deployed systems that have been certified as part of their development and deployment and, thus, are fixed in their operation without the benefits (and uncertainties) of on- or off-line learning. Applying LEAP nudging strategies on other search-based planning and scheduling systems, such as HTN planners, and other domains would demonstrate the general applicability of our approach and show how the life of deployed systems can be extended. This will undoubtfully be valuable to many government and civil agencies that have been using legacy planning systems for decades.

AKNOWLEDGEMENTS

Work sponsored by AFRL under contract FA8750-21-C-1005; the views and conclusions contained in this document are those of the authors and not AFRL or the U.S. Government.

REFERENCES

Colman, A.M. and Stirk, J.A. (1991). Singleton bias and lexicographic preferences among equally valued alternatives. *Journal of Economic Behavior & Organization*, **40**(4):337–351.

Ford, J.K., Schmitt, N., Schechtman, S.L.,Hults, B.M., and Doherty, M.L. (1989). Process tracing methods: contributions, problems and neglected research issues. *Organizational Behavior and Human Decision Processes*, **43**:75–117.

Sacerdoti, E.D. (1977). *A Structure for Plans and Behavior*. Elsevier.

Scott, R., Roth, E., Truxler, R., Ostwald, J., & Wampler, J. 2009. Techniques for Effective Collaborative Automation for Air Mission Replanning. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, **53**(4), 202–206.

Westenberg, M.R.M. and Koele, P. 1994. Multi-attribute evaluation processes: methodological and conceptual issues. *Acta Psychologica*, **87**:65–84.

Yaman, F., Walsh, T.J., Littman, M.L., desJardins, M. 2011. Democratic approximation of lexicographic preference models. *Artificial Intelligence*, **175**(7-8): 1290-1307

Zweben, M. and Fox, M. 1994. *Intelligent scheduling*. Morgan Kaufmann Publishers Inc.

DISTRIBUTION A. Approved for public release: distribution unlimited. Case Number AFRL-2023-5571. This document does not contain technology or Technical Data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.