Using Unstructured Data to Improve the Continuous Planning of Critical Processes Involving Humans

Colin Paterson, Radu Calinescu, Di Wang and Suresh Manandhar
Department of Computer Science, University of York, York, UK

***Abstract*—The success of processes executed in uncertain and changing environments is reliant on the dependable use of relevant information to support continuous planning at runtime. At the core of this planning is a model which, if incorrect, can lead to failures and, in critical processes such as evacuation and disaster relief operations, to harm to humans. Obtaining reliable and timely estimations of model parameters is often difficult, and considerable research effort has been expended to derive methods for updating models at run-time. Typically, these methods use data sources such as system logs, run-time events and sensor readings, which are well structured. However, in many critical processes, the most relevant data are produced by human participants to, and observers of, the process and its environment (e.g., through social media) and is *unstructured*. For such scenarios we propose COPE, a work-in-progress method for the continuous planning of critical processes involving humans and carried out in uncertain, changing environments. COPE uses a combination of runtime natural-language processing (to update a stochastic model of the target process based on unstructured data) and stochastic model synthesis (to generate Pareto-optimal plans for the process). Preliminary experiments indicate that COPE can support continuous planning effectively for a simulated evacuation operation after a natural disaster.**

***Index Terms*—natural-language processing; stochastic model synthesis; probabilistic model checking**

I. Introduction

The success of human-centric critical processes such as search and rescue, disaster relief operations, and emergency management relies on the dependable use of relevant information to support effective decision making. A very challenging decision-making task in this context is the continuous planning needed because of the uncertainty and frequent changes in the environment and goals of operational processes (e.g., [1]–[5]).

When models are used to derive operational plans, it is paramount to ensure that these models are both reliable and up to date. Accordingly, the models are updated at run-time, with their parameters (and sometimes their structure) derived from observations of data sources which record model features. The techniques used for this purpose typically require *structured data sources* (e.g., application logs or software-generated events). As such, these techniques cannot be applied to critical processes with human participants who generate relevant *unstructured data* through channels like social media, web-based forums, and verbal reporting of their observations.

This paper introduces a work-in-progress approach for the continuous planning of operational processes involving humans (COPE). The main contribution of COPE is its novel integration of: (i) natural language processing, to update the stochastic model of a human-centric critical process by exploiting information encoded in unstructured data streams such as Twitter; and (ii) stochastic model synthesis, to dynamically generate updated Pareto-optimal plans for the process.

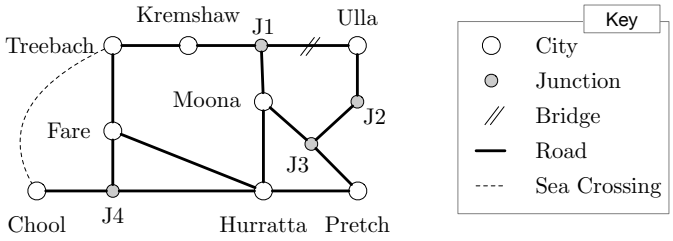

Fig. 1. Topological map showing the transportation infrastructure of Neopolis

II. Motivating Example

We consider an operation in which a disaster relief team must devise and communicate evacuation plans to people traversing a country to safety after an earthquake that led to shortages of food and medicine, infrastructure damage, areas of unrest, etc. The cities and the routes between them present risks to the success of the operation. In addition, traversing a city or completing a route has associated costs (e.g., associated to fuel consumption and journey time).

To illustrate the problem, we introduce the fictitious country Neopolis with the topological map from Fig. 1. Neopolis represents a dynamic environment in which the risks of travel can change very quickly and in which infrastructure is unreliable. We consider a scenario in which the news of a potential threat to a team located in Ulla has been received and a decision is made to evacuate them. The operation will be judged a success if they reach the evacuation point in Chool before a ship, which is waiting to transport them (and other similar evacuees) out of the country, leaves. Thus, we wish to construct an evacuation plan as a sequence of locations for the team to traverse from their current location to the evacuation point. The plan will be continually evaluated and updated using two criteria:

1) the probability of reaching the evacuation point;
2) the expected time to complete the journey.

Accordingly, we need to synthesise plans that: (i) satisfy constraints specified by the domain experts; and (ii) achieve Pareto-optimal trade-offs between the two criteria, i.e., they cannot be modified to satisfy one optimisation criterion without worsening their satisfaction of the other criterion.

To synthesise such plans, we want to construct a model of the environment based on the most up to date knowledge

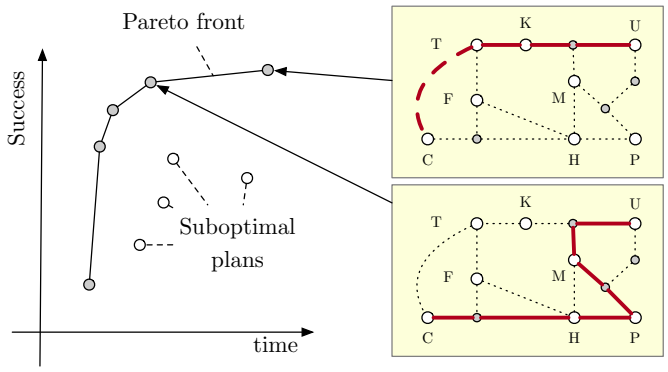

Fig. 2. Pareto front for the Neopolis evacuation scenarios (left) and two Pareto-optimal plans (right)

available. Some of the model parameters will be fixed, e.g. the distance between locations, whilst others will need to be estimated by domain experts (e.g., the risk of capture when traversing a city or road, and the likelihood of having to return to a previous stage of the journey due to a road block). Note that when knowledge is encoded in unstructured sources, it cannot be utilised by existing route planning services like Google Maps or Apple Maps. These services rely heavily on historic road traffic data (which is invalid during emergencies) and on real-time location data from devices running these services (which include no information about many of the risks and costs which impact evacuation route planning).

Fig. 2 shows a set of Pareto-optimal plans generated by COPE (as described in the next sections) for the Neopolis scenario. The upper solution, via Treebach, has the highest probability of success; however, this plan takes a long time to reach the evacuation centre due to the sea crossing. The second plan has a slightly lower chance of success but is much faster. Given the highly dynamic nature of the environment we cannot guarantee that the assumptions made in order to synthesise these plans will remain valid for the duration of the evacuation. Therefore, these plans must be updated in response to changes in the sensed environment. We assume that Twitter is used extensively in this region, and hence we wish to use this social media platform to inform model updates and planning decisions.

III. COPE Continuous Planning

A. Approach overview

Fig. 3 depicts the high-level workflow of our COPE approach to continuous planning of operational processes. The six steps of this workflow are described below.

Step 1 - Domain experts initiate the continuous planning by specifying constraints and optimisation criteria for the required plans based on the objectives of the operational process. For instance, for the planning of evacuation routes, the constraints may define a maximum route completion time, and the (conflicting) optimisation criteria may include minimising risk and costs (e.g. fuel consumption if refuelling is difficult or impossible). COPE allows for these constraints and optimisation criteria to be specified in probabilistic variants of stochastic logic [6] extended with costs and rewards [7].

Step 2 - Given these constraints and optimisation criteria, and considering the characteristics of the operation and their prior domain knowledge, the experts then develop an initial stochastic model of the operational process. For example, for the evacuation operation, the model captures the relevant locations and preliminary estimates for the time, success probability and cost of travelling between these locations. The COPE stochastic modelling process uses Markovian models annotated with costs. Such models have an established track record of supporting planning in uncertain scenarios [8]–[10].

Step 3 - Dealing with the uncertainties and frequent changes of operational processes requires the continuous exploitation of both dedicated data streams provided by trusted personnel and *openly available* information provided by the general public. COPE deals with the real-world scenario where data is encoded in messages available in natural language (i.e. as plain English text in the form of Twitter messages).

Step 4 - COPE utilises Natural language processing (NLP) techniques built on our recent research [11]–[13] to translate the plain-English items of information into a stream of machine-readable events. The NLP engine is configured to identifiy those events which are relevant for the calculation of risk, costs, durations and any other features required to update the COPE stochastic model.

Step 5 - The NLP-generated stream of machine-readable events drives the continuous stochastic model learning which underpins COPE. Up-to-date Markov models of the operational process are built and updated continuously, starting from the initial stochastic model developed by the domain experts. This key COPE step leverages recent research on adaptive model learning [14], [15] and refinement [16], [17].

Step 6 - This COPE step employs plan synthesis techniques to dynamically generate sets of feasible Pareto-optimal which optimise the objective in the presence of system constraints. These plans then act as an advisory input to the disaster relief team and are communicated to the end users via standard channels of communication.

B. Architecture and generic implementation

For a preliminary evaluation of COPE, we developed a generic implementation of its workflow from Fig. 3. We then specialised this implementation, whose architecture is shown in Fig. 4, to develop a simulator for the Neopolis evacuation operation introduced in Section II.

At regular intervals, the generic COPE implementation probes OSINT sources (for our simulator this was Twitter) and retrieves data in response to a search request. This data is then passed to the *COPE controller*, which sends the data through a (Python-implemented) gateway to the *NLP classifier*.

The NLP classifier is pre-trained to identify event types relevant to the application domain. For the Neopolis scenario, we considered three event types: gunshot, earthquake (because of the risk of aftershocks), and flood (which may be caused, for instance, by damages to dams and locks). If an event is identified then this is returned to the controller with a confidence level (a value between 0 and 1, where 1 denotes

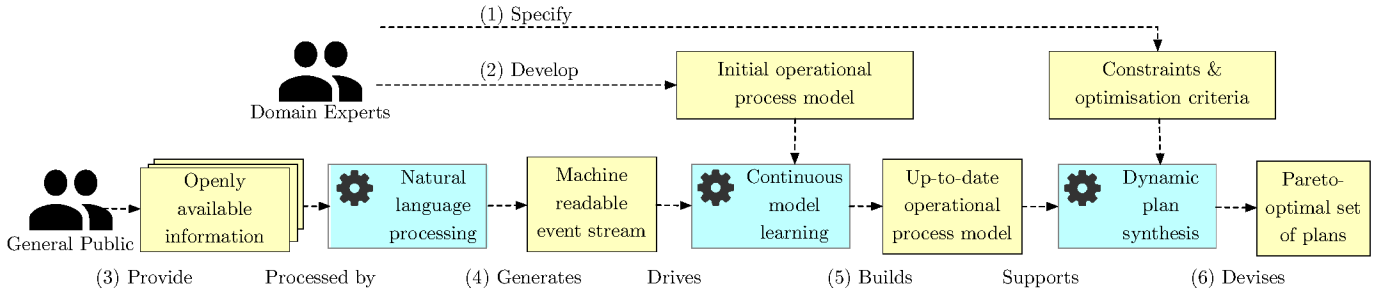

Fig. 3. COPE continuous planning workflow

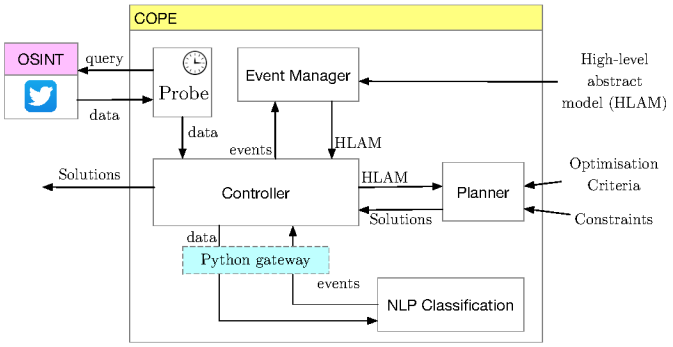

Fig. 4. COPE: Software Architecture

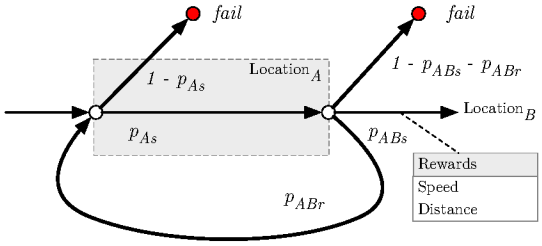

Fig. 5. Probabilistic modelling of locations and routes, showing the state transitions for one of the actions possible at location A (i.e., travel to B)

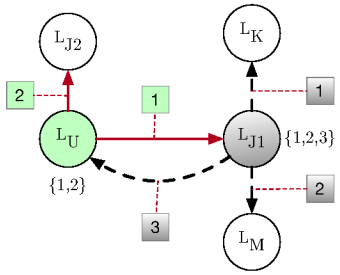

Fig. 6. Possible actions at location Ulla (L_U)

certainty that the event did occur) and metadata such as a location name for the evacuation scenario.

The COPE controller then passes these events to an *Event Manager* module. If the events identified are new, or significant updates (e.g., in confidence level) for already known events, then the parameters of the current model are updated to reflect the latest state of the environment. Finally, the updated model is passed to a *Planner* module, which uses a heuristic to find the Pareto-optimal solutions within specified constraints; the current version of COPE uses multi-objective genetic algorithms for this purpose. These solutions are then returned to the controller, and displayed to the user for selection of the most appropriate plan.

IV. COPE Planning for Evacuation Operations

A. Modelling locations and routes

For the motivating example we defined a probabilistic model for an agent traversing the map as shown in Figure 5. Having arrived in a location A, the team successfully navigates the location with probability p_{As} and with probability $1-p_{As}$ the team is captured and the operation fails. If they successfully traverse the location, then with probability p_{ABs} they will reach location B, but with probability p_{ABr} they will have to return to location A, e.g. because the road is blocked. Finally, with probability $1-p_{ABs}-p_{ABr}$ the operation will fail. To aid with the construction of the stochastic models, the COPE simulator allows a high-level abstract model (HLAM) specification to be provided in XML format.

At each location in the model a set of actions is available to an agent. For example, an agent at Hurratta may take one of four routes leading to Pretch, Fare, Moona or J4 (cf. Fig. 1). Enumerating these actions for each location defines the state/action set of a Markov decision process (MDP) that models the human-centric critical process. Fig. 6 shows all possible actions associated with Ulla and J_1. An agent starting at Ulla (L_U) may undertake one of two possible actions, i.e., attempt to travel to J_1 (L_{J1}) or J_2 (L_{J2}). If the first option is taken, and the agent successfully transitions to this new location, then three new actions are possible—travel to Kremshaw (L_K), to Moona (L_M), or return to Ulla.

B. Identifying events

COPE uses a probe to interface with an OSINT API; in the case of the Neopolis simulator, this is Twitter. At regular intervals the probe submits a query to the API and the batch of tweets is returned as JSON-encoded objects which represent the social media messages generated since the last probe. The batch of messages is then sent to the NLP classification module, which is written in Python to leverage the existing machine learning libraries Keras [18], Tensorflow [19] and NLTK [20]. Accordingly, COPE uses a Python Gateway implemented with the Pyro4 (Python remote objects) framework [21] to send messages between the COPE controller and the NLP module.

For the Neopolis evacuation scenario, we trained three NLP classifiers based on data obtained from real incidents. The

TABLE I
TWITTER DATA SETS USED TO TRAIN THE NLP CLASSIFIERS AND TO TEST THE COPE EVACUATION OPERATION SIMULATOR

Event type	Event Location & Date	#Tweets†	Training	Testing
Gunshot	Las Vegas Oct 2017	16,000	Yes	-
Gunshot	Lower Manhattan Oct 2017	50,000	-	Yes
Earthquake	Oklahoma Oct 2017	17,000	Yes	-
Earthquake	California Oct 2017	14,000	-	Yes
Flood	Louisiana Aug 2016	18,000	Yes	-
Flood	Colorado Sep 2013	31,000	-	Yes

†Different numbers of tweets were posted at locations and times close to where and when each of the events occurred (including many tweets unrelated to the event).

classifiers are implemented using neural networks with an embedding layer [22], three one-dimensional convolutional and max pooling layers, and a binary soft max layer. For training we utilised binary cross entropy as the loss function. Table I shows the data which was gathered for use in this work (for both training and testing). Each dataset was gathered using a set of event specific keywords as well as a GPS location and search radius around the event of interests . We selected 1000 tweets from each of the training sets by hand and labelled them as being related to the event or being normal tweets. This subset was then used as a training set for the classifier. Datasets marked 'Testing' in Table I were only used for the evaluation of the COPE simulator of the Neopolis evacuation operation, as described in Section V.

When a single tweet is passed to each of the NLP classifiers a value is returned which indicates the probability that the tweet refers to an event as identified during training. For a batch of tweets we record the number of tweets for which the probability reported by the NLP classifier is greater than 0.5 and calculate a confidence value that an event has been identified in the batch of tweets processed:

$$confidence = \frac{e^{r(n-t)}}{1 + e^{r(n-t)}} \quad (1)$$

where n is the percentage of tweets in the batch for which the probability of the event having occurred is at least 0.5; t is a threshold above which we assume an event to have occurred; and r is a scaling factor allowing for the gradient of the *confidence* function to be controlled.

To illustrate the role of the confidence function, Fig. 7 depicts the result of using the function with the gunshot classifier applied to the *testing dataset*, i.e. tweets posted in the Lower Manhattan area of New York within a time interval starting several hours before and ending several hours after the terrorist attack on 31st October 2017. The tweets were processed by the NLP classifier in batches of 100 tweets at a time. The confidence level that a gunshot (fired by the police at the end of the incident) was identified rises sharply after the gunshot is first mentioned in the stream of data (indicated by a dotted vertical line in Fig. 7). Peaks in the graph before this event are due to portions of the stream where other gun crime events, which occurred across the USA in the weeks leading up to this event, are mentioned.

In addition to identifying events in the OSINT data stream, COPE requires the location at which the event took place in order to update its stochastic models appropriately. COPE uses the Stanford Named Entity Recognizer [23] for this purpose. Since a single route or location may be referred to using multiple names COPE supports aliases and these are specified in the HLAM e.g. Ulla Bridge refers to the route between Ulla and J1. Thus, each tweet is processed as shown in Figure 8, with the list of events associated with a batch of tweets returned to the COPE controller module.

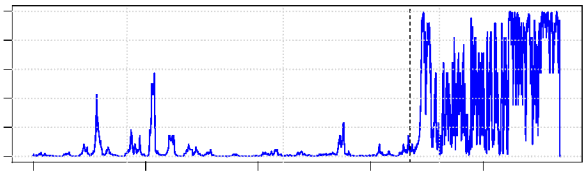

Fig. 7. Confidence level that a gunshot was identified in the testing dataset

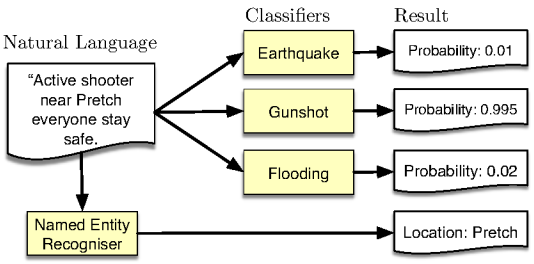

Fig. 8. COPE NLP processing of a single tweet

C. Updating models at run-time

The list of events identified by the NLP classification module is received by the COPE controller and passed to the Event Manager module. This module is responsible for maintaining a list of known, active, events and, when a new event is detected, updating the HLAM.

The confidence function returns a real valued number and a small change in the value should not typically trigger a re-planning. We therefore map the confidence to a severity band and re-planning occurs when the event moves between bands. For the Neopolis scenario, the set of possible event severities is $\{none, low, medium, high\}$. The probability range associated with each band is domain specific for the Neopolis scenario, these ranges are $[0, 0.2]$ for $none$, $(0.2, 0.5]$ for low, $(0.5, 0.8]$ for $medium$ and $(0.8, 1.0]$ for $high$.

A set of mapping rules are defined to specify how parameters in the model should be modified from the initial value specified by the domain experts. For the motivating example we can update: a reward associated with a location; a reward associated with a route; and a probability associated with a transition. Two examples of update rules, specified by domain experts in natural language, may therefore be formed as:

- If there is a Gunshot event identified at a location then increase the probability of operational failure in that location by 10%, 20% and 30% for low, $medium$ and $high$ severity events, respectively.
- If there is a Flooding event with severity $medium$ or $high$ detected on a route then decrease the speed associated

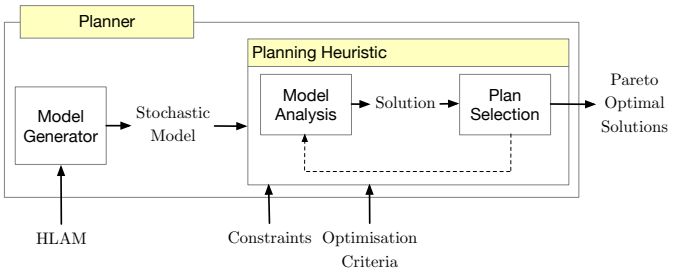

Fig. 9. Workflow for the COPE planner

with that route by 50% and increase the probability to return to the starting location by 30%.

These rules are then encoded within the Event Manager, which updates the parameters of the HLAM when needed, and then returns control to the COPE controller with a notification that the model has been modified.

D. Planning

The aim of the planning processes is to identify the optimal action to take in each state of the model with reference to the specified optimization criteria whilst respecting any specified constraints. Fig. 9 shows the planning workflow enacted by COPE. The first step is for a model generator to convert the HLAM to a stochastic model. The generator implements the probabilistic modelling assumptions defined by the domain experts, Fig. 5, and produces a model which encodes the action sets, Fig. 6. For our running example the model generated is a discrete-time Markov chain (DTMC) in the PRISM high-level modelling language [24], with the actions encoded as integer-valued parameters in the model. This encoding is equivalent to a Markov decision process, but we prefer it in the current version of COPE because of the way in which our synthesis of new plans with multiple constraints and multiple optimisation criteria works.

Each of the optimisation criteria are then encoded as model properties in temporal logics. For the DTMC models used in our example these are encoded as probabilistic computation tree logic (PCTL) [6].

The planning heuristic next creates a concrete plan by setting the values associated with the actions. The stochastic model and optimisation criteria are then passed to a model analysis engine, for our example this is PRISM.

The action set and resulting property values form a solution and this is then evaluated against the specified constraints. If a constraint is violated then the solution is discarded otherwise the solution is compare to the best current solutions. The planning heuristic maintains a list of Pareto optimal solutions and, when the stopping criteria is reached, these solutions are returned to the user.

For small models it is possible to run an exhaustive search of the solution space, however, as the number of locations and actions increases this becomes unfeasible and therefore we look to more efficient search heuristics. For our example the search heuristic was implemented using the EvoChecker approach which we previously developed [3], [25], [26].

TABLE II
EVACUATION SCENARIO

Time	Event (data source)
Day 1, 08:10	Initial evacuation plan generated.
Day 1, 09:45	Aftershock affecting Ulla bridge.
	(1000 tweets from California earthquake dataset)
Day 1, 10:00	Team depart Ulla.
Day 1, 23:00	Gunshot in Pretch.
	(2000 tweets from the Lower Manhattan gunshot dataset)
Day 2, 05:00	Flood on Hurratta Highway.
	(3000 tweets from the Colorado flood dataset)

V. INITIAL EXPERIMENTS AND RESULTS

In order to evaluate COPE we constructed a HLAM for the motivating example given in Section II. We then synthesized an evacuation scenario as a Twitter stream with events extracted from each of the test sets given in Table I. These events were shifted in time to fit the synthesised scenario, and the location names were mapped to locations in Neopolis; otherwise, the data sets were left unaltered. The complete scenario is shown in Table II.

The data which precedes events in Table I was combined to create a set of "normal" tweets. Tweets from this set was then sampled randomly and added to the event data to construct a complete synthesised stream for the scenario.

COPE was then executed using the synthesized data set, and the resulting routes and decisions are shown in Fig. 10. Every time when an event is detected, COPE generates a set of Pareto-optimal plans which are displayed to the user with the associated probability of success and estimated time to arrival (E.T.A.). The user then selects one of the planned routes (indicated by * in the diagram) and the route is then followed until the next event occurs. For each plan a solid line indicates the section of the route still to be completed whilst the dashed line indicates that portion of the route already undertaken. As the team travels across the country COPE successfully detects each event and calculates the Pareto-optimal routes from the team's present location to the evacuation point at Chool.

The motivating example used in our preliminary experiments generates models with 12 decision parameters, one for each location in the map. The total search space for this problem is then 46,656 (including combinations that correspond to invalid solutions to the problem). Searching this problem space for Pareto-optimal solutions was undertaken using NSGA II [27], an efficient multi-objective genetic algorithm which has been widely used in a range of problem domains, and which is implemented into COPE.

To evaluate the impact of model size on the performance of COPE, we carried out additional experiments to increase the search space. The model generation module was modified to allow for multiple routes across each location in Neopolis. This involved the addition of many different ways to travel across cities, and replacing junctions with multiple different routes through rural regions. The risks and costs associated with each of these routes was randomized, and Table III shows the search space and time for COPE to generate plans for these much larger models. All of the experiments were run on a 2.9 GHz MacBook Pro i5 with 16 GB of memory.

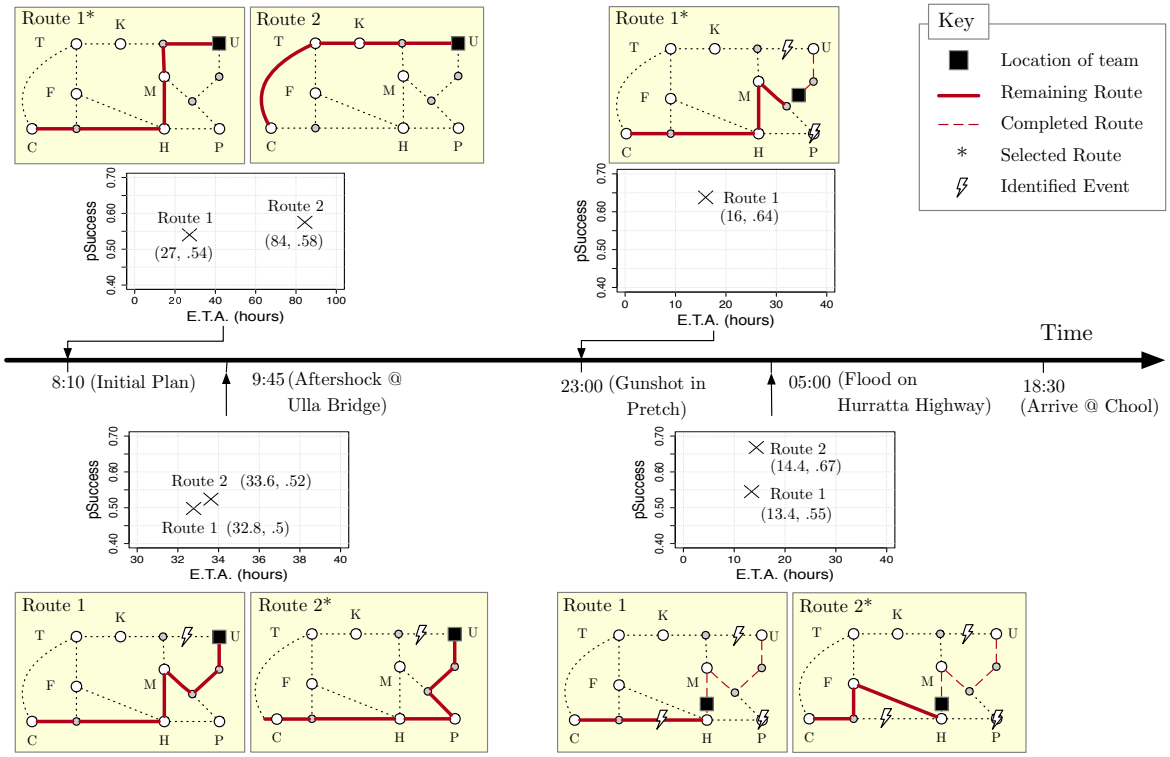

Fig. 10. Evacuation scenario with COPE generated plans

TABLE III
COPE SCALABILITY EXPERIMENTS

#Routes through cities & junctions	Search space	Time (s)
1	$46,656$	45
5	244×10^6	49
20	4×10^{15}	81

The NSGA II algorithm used a population size of 100 and evaluated 1000 model instances.

Whilst we are aware that more experimentation needs to be undertaken to assess the scalability of our approach, these experiments suggest that COPE is able to handle stochastic models which include a large parameter search space.

VI. Related Work

COPE is an example of a human-in-the-loop system [28] where humans are not only consumers of information but also provide input and operate as system-level effectors. The use of stochastic models enables the accurate analysis of systems in uncertain environments. However, maintaining up-to-date stochastic models at run-time is challenging, and several approaches have been proposed to learn model parameters using maximum likelihood estimators [29], Bayesian estimators [14], [30]–[32] and Kalman filters [33]. These approaches require structured data sources and, unlike COPE, cannot leverage information encoded in unstructured data.

Social media has become increasingly useful in providing timely information and valuable insights during crisis situations [34], [35] and constructing event detection algorithms which analyse social media at run-time is now an active research area [36], [37]. More widely, the extraction of information from unstructured data has seen application in a range of domains from healthcare [38] to politics and finance decision support systems [39]. The primary focus of this work is information extraction and, unlike COPE, not the updating of stochastic models at run-time.

To the best of our knowledge, COPE is the first approach that integrates event detection from unstructured data and the run-time updating of stochastic models.

VII. Conclusions and Further Work

In this paper we introduced COPE, a work-in-progress approach for updating stochastic models of human-centric processes through the exploitation of information encoded in unstructured data streams.

We implemented a specialisation of the COPE architecture and demonstrated how natural language processing and stochastic synthesis could be integrated to provide Pareto-optimal solutions for a planning problem in the presence of data extracted from real-world events.

In future work we intend to integrate information extraction from additional unstructured sources, including longer narrative forms and audio data. Data collected will also inform the deployment of resources to acquire additional data, e.g. the deployment of drones to acquire aerial imagery. In addition, we will formalise the derivation of model update rules in response to observed data and domain expertise. We will also assess the scalability of COPE, specialise the generic architecture for additional application domains, and explore options for the assurance of COPE applications by extending the methodology from [40].

REFERENCES

[1] C. Ayora, V. Torres, M. Reichert, B. Weber, and V. Pelechano, "Towards run-time flexibility for process families: open issues and research challenges," in *International Conference on Business Process Management*. Springer, 2012, pp. 477–488.

[2] R. Calinescu, "Emerging techniques for the engineering of self-adaptive high-integrity software." *Assurances for Self-Adaptive Systems*, vol. 7740, pp. 297–310, 2013.

[3] S. Gerasimou, G. Tamburrelli, and R. Calinescu, "Search-based synthesis of probabilistic models for quality-of-service software engineering (t)," in *30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 319–330.

[4] S. Luna and M. J. Pennock, "Social media applications and emergency management: a literature review and research agenda," *International Journal of Disaster Risk Reduction*, vol. 28, pp. 565–577, 2018.

[5] R. de Lemos, D. Garlan, C. Ghezzi, H. Giese, J. Andersson, M. Litoiu, B. Schmerl, D. Weyns, L. Baresi, N. Bencomo, Y. Brun, J. Camara, R. Calinescu, M. B. Cohen, A. Gorla, V. Grassi, L. Grunske, P. Inverardi, J.-M. Jezequel, S. Malek, R. Mirandola, M. Mori, H. A. Müller, R. Rouvoy, C. M. F. Rubira, E. Rutten, M. Shaw, G. Tamburrelli, G. Tamura, N. M. Villegas, T. Vogel, and F. Zambonelli, "Software engineering for self-adaptive systems: Research challenges in the provision of assurances," in *Software Engineering for Self-Adaptive Systems III. Assurances*, R. de Lemos, D. Garlan, C. Ghezzi, and H. Giese, Eds. Springer, 2017, pp. 3–30.

[6] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal Aspects of Computing*, vol. 6, no. 5, pp. 512–535, 1994.

[7] S. Andova, H. Hermanns, and J.-P. Katoen, "Discrete-time rewards model-checked," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2003, pp. 88–104.

[8] R. Calinescu, M. Autili, J. Cámara, A. Di Marco, S. Gerasimou, P. Inverardi, A. Perucci, N. Jansen, J.-P. Katoen, M. Kwiatkowska *et al.*, "Synthesis and verification of self-aware computing systems," in *Self-Aware Computing Systems*. Springer, 2017, pp. 337–373.

[9] G. Mason, R. Calinescu, D. Kudenko, and A. Banks, "Assurance in reinforcement learning using quantitative verification," in *Advances in Hybridization of Intelligent Methods*. Springer, 2018, pp. 71–96.

[10] R. Calinescu and M. Kwiatkowska, "CADS*: Computer-aided development of self-* systems," in *International Conference on Fundamental Approaches to Software Engineering*. Springer, 2009, pp. 421–424.

[11] A. Komninos and S. Manandhar, "Dependency based embeddings for sentence classification tasks." in *HLT-NAACL*, 2016, pp. 1490–1500.

[12] ——, "Structured generative models of continuous features for word sense induction." in *COLING*, 2016, pp. 3577–3587.

[13] ——, "Feature-rich networks for knowledge base completion," in *ACL*, 2017.

[14] R. Calinescu, Y. Rafiq, K. Johnson, and M. E. Bakır, "Adaptive model learning for continual verification of non-functional properties," in *5th ACM/SPEC International Conference on Performance Engineering*. ACM, 2014, pp. 87–98.

[15] A. Filieri, L. Grunske, and A. Leva, "Lightweight adaptive filtering for efficient learning and updating of probabilistic models," in *37th International Conference on Software Engineering*. IEEE Press, 2015, pp. 200–211.

[16] C. Paterson and R. Calinescu, "Accurate analysis of quality properties of software with observation-based Markov chain refinement," in *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2017, pp. 121–130.

[17] C. A. Paterson and R. Calinescu, "Observation-enhanced QoS analysis of component-based systems," *IEEE Transactions on Software Engineering*, vol. PP, p. 1, 2018. [Online]. Available: https://doi.org/10.1109/TSE.2018.2864159

[18] F. Chollet, "keras," https://github.com/fchollet/keras, 2015.

[19] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[20] E. Loper and S. Bird, "NLTK: The natural language toolkit," in *ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, 2002, pp. 63–70.

[21] I. de Jong, "Pyro4," https://github.com/irmen/Pyro4, 2018.

[22] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[23] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by Gibbs sampling," in *43rd Annual Meeting of the Association for Computational Linguistics*, 2005, pp. 363–370.

[24] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *23rd International Conference on Computer Aided Verification*. Springer, 2011, pp. 585–591.

[25] S. Gerasimou, R. Calinescu, and G. Tamburrelli, "Synthesis of probabilistic models for quality-of-service software engineering," *Automated Software Engineering*, vol. 25, no. 4, pp. 785–831, 2018.

[26] R. Calinescu, M. Češka, S. Gerasimou, M. Kwiatkowska, and N. Paoletti, "Efficient synthesis of robust models for stochastic systems," *Journal of Systems and Software*, vol. 143, pp. 140–158, 2018.

[27] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[28] J. Cámara, G. A. Moreno, and D. Garlan, "Reasoning about human participation in self-adaptive systems," in *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE Press, 2015, pp. 146–156.

[29] T. W. Anderson and L. A. Goodman, "Statistical inference about markov chains," *The Annals of Mathematical Statistics*, pp. 89–110, 1957.

[30] R. Calinescu, K. Johnson, and Y. Rafiq, "Using observation ageing to improve Markovian model learning in QoS engineering," in *2nd ACM/SPEC International Conference on Performance engineering*. ACM, 2011, pp. 505–510.

[31] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli, "Model evolution by run-time parameter adaptation," in *31st International Conference on Software Engineering*. IEEE Computer Society, 2009, pp. 111–121.

[32] I. Epifani, C. Ghezzi, and G. Tamburrelli, "Change-point detection for black-box services," in *18th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2010, pp. 227–236.

[33] T. Zheng, C. M. Woodside, and M. Litoiu, "Performance model estimation and tracking using optimal filters," *IEEE Transactions on Software Engineering*, vol. 34, no. 3, pp. 391–406, 2008.

[34] S. Vieweg, C. Castillo, and M. Imran, "Integrating social media communications into the rapid assessment of sudden onset disasters," in *International Conference on Social Informatics*. Springer, 2014, pp. 444–461.

[35] M. Imran, C. Castillo, F. Diaz, and S. Vieweg, "Processing social media messages in mass emergency: A survey," *ACM Computing Surveys (CSUR)*, vol. 47, no. 4, p. 67, 2015.

[36] F. Atefeh and W. Khreich, "A survey of techniques for event detection in twitter," *Computational Intelligence*, vol. 31, no. 1, pp. 132–164, 2015.

[37] M. Hasan, M. A. Orgun, and R. Schwitter, "A survey on real-time event detection from the twitter data stream," *Journal of Information Science*, p. 0165551517698564, 2017.

[38] P. Yadav, M. Steinbach, V. Kumar, and G. Simon, "Mining electronic health records (EHRs): a survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, p. 85, 2018.

[39] F. Hogenboom, F. Frasincar, U. Kaymak, F. De Jong, and E. Caron, "A survey of event extraction methods from text for decision support systems," *Decision Support Systems*, vol. 85, pp. 12–22, 2016.

[40] R. Calinescu, D. Weyns, S. Gerasimou, M. U. Iftikhar, I. Habli, and T. Kelly, "Engineering trustworthy self-adaptive software with dynamic assurance cases," *IEEE Transactions on Software Engineering*, vol. 44, no. 11, pp. 1039–1069, 2018.