Planning, Scheduling and Monitoring for Airport Surface Operations

Robert Morris[1] Corina Pasareanu[2] Kasper Luckow[3] Waqar Malik[4] Hang Ma[5] K. Satish Kumar[7] Sven Koenig[8]

***Abstract*— This paper explores the problem of managing movements of aircraft along the surface of busy airports. Airport surface management is a complex logistics problem involving the coordination of humans and machines. The work described here arose from the idea that autonomous towing vehicles for taxiing aircraft could offer a solution to the 'capacity problem' for busy airports, the problem of getting more efficient use of existing surface area to meet increasing demand. Supporting autonomous surface operations requires continuous planning, scheduling and monitoring of operations, as well as systems for optimizing complex human-machine interaction. We identify a set of computational subproblems of the surface management problem that would benefit from recent advances in multi-agent planning and scheduling and probabilistic predictive modeling, and discuss preliminary work at integrating these components into a prototype of a surface management system.**

I. Background and Motivation

Congestion at airports is recognized as one of the most prominent problem areas in the international commercial airspace. In particular, increasing the capacity of surface area used for taxiing is a major logistical challenge. The normal approach to increasing capacity by expanding the surface area used for taxiing has a number of problems, including harmful impact to the environment (increased noise and pollution) as well as adding to human workload and increasing the overall complexity of operations. These difficulties of increasing capacity through airport expansion motivate the search for solutions that make more efficient use of existing space rather than creating new space. Automated tools on the ground or in the control tower are being matured and integrated into decision support systems. At the same time, NASA and other government agencies are encouraging the development of 'game changing' technologies for radically transforming future airspace operations.

In the spirit of 'changing the game', we envision a future in which automated surface movement management in the control tower is combined with autonomous aircraft towing vehicles. Specifically, by autonomous engines-off taxiing, we mean a taxiing system involving a towing vehicle that will, on command, autonomously navigate to an assigned aircraft, attach itself, tow the aircraft to an assigned location (a runway for departures, a gate for arrivals), autonomously detach itself, and navigate to an assigned location, either a staging area or to service another aircraft' [1]. The focus in this paper is not on the problem of autonomous taxiing, but rather the automation in the tower required to support autonomous operations. Specifically, we identify a set of sub-problems in planning, scheduling, and execution monitoring, that combine to provide a system for continuous surface operations. The main sections of the paper consist of an overview of the approach, a discussion of the planning and scheduling components, and a monitoring device that is based on recent advances in predictive analysis.

II. Overview of Approach

Airport surface operations consists of all the activities required for handling arrivals and departures of commercial aircraft. In principle this collection of activities includes servicing tasks such as refueling or loading/unloading baggage, but here the focus is on the movement of aircraft: in the departure phase, movements consist of pushback, navigation to the entrance to the taxiway (called the spot), taxiing and takeoff; in the arrival phase, movements consist of landing, entering the taxiway, entering the ramp area, and navigating to gate.

This work was supported by grants from NASA and the Air Force Research Lab (AFRL). This work was conducted as part of a combined effort with Northrop Grumman Corporation (NGC).

[1] Robert Morris is a researcher at NASA Ames Research Center robert.a.morris@nasa.gov

[2] Corina Pasareanu is a researcher at NASA Ames and Carnegie Mellon University corina.s.pasareanu@nasa.gov

[3] Kasper Luckow is a postdoctoral researcher at Carnegie Mellon University kasper.luckow@sv.cmu.edu

[4] Waqar Malik is a research scientist at the University Affiliated Research Center, Moffett Field, CA 95035. waqarmalik@gmail.com

[5] Hang Ma is a PhD student at the University of Southern California hangma@usc.edu

[6] T. K. Satish Kumar is a Research Scientist in the Computer Science Department at the University of Southern California tkskwork@gmail.com

[7] Sven Koenig is a professor at the University of Southern California skoenig@usc.edu

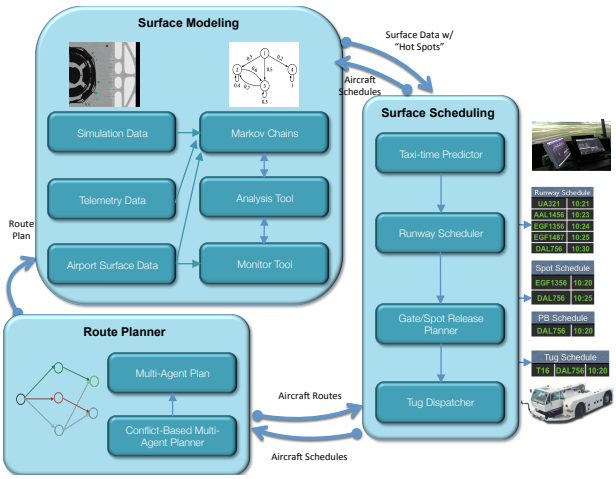

Fig. 1. System architecture

Airport surface operations is a complex logistics problem, involving the coordination of potentially large numbers of humans and machines. The two main criteria for effective operations are safety and efficiency. At the center of operations is ramp or Air Traffic control. Currently tower operations for surface movement is heavily based on human controllers interpreting surveillance data from surface detection equipment such as ASDE-X, and communicating routing information to pilots. Although the operating environment is based on rigid rules and procedures, there is much uncertainty in surface movement, primarily in the form of delays due to changes in the operating environment. Uncertainty is currently handled through continuous replanning and rescheduling movements by human controllers to respond to delays or other contingencies.

Here we propose a new design for tower operations based on automation. We focus on two areas in which automated decision making could assist the human controller in more safe and efficient operations: data analysis for behavior modeling and route planning and scheduling. The overall system is visualized in Figure 1. The three main components of the system are *models*, which are of two kinds: topological and behavioral; *route planning*, and *surface scheduling*, comprised of the following sub-problems: runway sequencing and scheduling [2]; spot or gate release scheduling [3]; gate allocation [4] and taxi route planning and scheduling [5]. To this list we add towing vehicle dispatching for autonomous towing vehicles [6].

Surface movement optimization is NP-hard [7]. Several types of constraints are involved, including push-back times, taxiway layouts, and runway and taxi-way separation. Planning is dynamic, with aircraft continuously entering and leaving the planning space, and replete with uncertainty and unexpected events. These complexities and the dynamic nature of the environment motivate approaches to automated planning that require reduced computational overhead while achieving useful results.

III. Planning and Scheduling

Surface planning and scheduling with autonomous towing vehicles is viewed here a centralized process, performed by a decision-support tool used by ramp controllers, or tower (ATC) operators. The airport ground routing and scheduling problems require directing aircraft to their destinations in a timely manner, with the aim being to reduce the overall travel time, delays at the runway queue, and to maximize throughput. Separation constraints between taxiing aircraft maintain safety. Whereas for smaller airports the routing problem can be simple to solve, for larger airports, especially during peak hours, the interaction between the routes of different aircraft often requires the application of a sophisticated routing algorithm.

The overall approach to planning and scheduling towing vehicle-based surface operations is an extension of the Spot and Runway Departure Advisor (SARDA) approach [3]. The SARDA scheduler addresses the highly dynamic and uncertain planning environment by a multi-stage process. The next paragraphs summarize this process.

A. Route Planning

Airport surface route planning has received some attention in the AI and OR communities [8], [9], [10] [11]. In anticipation of autonomous towing in future operations, we consider here the application of recent approaches to multi-agent path finding (MAPF) for route planning of towing vehicles. MAPF seeks a set of conflict-free paths for a set of agents. Although NP-hard to solve optimally [12], many powerful algorithms have been presented for solving MAPF. Among them are solvers based on reduction to other problems (SAT, ILP, ASP) [13], [14], [15], ad-hoc enhancements to the A* algorithm [16], [17], [18], unique state space representations [19], [20] and other techniques for non-optimal solutions [21], [22], [23]. In particular, the Conflict-Based Search (CBS) algorithm [20] is considered state-of-the-art for many MAPF domains.

MAPF is a discrete time planner: the output is a set of synchronized paths that assign each agent to a location at each discrete time step. An example of an MAPF instance and output on a grid-shaped problem

is illustrated in Figure 2. Although an airport is not a grid, using a strategic placement of obstacles, a grid can be transformed into a general graph. We can made this transformation in order to create a graphical representation of part of Dallas Fort Worth airport (DFW); see Figure 3.

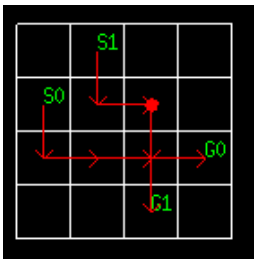

Fig. 2. An instance of MAPF with start and goal locations being 13 and 22 for Agent 0, and 8 and 27 for Agent1. The MAPF solution generated by the CBS algorithm is: path of Agent 0: $\{13, 19, 20, 21, 22\}$; path of Agent 1: $\{8, 14, 15, 15, 21, 27\}$. The solution is no longer feasible to execute if Agent 1 arrives at location 21 from location 15 before Agent 0 gets to or leaves location 21.

B. Dispatching and Scheduling

The SARDA scheduler contains three main components: a runway sequencer and scheduler; a spot and gate release scheduler; and a towing vehicle dispatcher. The spot and gate release scheduler selects times for pushback from the gate, and times for releasing the towing vehicle/aircraft for entry into the taxiway (the spot is the entry point into the taxiway from the ramp area). GIven a multi-agent plan generated by the MAPF, the scheduler continuously controls the scheduling of towing vehicle movements.

The inputs to the scheduler consist of the current snapshot of the airport (the current locations of each active towing vehicle on the surface), scheduled push back and arrival times for some time into the future (currently, the next 15 minutes), and various constraints such as aircraft-specific parameters and separation constraints. To handle uncertainty in surface dynamics, these inputs are refreshed every few seconds. To control the number of changes made to the outputs of the schedule, a freeze horizon is imposed which precludes major changes to be made to the current schedule.

The outputs of the scheduler, are, in fact, three schedules: a runway schedule, a spot and pushback schedule, and a towing vehicle schedule, which are communicated to the towing vehicles.The times computed by the scheduler represent each vehicle's earliest possible arrival time at each node. However, this set of routes may contain numerous conflicts (separation constraint violations). To resolve such conflicts, the system contains a flow model and a network event simulator to model arrivals at nodes representing intersections, to determine the amount of time that aircraft must hold at current locations to maintain separation requirements, and to ensure other safe conditions (e.g. at intersection crossings, or to maintain wake vortex separation). The flow model assumes conflict avoidance on the surface to be the combined responsibility of the controller and towing vehicle. The controller identifies spatial violations in the schedule such as aircraft approaching head on. The towing vehicle determines possible conflicts at the node it is currently approaching, and adjusts its speed accordingly. Together, the scheduler and deconfliction model approximate the taxi routings and resource utilization (gates and runways) that are most likely to be used by tower controllers.

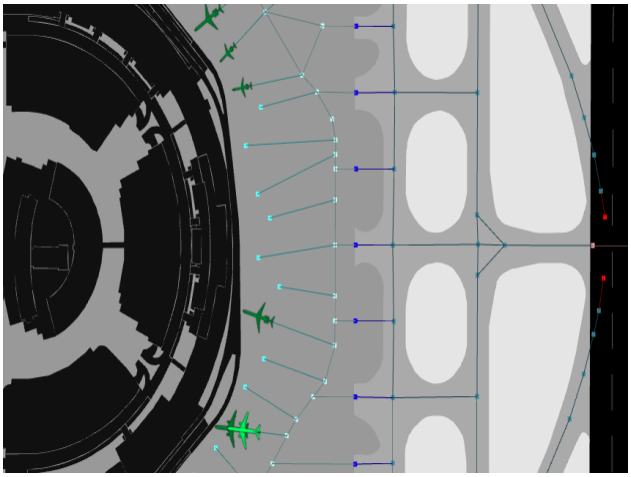

Fig. 3. Graphical representation of surface of part of Dallas Fort Worth Airport

A towing vehicle dispatcher is a kind of resource scheduler: given an available towing vehicle, and an aircraft that needs to be towed, the dispatcher assigns the towing vehicle to the aircraft, and generates a shortest-path route for the towing vehicle to navigate to reach the assigned craft. Ordering the available towing vehicles to determine the most efficient allocation can be decided using different criteria. We currently use a simple shortest distance criterion: the available towing vehicles are ordered by distance between towing vehicle and attachment point (i.e. gate or runway exit), and the one with the smallest distance is assigned. A subset of nodes in the graph are designated as towing vehicle depots that provide a re-charging station and designated locations for dispatching idle towing vehicles. Towing vehicle depots should be strategically placed along the surface

to reduce the time between dispatching an idle towing vehicle and reaching its assigned aircraft for attachment. Towing vehicles can also be dispatched from locations other than depots; for example, a towing vehicle might have completed a towing operation to one gate, and be then dispatched to a nearby gate for the next departure towing task. Problems similar to the dispatching problem have been studied in the AI literature; for example, [6].

C. Discussion

A discrete time approach to route planning has a number of limitations: first, it requires the synchronizion of the movements of all agents perfectly, which can be difficult; second, the size, velocity, safety margin and other properties of each of the towing agents are not uniform, and this approach abstracts away these differences; finally, and most importantly, surface operations are laden with temporal uncertainty and the concrete plans generated by an MAPF algorithm do not allow for temporal uncertainty. As a result, the execution of the synchronized discrete MAPF solutions in environments with such complex dynamics involves continuous execution modeling and potential replanning when the MAPF solutions are no longer executable.

Simple Temporal Networks (STNs) have been used to represent temporal flexibility in plans. Here we propose an approach that takes a synchronized discrete solution generated by MAPF algorithms as input and generalizes it to a continuous routing plan using STNs. A centralized executive and monitoring system can then use the STN to dispatch towing vehicles and incrementally assign times for vehicle locations along their assigned route. The main idea is to consider only those locations visited by more than one agent at different time steps. It is essential for the continuous routing plan to respect the order of visitation by the agents to the same location given by the MAPF solution, since the consistency of the order of visitation is sufficient for the MAPF solution to remain feasible to execute. This idea is demonstrated in Figure 2. The partial order obtained in this way can then be represented as a temporal plan graph in which a vertex corresponds to the event of an agent visiting a location, and a directed edge corresponds to the temporal precedence between two events. This temporal plan graph allows us to add additional temporal constraints between events. By analyzing this temporal plan graph in the simple temporal problem (STP) framework, we can efficiently reason about the time interval that any given event can occur in, which provides a principled formalism for performing probabilistic reasoning at any time during the execution of the routing plan. An example of a STN corresponding to the MAPF instance shown in Figure 2 is found in Figure 4.

In this section we have sketched a system for a hybrid centralized system for airport surface route planning and scheduling and distributed execution using autonomous agents. A prototype of the SARDA-based planning and scheduling system has been implemented in Python and C++, along with a simulator of DFW terminal behavior; see [1] for details about the implementation.

The main challenge for such a system is handling the uncertainty on the airport surface, as the result of interference with other surface vehicles, delays in communication, changes in weather, and unexpected changes in traffic volume in the air. Our current procedure combines simple pre-defined route planning with continuous scheduling and monitoring to handle uncertainty. The efficiency of operations can be improved upon, we claim, with more sophisticated planning and modeling of surface behavior. This section has illustrated the use of STNs to represent the temporal flexibility inherent in the problem. In the next section, we describe work on probabilistic modeling and analysis for increasing the predictive capabilities of a surface management system.

IV. Modeling and Predictive Analysis

To address the temporal uncertainty on the airport surface we propose a behavioral model which is automatically inferred from telemetry data, log files and simulation data available from previous or similar operations at the airport. The model uses a grid abstraction of the surface as the basis for a Discrete Time Markov Chain (DTMC). The model can be analyzed using temporal logic queries to obtain predictions about the likelihood that temporal constraints for avoiding conflict will be violated. This information can be used by humans or automated tools to monitor surface behavior or alter plans. This section describes the creation and deployment of the model for predictive analysis.

A. Model Inference

The goal of the model inference component is to build "behavioral models" of surface operations (for both towing vehicles and airplanes within a specified airport). Such models contain key information that enable analysis with respect to safety, delays, throughput etc. One can use such models to optimize planning decisions for minimizing delays in taxiing and avoiding congestions.

Our models are DTMCs, i.e. automata labeled with outgoing probabilities on their transitions. We infer these models from time series data; each step encodes the value of the states observed at each time step. Thus the states of the inferred model represent "abstractions"

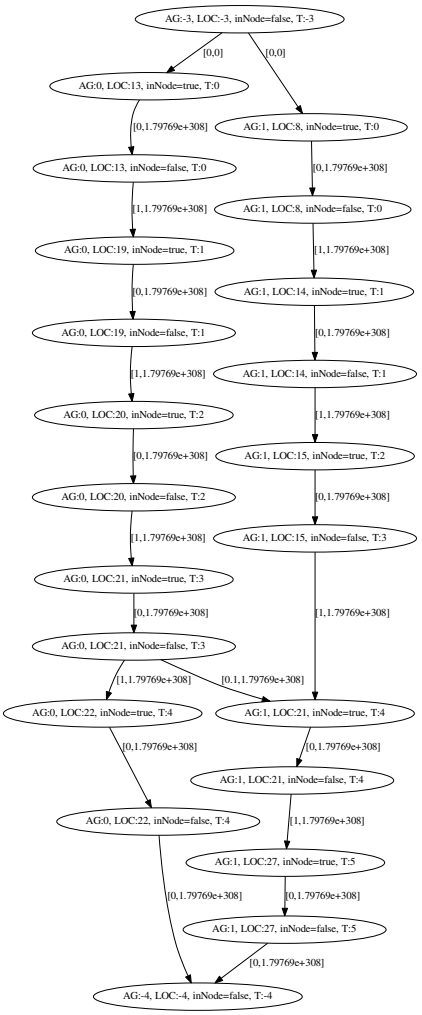

Fig. 4. A temporal plan graph generated from the MAPF instance shown in Figure 2. Each vertex corresponding to the event "j moves into loc" indexed by t is labeled as "AG:j, L:loc, inNode=true, T:t" in the figure, while each vertex corresponding to the event that "j moves out of loc" indexed by t is labeled as "AG:j, L:loc, inNode=false, T:t" in the figure. X_{Start} has all labels being -3 and false. X_{Finish} has all labels being -4 and false. The minimum makespan of the continuous routing plan, that equals the earliest time for the event X_{Finish}, is 4.1. Agent 1 can move into location 21 0.1 time units after Agent 0 moves out of location 21.

of the state reported in the log file and transitions in the model correspond to the time steps in the log file. The abstraction chosen depends on the properties of interest. The log data is discretely sampled, in some cases many times per second; therefore it is necessary to select a resolution to allow for realistic state transitions and to prevent state space explosion. The probability distribution for a particular state is estimated by computing the ratio between the number of traversals for each outgoing transition and the total number of traversals of the transitions exiting states; this corresponds to the maximum likelihood estimator for the probability distribution at that state.

The abstraction we used is based on dividing the airport surface into a grid. For each grid a "count" of the number of vehicles (airplanes, towing vehicles) is stored. The counts of the grid elements at a specific discrete time constitute a state in the generated model. Using this abstraction, we can either build a single model for the whole surface operations or we can build several towing vehicle-centric models which capture the operations of each towing vehicle separately, together with the interactions with other vehicles in its immediate vicinity. The advantage of this latter approach is that the models are smaller and therefore easier to build, while at the same time being more precise.

We have developed a prototype tool, called LOG2MODEL, written in Java that implements our approach. It builds models with different levels of abstraction, allowing for the adjustment of the granularity of abstraction: decomposing the airfield into a coarse (fine) grid is likely to yield a less (more) precise model at the expense of a smaller (bigger) state space. Further, as noted, one can build models capturing the activities of all the vehicles on the surface, or models focused on specific vehicles. The generated DTMC can be visualized (via DOT files) and analyzed automatically using a translation to the modeling formalisms of the PRISM [24] and UPPAAL [25] model checkers.

B. LOG2MODEL

Figure 5 provides an overview of LOG2MODEL. It is comprised of three main components: a log parser, an intermediate model generator, and a model generator. The modularity, along with the design approach that provides several extension points, facilitates easy experimentation with pre-processing and abstraction techniques and to define translations from a generic (intermediate) model to different modeling formalisms.

The log parser processes the input log, and uses definitions of a state and the transition function to generate the

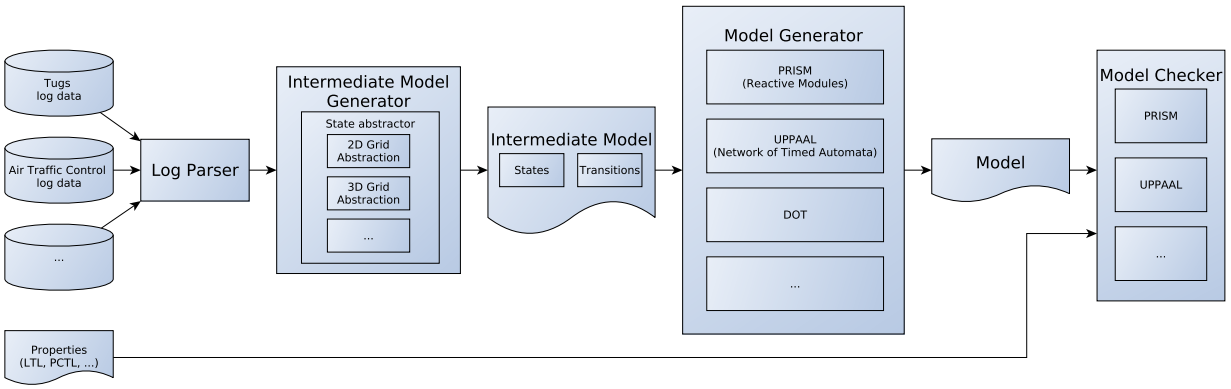

Fig. 5. Tool architecture overview.

intermediate model. The intermediate model is domain agnostic and defined as a generic automaton that allows to generate input models for different model checkers, e.g., reactive modules for the PRISM model checker. A state is an assignment of values to model variables and transitions can have attributes/labels attached. The attributes are used for defining DTMCs. Translating the intermediate model requires mapping states and transitions from the intermediate model to the target modeling formalism.

Currently, LOG2MODEL supports surface log data from Dallas Fort Worth airport. In addition, LOG2MODEL supports several variations of grid abstractions applicable for abstracting "positional" log data. Among them, LOG2MODEL has 3D and 2D grids with homogeneous and heterogeneous grid sizes. We provide translations from the intermediate model to the modeling formalisms of the PRISM model checker (reactive modules) and to the UPPAAL model checker (Networks of Timed Automata). We also provide a translation to DOT for model visualization and to facilitate systems understanding and debugging. Figure 6 shows an example model for a 2x2 grid abstraction.

1) Model Analysis: The models are analyzed based on queries written in temporal logics. For a data log recording the positions of more than 30 autonomous towing vehicles and airplanes each second, for 70 minutes of activity, we analyzed 123 MB of data in less than 5 seconds. The generated models have 75 states (10x2 grid abstraction) and 96 states (4x4 grid abstraction), respectively. For the corresponding DTMC generated as reactive modules for PRISM, we analyzed example qualitative and quantitative (PCTL) properties:

- $\mathbf{P < 0.6 \quad [F < 50 \quad q0 < 30]}$: *The probability that less than 30 towing vehicles/airplanes are present in quadrant 0 within the first 50 time units (seconds), is less than 0.6.*
- $\mathbf{P =? \quad [F < 300 \quad q0 > 33]}$: *What is the probability that more than 33 towing vehicles/airplanes are present in quadrant 0 within the first 300 time units (seconds)?*

Verifying the properties takes less than a second, respectively returning *true* and 0.1058.

As hinted, different partitioning schemes of the grid affect the resulting state space of the model and the granularity of the analysis. Table I shows examples of this, where we generated two different models: one that abstracts the positions of all agents present in the model (row *Multi*); and one that outputs separate models for each of the agents (rows SA_*), with *min* (*max*) denoting the minimum (maximum) number of transitions in the model generated for a towing vehicle. Putting the individual models in parallel yields a complete system. For each model, we analyzed the example property $\mathbf{P = \quad [F < 3 \quad q\_0\_1 > 0]}$, i.e. *What is the probability of reaching the grid cell represented by state* q_0_1 *within 3 discrete time steps?*.

Analyzing properties on the multi-agent model take longer than for the single agent models. Also note that for agent #19, different grid sizes yield the same probability. Consider the following plan:

```
AGENT 0 Path: 11 11 19 27 28 29 21 13
AGENT 1 Path: 19 19 27 28 29 30 22
AGENT 2 Path: 27 28 29 37 45 37 29
AGENT 3 Path: 35 27 28 29 37 38
```

Where Agents $0 \ldots 4$ represent towing vehicles and $11, 19, 27 \ldots$ represent grid positions. For example the first line in the plan states that in the first time step,

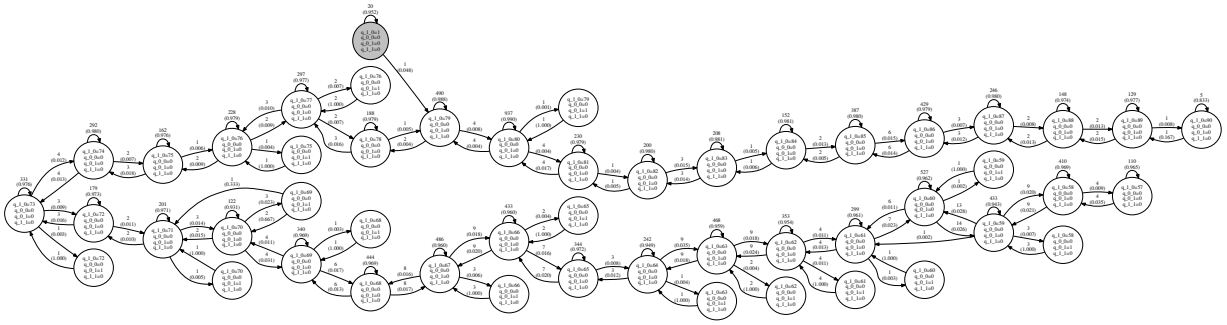

Fig. 6. Discrete Time Markov Chain inferred from example log data. States are represented by the number of vehicles in each cell of a 2x2 grid projected on the airport surface.

	2x2					**4x4**					**8x8**					**10x4**					**4x10**					**10x10**				
	ID	$T_\#$	$S_\#$	P	T	*ID*	$T_\#$	$S_\#$	P	T	*ID*	$T_\#$	$S_\#$	P	T	*ID*	$T_\#$	$S_\#$	P	T	*ID*	$T_\#$	$S_\#$	P	T	*ID*	$T_\#$	$S_\#$	P	T
Multi	-	257	115	0.0367	4	-	546	288	1.00	15	-	949	549	1.00	16	-	925	524	1.00	11	-	808	462	0.00	6	-	1143	688	0.00	16
SA_{min}	#19	5	3	0.0138	3	#19	13	7	0.0138	3	#19	23	12	0.0138	2	#14	26	13	0	2	#19	19	10	0.0138	3	#19	33	17	0.00	1
SA_{max}	#10	11	4	0.00825	2	#29	31	13	0.00	3	#24	85	38	0.00	3	#24	60	25	0.00	3	#24	59	25	0.00	3	#24	103	45	0.00	3

TABLE I

RESULTS FOR DIFFERENT GRID CONFIGURATIONS FOR MULTI-AGENT (ROW MULTI) AND SINGLE AGENT (ROWS SA_*) MODELS. *ID* IS THE AGENT ID; $T_\#$ ($S_\#$) IS THE NUMBER OF TRANSITIONS (STATES) IN THE GENERATED MODEL; P IS THE OBTAINED PROBABILITY FOR THE PROPERTY $\mathbf{P} =? \quad [\mathbf{F} < \mathbf{3} \quad \mathbf{q\_0\_1} > \mathbf{0}]$; AND T IS THE ANALYSIS TIME FOR THE PROPERTY.

agent 0 is in grid position 11. It will stay in the same position in the 2nd time step, while in the third time step it will move into position 19.

The analysis proceeds as follows. Suppose we have build DTMC models for each agent using past telemetry data (using the procedure described in the previous section). We can then analyze the plan with respect to to the built models: the plan defines implicit constraints on the agent. For example Agents 0 and 1 should not be in the same square (id 19) after 3 time steps. We can encode this constraint as the following PCTL properties:

- $\mathbf{P} =? \quad [\mathbf{F} < \mathbf{3} \quad \mathbf{q19} > \mathbf{0}]$: *What is the probability that Agent 0 is present in quadrant 19 within the first 3 time units (seconds)? We check the first property against Agent 0's model*
- $\mathbf{P} =? \quad [\mathbf{F} < \mathbf{3} \quad \mathbf{q19} > \mathbf{0}]$: *What is the probability that Agent 1 is present in quadrant 19 within the first 3 time units (seconds)? We check this second (identical) property against Agent 1's model. etc*

Note: we expect a lot of symmetries in terms of towing vehicle-centric models and properties. We plan to address this in the future.

We have described our preliminary efforts in developing a model-inference component for which predictive analysis is envisioned to support the decision procedure of the towing vehicle dispatcher. For future work, we plan to refine our models. In particular we want to distinguish between arrivals and departures and create a more detailed model with a finer grid that takes into account the configuration of the airport. For example, a finer grid may take into account the gate area, taxiways, and runways and allow a decomposition of this area into polygons. To bootstrap the model-inference, we would also like to use random testing. We also plan to integrate the model in the dispatcher to minimize delays in taxiing, to avoid congestions, and to maximize throughput to increase capacity of the airport. Another area that we plan to investigate is the behavior of towing vehicles around intersections.

V. SUMMARY

Airport surface operations is a complex logistical problem involving the coordination of humans and machines to achieve the maximum use of existing capacity. Automation for increasing capacity and reducing human workload is slowly being integrated into tower operations. This paper proposed a integrated approach for automating all phases of surface operations, from route planning to execution monitoring, including the potential for autonomous towing of aircraft. One of the interesting issues in developing a complete system for surface operations is the handling of uncertainty. One solution, close to that adopted in current operations, is based on continuous scheduling and replanning of

surface trajectories to address uncertainty during plan execution. Our general approach is to 'migrate' part of the uncertainty into probabilistic or flexible predictive models, which then will enable decision making, either human or automated, under more informed conditions. This approach will guide future research on this problem.

REFERENCES

[1] R. Morris, M. L. Chang, R. Archer, E. V. C. II, S. Thompson, J. L. Franke, R. C. Garrett, W. Malik, K. McGuire, and G. Hemann, "Self-driving towing vehicles: A preliminary report," in *Proceedings of the Workshop on AI for Transportation (WAIT)*, 2014, pp. 35–42.

[2] S. Rathinam, Z. Wood, B. Sridhar, and Y. Jung, "A generalized dynamic programming approach for a departure scheduling problem," in *AIAA Guidance, Navigation, and Control Conference*, 2009, pp. 10–13.

[3] W. Malik, G. Gupta, and Y. Jung, "Spot release planner: Efficient solution for detailed airport surface traffic optimization," in *Proceedings of the 12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference.*, 2012.

[4] V. Cheng, V. Sharma, and D. Foyle, "A study of aircraft taxi performance for enhancing surface operations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 2, no. 2, 2001.

[5] P. C. Roling and H. G. Visser, "Optimal airport surface traffic planning using mixed-integer linear programming," *Int. J. Aero. Eng.*, vol. 2008, no. 1, pp. 1:1–1:11, Jan. 2008. [Online]. Available: http://dx.doi.org/10.1155/2008/732828

[6] L. Hiatt and R. Simmons, "Pre-positioning assets to increase execution efficiency," *IEEE International Conference on Robotics and Automation*, pp. 324 – 329, 2007.

[7] J. Reif, "Complexity of the mover's problem and generalizations," in *Proceedings of the 20th IEEE Symposium on the Foundations of Computer Science*, 1979, pp. 421–427.

[8] J. Y. Du, J. O. Brunner, and R. Kolisch, "Planning towing processes at airports more efficiently," *Transportation Research Part E*, vol. 70, pp. 293 – 304, 2014.

[9] C. Lesire, "An iterative a* algorithm for planning of airport ground movements," in *ECAI 2010*, 2010, pp. 413 – 418.

[10] S. Ravizza, J. Atkin, and E. Burke, "A more realistic approach for airport ground movement optimisation with stand holding," *Journal of Scheduling*, vol. 17, no. 5, pp. 507–520, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10951-013-0323-3

[11] J. A. D. Atkin, E. K. Burke, and S. Ravizza, "The airport ground movement problem: Past and current research and future directions," *4th International Conference on Research in AIr Transportation*, pp. 131 – 138, 2010.

[12] J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *AAAI Conference on Artificial Intelligence*, 2013, pp. 1444–1449.

[13] P. Surynek, "Towards optimal cooperative path planning in hard setups through satisfiability solving," in *Pacific Rim International Conference on Artificial Intelligence*, 2012, pp. 564–576.

[14] J. Yu and S. M. LaValle, "Planning optimal paths for multiple robots on graphs," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 3612–3617.

[15] E. Erdem, D. G. Kisa, U. Oztok, and P. Schueller, "A general formal framework for pathfinding problems with multiple agents," in *AAAI Conference on Artificial Intelligence*, 2013.

[16] T. S. Standley, "Finding optimal solutions to cooperative pathfinding problems." in *AAAI Conference on Artificial Intelligence*, 2010.

[17] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 3260–3267.

[18] M. Goldenberg, A. Felner, R. Stern, G. Sharon, N. R. Sturtevant, R. C. Holte, and J. Schaeffer, "Enhanced partial expansion A*," *Journal of Artificial Intelligence Research*, vol. 50, pp. 141–187, 2014.

[19] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, "The increasing cost tree search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 195, pp. 470–495, 2013.

[20] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[21] R. Luna and K. E. Bekris, "Push and swap: Fast cooperative pathfinding with completeness guarantees," in *International Joint Conference on Artificial Intelligence*, 2011, pp. 294–300.

[22] B. de Wilde, A. W. ter Mors, and C. Witteveen, "Push and rotate: Cooperative multi-agent path planning," in *International Conference on Autonomous Agents and Multi-agent Systems*, 2013, pp. 87–94.

[23] L. Cohen, T. Uras, and S. Koenig, "Feasibility study: Using highways for bounded-suboptimal multi-agent path finding," in *Annual Symposium on Combinatorial Search*, 2015, pp. 2–8.

[24] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, ser. LNCS, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 585–591.

[25] K. G. Larsen, P. Pettersson, and W. Yi, "Uppaal in a nutshell," *Int. Journal on Software Tools for Technology Transfer*, vol. 1, pp. 134–152, 1997.