Journal of Computer Science

Original Research Paper

A Hybrid Approach to Process Planning: The Urban Traffic Controller Example

Jimoh Falilat Olaitan, Simon Parkinson and Thomas Leo McCluskey

Centre for High-Performance Intelligent Computing, Department of Informatics, University of Huddersfield, Queensgate, Huddersfield, HD1 3DH, UK

Article history
Received: 02-01-2017
Revised: 10-07-2017
Accepted: 15-08-2017

Corresponding Author:
Falilat Olaitan Jimoha
Centre for High-Performance Intelligent Computing,
Department of Informatics,
University of Huddersfield,
Queensgate, Huddersfield,
HD1 3DH, UK
Email: F.Jimoh@hud.ac.uk

Abstract: Automated planning and scheduling are increasingly utilised in solving evsery day planning task. Planning in domains with continuous numeric changes present certain limitations and tremendous challenges to advanced planning algorithms. There are many pertinent examples to the engineering community; however, a case study is provided through the urban traffic controller domain. This paper introduce a novel hybrid approach to state-space planning systems involving a continuous process which can be utilised in several applications. We explore Model Predictive Control (MPC) and explain how it can be introduce into planning with domains containing mixed discrete and continuous state variables. This preserves the numerous benefits of AI Planning approach by the use of explicit reasoning and declarative modelling. It also leverages on the capability of MPC to manage numeric computation and control of continuous processes. The hybrid approach was tested on an urban traffic control network to ascertain it practicability on a continuous domain; the results show its potential to control and optimise heavy volumes of traffic.

Keywords: Automated Planning, Model Predictive Control, Urban Traffic Control

Introduction

Process planning is the act of selecting and assigning resources towards achieving a desired goal. Process planning is performed programmatically and It involves the design of autonomous computer program; such computer programs are self-aware of their environment, can adapt to change, generate and scrutinise goals (Russell *et al*., 1995). There have been many successful implementations of autonomous planning for processing planning. There has been the successful implementation of automated planning and scheduling for many engineering processes. For example, early work by Khoshnevis and Chen (1991) utilised automated planning and scheduling in manufacturing processes for comprehensive resource selection and allocation.

This early success motivated the use of autonomous planning and scheduling for many different applications; however, as each solution often contained tightly coupled domain knowledge alongside the algorithms, researchers were often spending large amounts of time developing systems which shared similar core algorithms. This resulted in the establishment of domain-independent automated planning where state-of-the-art algorithms are designed in isolation from the domain knowledge. These algorithms are then used alongside an action model representing the domain specific knowledge. Also, the emerging development in the field of automated planning with constraints processing has facilitated the deployment of deliberative reasoning to real-time control applications (Heinrich *et al*., 2015; Chen *et al*., 2015). There are many successful applications of domain-independent planning to real-world problems. Example could be found in the computer integrated manufacturing process (da Silva Fonseca *et al*., 2016); relocation problem (Tierney *et al*., 2012), calibration of machine tools (Parkinson and Longstaff, 2015), clinical validation (Dinapoli *et al*., 2016) and crowd sourcing (Machado *et al*., 2016).

It is vital to enable deliberative reasoning in systems. Introducing deliberation into a controller enables it to reason with its components, environment and functionality. It enable the generation of effective plans towards achieving desirable goal within the control system (Jimoh and McCluskey, 2012). This facilitates its effectiveness to deal with unexpected situations that might not have been learnt,

adopted nor programmed beforehand into such a system Dusparic *et al*. (2016). Embedding automated planning into urban traffic control systems will introduce deliberative reasoning in urban traffic controllers. Deliberative reasoning in a controller would introduce intelligence into the UTC systems through the generation of plans and schedules for self management. This will ultimately contribute to the reduction of traffic congestion and carbon emissions on roads.

However, as the variety of possible planning applications increases so is the complexity of the domain knowledge (Jimoh and McCluskey, 2016). The complexity is significantly hindering the uptake of novel automated planning applications due to limitations of planning applications to handle continuous change in numeric values. To avoid this limitation, the complexity of the domain knowledge are currently being relaxed through the discretisation of continuous transformation into a discretised profile of linear change (Lhr *et al*., 2012). For example, the application to machine tool calibration, non-linear change in environmental temperate is discretised to reduce complexity (Parkinson *et al*., 2014). However, this discretisation is often at a cost to the quality and accuracy of the generated plan and a compromise has to be established. This also motivates the requirement for a novel approach to handling continuous processes in planning for control systems. The next section explains a hybrid algorithm that uses automated planning with an embedded MPC strategy to create an algorithm that can reason with planning problems containing numerics with continuous change. The specific example provided is in the urban traffic control environment to generate plans for a controller to optimise the traffic situation.

In this study, a hybrid planning system is presented through the introduction of Model Predictive Control (MPC) approach into a classical state-based planning system. It facilitates efficient planning in the presence of complex numeric and logical changes within a problem domain. The technique's primary application is in autonomous traffic management and will be provided as an example throughout the paper. However, the traffic management domain has many of the similar characteristics with complex engineering and manufacturing planning problems.

The layout of the paper is as follows: The first section presents a survey of work related to this paper. This leads to the description of the developed hybrid approach. Following this, a case study is presented where the technique is applied to the urban traffic controller.

Background

The increase in demand for innovative plan generation techniques, plan execution, monitoring and recovery; has stemmed awareness towards evolving system designs which make use of advance planning and implementation frameworks (Jimoh and McCluskey, 2015; Laguna *et al*., 2014). Teleo-Reactive Executive (T-Rex) is an example of such design. T-Rex is a goal-oriented autonomous underwater vehicles that integrates automated planning technology for real-time plan generation and execution. T-Rex framework is designed to improve research in the field of oceanic science (Pinto *et al*., 2012). Another example of planning design is Planning and Execution L-Earning Architecture (PELEA). PELEA introduce adaptable modular design that integrates learning with planning and execution. It also incorporates sensing and monitoring for realtime re-planning (Jimnez *et al*., 2013). We propose the use of Model Predictive Control (MPC) design in continuous planning to create reasoning in controllers that can solve problems in domains which are modelled using variables whose values are changing continuously. Similarly, Domain Predictive Control (DPC) is another design that is proposed for continuous (re-) planning in hybrid systems (Lhr *et al*., 2012). It involves the extraction of a discretised domain model from given MPC dynamic equation of a system to control realtime applications. This is different from the work in this study; which involves the creation of symbolic continuous domain model of a system while leveraging on MPC derived from a model of dynamical equations of the same system as a heuristic to control the search space in symbolic planning.

Control systems which support Urban Traffic Control (UTC), such as those controlling networks of traffic lights, have utilised AI techniques since the 1970's (Jimoh and McCluskey, 2014). These systems are embedded in a real-time control environment and are often based on algorithms that rely on feedback and adaptation. They make use of road traffic data which may be gathered every few seconds or gathered over several years. Resulting in traffic control systems operating with the fundamental of adaptive signal control in road networks established from stored traffic data. However, these approaches to traffic control has some limitations during unprecedented situations such as road accidents or an unexpected change in traffic demand within short interval of time (De Oliveira and Bazzan, 2009; Jimoh *et al*., 2013b). In such circumstances, traffic control systems usually use fixed traffic signal timing or apply some hardcoded approach to revert into a recognised state. Therefore, there is a need for intelligent controls that can effectively generate plans and execution towards restoring an unpredicted traffic situation to desired condition. One promising direction is by creating a hybrid control design that will support intelligent systems to spontaneously reason and deliberate with their declarative knowledge, towards managing themselves during unexpected situations (Jimoh *et al*., 2013a). Such intelligent controls would be

an achievement in the urban traffic control domain and this paper is a step towards realising such goal.

Model Predictive Control

Control engineering is a field of knowledge within the engineering discipline, which applies control theory to design and implement systems with desired behaviours. Predictive Controls is a sub-set of Control Engineering utilised in adopting and anticipating the future pattern of control processes in other to control its inputs for a desirable future goal.

MPC attracts remarkable consideration in the control of dynamic systems which makes it an essential aspect of control practice (Osusky and Vesely, 2015). MPC was established within the industrial sector as an alternative option of control compared with traditional Proportional Integrate Derivative (PID) controls (Bennett, 1993). MPC formulation incorporates optimal control, multi-variable control, stochastic control, deadtime processes and future references where applicable (Camacho and Bordons, 1999).

MPC has several algorithms; they differ in the way they represent the model of the process as well as the cost function to be minimised. MPC algorithms have been continually enhanced to increase its robustness and scalability for instantaneous processes (Al-Gherwi *et al*., 2011; Falugi *et al*., 2010; Tay, 2007; Osusky and Vesely, 2015).

MPC has been implemented in a variety of applications ranging from production planning (Mezghiche *et al*., 2015; Baldea *et al*., 2015); industrial production (Zhu *et al*., 2015; Alanqar *et al*., 2017; Grosso *et al*., 2016) and supply chain (Chu and You, 2015; Schildbach and Morari, 2016); intelligent Transport Systems (Mahalingam and Agrawal, 2016; Roncoli *et al*., 2016); agriculture (Graf Plessen and Bemporad, 2017) and robot manipulation in path planning (Ji *et al*., 2017; Joos *et al*., 2017).

The MPC Approach

The mathematical model of a controlled process, as well as the assumed disturbances that might occur during its operation, is built based on the past experience of operation and past data from similar operations within the same system. A cost function is derived from the available resources and constraints that need to be optimised for the entire duration of the process. The system uses the pre-defined model as a guide to maximise the cost function when given a set of varying input parameters, output parameters and the dynamic changes in the state of the environment. The system plans over a period of time, which is known as the horizon. The generated plan is applied to the system to control the process by changing its current state to a desirable state for a given period of time. The new state is sampled again. It re-plans for another horizon taking the present state from the feedback loop as well as all the system constraints into consideration. This approach of planning is called "receding the horizon". This planning and re-planning approach make MPC robust and able to keep a control process in a desirable state over a given period. It also allows it to function in a partially observable environment, because of its ability to sample dynamic environment at every sampling time during a re-plan.

The Store-and-Forward Model

In 1963, Gazis and Potts introduced the store-and-forward traffic flow model with the aim of achieving a sensible compromise between computational efforts and precision control in dynamic systems. A store-and-forward traffic flow model is utilised in this study to formulate a state space predictive control model; it helps to creates a dynamic mathematical formulation of the network model (Guo *et al*., 2014). Figure 2 depicts a diagrammatic representation of the application of MPC into a UTC structure. The simplified store-and-forward traffic model only allows for split optimisation. Cycle time and offsets must be calculated by other control algorithms.

Roads networks is represented as sets of junctions $j \in J$ and links $z \in Z$ and as shown in Fig. 1. Each signalised junction j has sets of outgoing links O_J and incoming links I_j. A sample of urban road is shown in Fig. 1. It has two junctions M and N adjacent to each other, such that $z \in I_N$ and $z \in O_M$. The remaining fundamental variables are:

- i represents the stage identifier
- $x_z(t)$ is the state variable indicating the number of vehicles in link z at step t
- j represents the junction identifier
- $g_{j,i}$ the control input indicating the green time of stage i at junction j
- t discrete time index, $t = 0,1,2...$
- S_z represents the saturation flow of link Z
- v_z represents the set of stages where link z has right of way
- $t_{w,z}$ turning rate; towards link Z from the links w that enter junction M
- T the control interval in discrete time step
- C_j junction j cycle time

Given that the cycle times C_j for all junctions $j \in J$ are the same and fixed such that $C_j = C$. Equation 1 can consequently denote the dynamics of link z:

$$x_z(t+1) = x_z(t) + T\begin{bmatrix} \left(1-\tau_z\right)\sum\limits_{w\in I_M}\dfrac{S_w\sum\limits_{i\in v_w} g_{N,j}(t)}{C} \\ -\dfrac{S_z\sum\limits_{i\in v_z} g_{N,j}(t)}{C} \end{bmatrix} \quad (1)$$

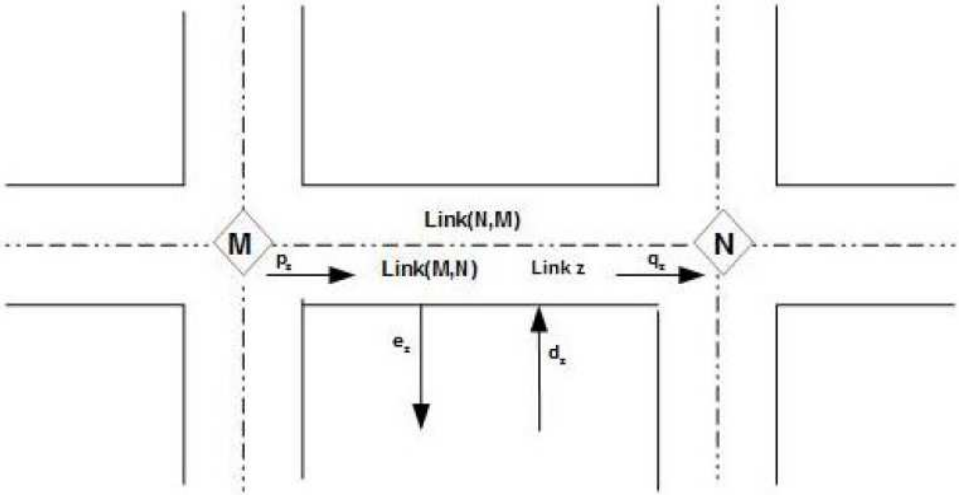

Fig. 1. Links and Junctions Illustration of an Urban Road

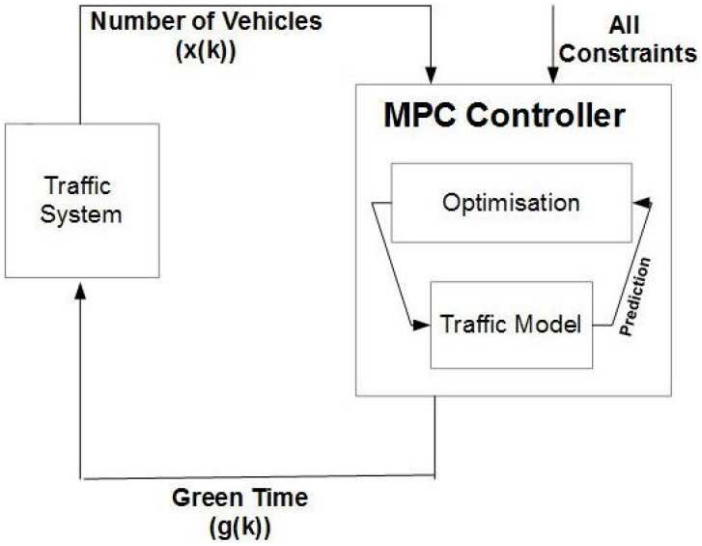

Fig. 2. Applying MPC to UTC Structure

Each $z \in Z$ has an outflow capacity at specific green times; this is represented by the Saturation flow S_z. S_z could be fixed using a standard value or calculated by another approach; we assumed it is known and constant. Turning rates $t_{w,z}$ of $z \in O_j$ and also $w \in I_j$, could be calculated in real-time or estimated utilising statistical values. Assuming $T = C$; a further simplification of the variables (replacing both second and third term) from Equation 1 will yield Equation 2:

$$x_z(t+1) = x_z(t) + T\left[p_z(t) - q_z(t) + d_z(t) - e_z(t)\right] \quad (2)$$

Such that, $p_z(t)$ represents inflow to link z, $q_z(t)$ represents outflow from link z. Also, $d_z(t)$ represents demand in the link z and $e_z(t)$ represents exit flow in the link z; in the sample time $[tT, (t+1)T]$. The exit flow $e_z(t)$ can be estimated by $s_z(t) = t_z p_z(t)$ while assuming that the exit rates t_z are known. The resulting outflow is given in Equation 3:

$$q_z(t) = \frac{S_z \sum_{i \in v_z} g_{N,i}(t)}{C} \quad (3)$$

In a bit to reduce computational efforts, red-green switching in a cycle is not taken into consideration in the model. However, the modelled flow represents the average real flow for each period.

A linear scalar equation that represents a specified link is shown in Equation 1. Organising all interconnected conservation equations in a state space

form (for individual link), equation 4 would represents a state space model that defines an entire traffic network:

$$x(t+1) = x(t) + Bg(t) + d(t) \quad (4)$$

Such that $x(t)$ represents numbers of vehicles in each link (state vector); $g(t)$ represents all green time settings (control vector) and $d(t)$ represents any disturbance within the network. B is the network characteristics, it is represented by a constant coefficient matrix of proper dimensions. For instance the network topology is represented by B.

MPC Constraints on UTC

Given a UTC traffic model, there are some constraints that have to be considered. The constraints are formulated from the store-and-forward model discussed in the previous section

Non-Negative Control Constraints

At any given time t there cannot be a negative volume of traffic flowing through link z. Also, the green split timing at any given junction falls between the traffic light cycle at that junction:

$$g_{j,i} \geq g_{j,\min}, \forall i \in J \quad (5)$$

Traffic Light Cycle Constraints

All green time constraint holds for every stage i at junction j:

$$\sum_{i=I}^{N_j} g_{j,i}(t) \leq C - L_j, \forall j \in J \quad (6)$$

Such that L_j represents the set lost time and N_j represents the value of stages, at the junction j.

Green Duration Constraints

Equation 7 represents the lower and upper bounds constraints on the green time at a junction:

$$g_{j,\min} \leq g_{j,i} \leq g_{j,\max}, \forall j \in J \quad (7)$$

Such that $g_{j,\max}$ represents maximum permissible time and $g_{j,\min}$ represents minimum permissible time at junction j.

Flow Conflict Constraints

This is to avoid collision between links at a junction. Given a connected link only one link could be active at a time.

Non Negative Queue Constraints

The queues on a given link are restricted to length of the link connecting two junctions:

$$0 \leq x_z \leq x_{z,\max}, \forall z \in Z \quad (8)$$

such that $x_{z,\max}$ value specifies the maximum number of vehicles that can be admitted into link z. This restriction helps to eliminate over saturation of a link in rush hours. It also makes sure that the value of a queue length on the road is nonnegative during the computation of control input.

Capacity Constraints

The capacity of a link must not be exceeded. Thus, the number of vehicles leaving any link will be limited by the state and capacity of the downstream link.

The Objective Function

The objective of this MPC formulation is to reduce the number of vehicles waiting in line at a junction. This is evaluated as the total time it requires to exit the vehicles waiting at individually connected junctions within a network of connected links. Thus, to reduce queuing distance on links, Equation 9 represents a quadratic costs function that satisfies Equation 4, 6 and 7; with the aim of minimising queues and optimising green times at a junction:

$$J = \sum_{t=1}^{N_p} \left(\| x(t) \|_Q^2 + \| g(t) \|_R^2 \right) \quad (9)$$

Automated Planning

The ability to reason with the dynamics of life and its environment by creating and implementing plans to solve challenges is one of the uniqueness of human race. Embedding this quality of man into artificial entities such as machines, is the foundation of Automated Planning. AI planning is a field that involve the formation of sequence or partially ordered plans whose execution solves a given problem; from an initial state or situation to a state that satisfies it specified goals conditions (Gupta *et al*., 1998; Fox and Long, 2003; Garrido *et al*., 2001). To embed deliberative property in control system, it is essential for the controller to be situationally aware of its components, its operating environment and the correlation between its component and environment (Jimoh *et al*., 2013b). This is accomplished through the extraction of the operational knowledge of a given domain, in this case, a road traffic domain. The extracted knowledge is declaratively represented in a language that can be understood by the planning system. The domain knowledge employed in the implementation of this work is represented in a language that is close to PDDL+ (Fox and Long, 2006). This structural language provides a formal declarative representation of the problem and domain entities along with all the operational policies of the domain.

Modelling UTC Domain

Given a domain of interest with facts and description of the environment and problems within that domain, a UTC model could be defined as a symbolic system which has inference and rules that represent the domain of interest. Traffic flow models are of three distinguished types: Macroscopic model; microscopic model and mesoscopic model. Refer to the work of Hoogendoorn and Bovy (2001) for a detailed overview of existing traffic models. A macroscopic model is considered in this analysis through the use of aggregated variables to describe traffic flows.

The syntax and semantics of the domain description language used in this implementation are similar to PDDL+. Detailed explanation of PDDL+ syntaxes and semantics is in the work of (Fox and Long, 2006); this includes the semantics for the construction and implementation of state representation and progression.

A domain model has been encoded from a case study town centre area in the United Kingdom as shown in Fig. 3. The domain model is made up of static and dynamic part (Jimoh *et al*., 2013a). The static part represents road network topology, such as road name, road capacity, road length and junctions linking the roads. A directed graph is used to represent the road network layout, edges represent roads and vertices represent either source road, sink road or junction. Vehicles enter the network through the source road and exit the network through the sink road. The dynamic aspect of the model is represented by the flow rate of vehicles on each road and the queuing distance such road.

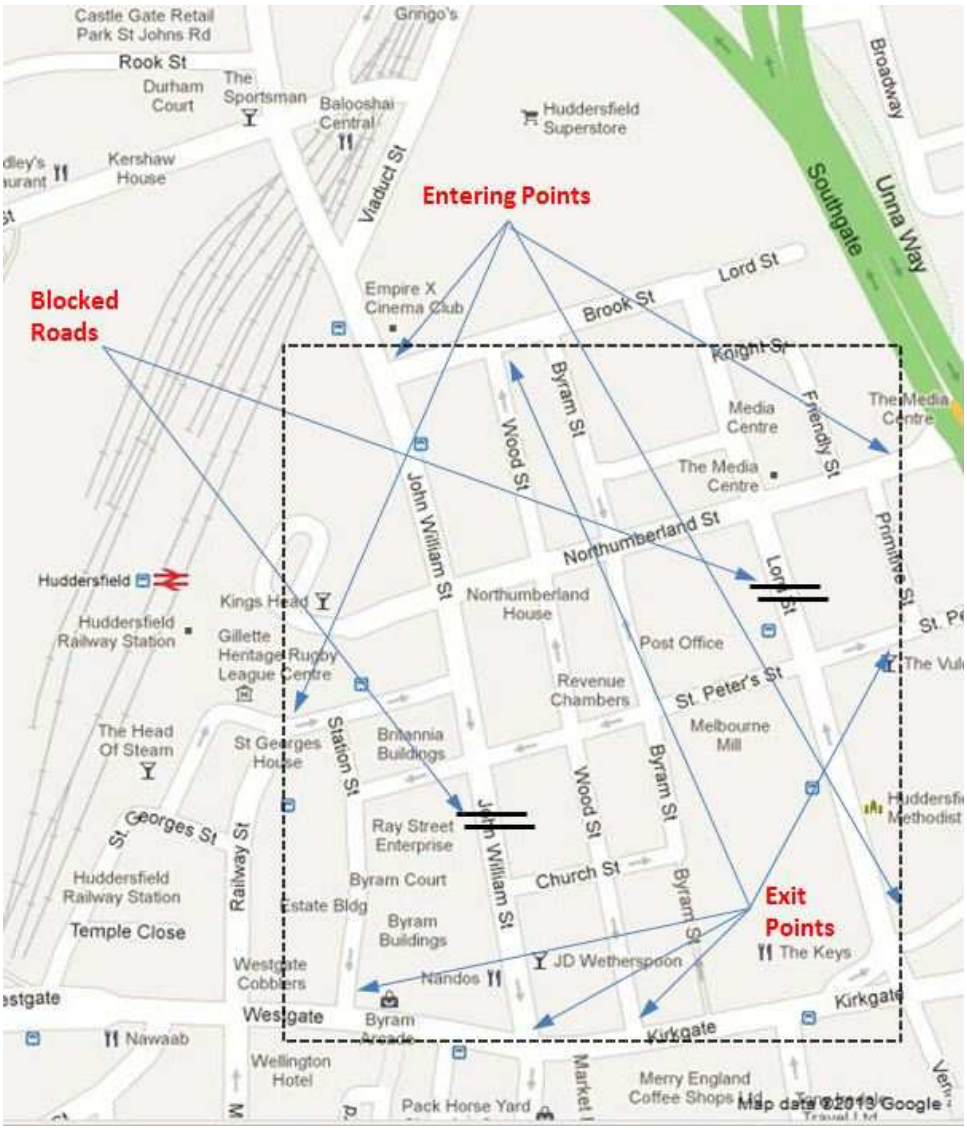

Fig. 3. Excerpt of map showing the network blocked roads, source road and sink roads within a town center area of Huddersfield, West Yorkshire, United Kingdom

```
(:action  switchGreen
:parameters [atJunction, thisPhase, fromRoadA, toRoadB]
:duration dur
:precondition [(intersect atJunction thisPhase fromRoadA toRoadB)]
               [(>(queuelenght (fromRoadA 0.0))
               (<(interuptlevel (toRoadB 7.0))]
:effect ([(JflowActive atJunction thisPhase fromRoadA toRoadB)])
       )
```

Fig. 4. Sample declaration of an action operator

```
 (:process  RtrafficFlow
:parameters [atJunction, thisPhase, fromRoadA, toRoadB]
:duration dur
:precondition [(RflowActive atJunction
                thisPhase fromRoadA toRoadB)]
:effect ([(decrease(queuelenght (fromRoadA (* #10.0 1.0))),
(increase(queuelenght (toRoadB (* #10.0 1.0)))])
```

Fig. 5. An excerpt from a process declaration

```
 (:event  upStreamSaturated
:parameters [atJunction, fromRoadA, toRoadB]
:duration dur
:precondition [(>=(queuelenght (toRoadA roadBcapacity)))]
:effect ([(assign(queuelenght (toRoadA roadBcapacity))
            (assign(interuptlevel (toRoadB 7.0))])
        )
```

Fig. 6. Sample declaration of an event

The dynamic aspect of the model is continuously changing due to movement of vehicles in the road network. The UTC environment is modelled with predicates and fluents. The relationship between objects are represented with predicates. For example, given a predicate (link nLSouth wDStr) in a state *S*, it indicate that the road nLSouth is connected to wDStr in that state. Thus, traffic is allowed to flow from nLSouth to wDStr, provided all given constraints are satisfied. Fluents could be logical or numeric; it status are subject to changes within the model. Rich numeric expressions are possible with the use of numeric fluents. For example, a fluent = (queueLenght (nLNorth 300.0) indicates the current value of the queue in nLNorth to be equal to 300 m.

A UTC Planning Problem involves the effective navigation of vehicles within a network of roads with the purpose of optimising traffic flow. In our model, we consider action operators, grounded processes and events. Figure 4-6 shows a sample declaration of an action operator, grounded processes and events respectively.

The Hybrid Approach

Exploiting the relationship and building on the synthesis of MPC and AI planning techniques to solve problems involving both discrete and continuous state variables lies at the heart of this research work. The hybrid approach uses an A^* search algorithm technique for node exploration. The point within search space where search frontiers intersect or branch is referred to as the Node. State information and transitions are also stored in a node. The current node is expanded by comparing the preconditions of each operator with the proposition and numeric fluent; if it is satisfied given all other constraint are fulfilled; the effect of the operator is applied at the node. The declared numeric resource and constraints within the model are computed and updated at selected nodes during node exploration. Applicable operators are chosen and applied, in a receding horizon, to each state until the goal condition is satisfied or the expanded set of nodes becomes empty. Some essential definitions in the design and implementation of the hybrid algorithm are explained in the next section.

Preliminary Definitions

Definition 1 (State)

Assuming a Close World Assumption (CWA) on *S*, a state *S* gives a description, at any given snapshot of time,

the true situation of some world. Given that N is an assignment for the numeric variable to values and P are the set of atomic propositions. S is a pair $\langle P,N\rangle$.

Definition 2 (Initial State)

Given that N is an assignment of values to numeric variables and P is the set of atomic propositions that are true at the start of a planning problem. Initial State $I = \langle P,N\rangle$.

Definition 3 (Goal Condition)

Given that N is a set of numeric variables and P is a set of atomic propositions then, a Goal Condition $G = \langle P,N\rangle$. For a goal G to be satisfied in some state S values v satisfies some numeric constraints $vL < v < vU$ specified by G. Thus, S satisfies the goal condition if S satisfies every proposition in P and $\exists v = c \in N$: $VL < c < VU$ for all v in N. Here c is a constant representing a value between the lower and upper bound of v.

Definition 4 (Domain Model)

The Domain Model (DM), consist of:

- Set of Functions $\{n_1,...,n_k\} \in N$
- Set of Propositions $\{p_1,..., p_k\} \in P$
- Set of numeric Resources $\{r_1,..., r_k\} \in R$ and
- Set of Actions $\{a_1,...,a_k\} \in A$
- Set of Events $\{e_1,...,e_k\} \in E$
- Set of Processes $\{c_1,...,c_k\} \in C$

Definition 5 (Action)

An instantaneous action is characterised by sets of preconditions that must be true prior to the execution of the action and effects that becomes true after the execution of the action. The logical basis for actions is modelled using a collection of propositions, with vectors of numeric variables. Both P and v are manipulated and referred to by actions. The executability of an action is determined by its preconditions.

For example, the action switch to green has the precondition that the light is red with an effect that the light is green. A durative action A has three sets of preconditions: The condition that must hold at start $pre_{\Leftarrow}A$, at the end $pre_{\Rightarrow}A$ and throughout the execution of the action $pre_{\Leftrightarrow}A$. Effect could be durative or instantaneous, instantaneous effects are bound to the start $eff^{+}_{\Leftarrow}$ and $eff^{-}_{\Leftarrow}$ or end of the action $eff^{+}_{\Rightarrow}$ and $eff^{-}_{\Rightarrow}$ where positive and negative denote the propositions added and deleted at the start and end of A respectively. Also numeric effect $eff^{n}_{\Leftarrow}$ and $eff^{n}_{\Rightarrow}$ are updated at the start and end respectively. An example of action declaration is shown in Fig. 4.

Definition 6 (Processes)

A process p comprises of a precondition, C and a set of continuous effects, E, such that, if $S \models C$ then the continuous effects are active at state S.

For instance, the inflow process of vehicles V to a road R through a junction J. This process has a precondition that a given phase at junction J is active that is 'Green' and that the road use level of R less than the road-capacity-level; and the constraint that J is a connected inflow junction to road R. Once R is filled or blocked, an event is triggered that stops the process. The effect of Inflow process increases R traffic level at the flow rate of V as shown in Fig. 5. The derivative of traffic level in R is the summation the active inflow processes rates of the at any given time.

Definition 7 (Event)

The event e is activated in a state S such that $S \models C$, where C is an assertion expressing what triggers the event e. Given that E describes the effects of C on event e; then event e is defined as a state transition of (C, E). The application of effect E on state S produce a new state s' such that $s' \text{-}| E$. For example, an event 'upstreamFilled' to be triggered, it requires the estimated number of vehicles on such road to be equal or greater that the road capacity limit of such road as shown in Fig. 6.

Definition 8 (Operators)

Given a set of proposition $P_{(s)}$ and numeric fluents $N_{(s)}$, a numeric operator $\delta = \langle \text{pre}(\delta); eff(\delta)\rangle$ given that:

- The condition for applicability pre(δ) of an operator δ consist of:
  - A proposition or set of propositions $pre_{prop}\delta$ define over P
  - A numeric or set of numeric comparisons $pre_{num}\delta$ in the form of (exp {>, ≥,<,≤, =} exp′).
- The effect of an operator $eff(\delta)$ consists of:
  - An additional proposition $eff^{+}(\delta)$ produced and A deleted proposition $eff^{-}(\delta)$ removed after the operator execution.
  - Set of numeric operations $eff^{+}_{num}(\delta)$ in the form (n, op, exp)

In this definition, the arithmetic expression exp and exp′ involves variables from N. These are recursively defined among expressions in the form of arithmetical combination of {+, *, –,/}, numeric fluent and constants.

Definition 9 (Operators Applicability)

An operator δ is applicable in a state S iff, s is satisfied the operators propositional and numeric preconditions. That is:

- $pre_{prop}(\delta) \subseteq P(s)$ and
- $pre_{num}(\delta)$ must be valid(i.e., equal or in range of values) in all n where $n \in N(s)$

Definition 10 (Plan)

A plan comprises of action sequences and initiated processes; that could lead the initial state into a state satisfying the goal conditions, taken all the stipulated constraints into consideration. Given a continuous planning problem $Y = \{I, G, DM\}$ where, I is the initial state, G is a set of goal conditions and DM comprises of a set of operators. A solution for Ψ is a total ordered set of operators from δ, such that the ordered sequence of execution of these operators transforms I into a state where G is satisfied.

UTCPLAN: Top Level Algorithm

The planner input five components. These are: (a) The initial state (b) the goal condition (c) the domain model (d) the horizon prediction value and (e) the control horizon. The initial state "*S*" comprises of a set of propositions "*P*" and a sequence of the numerical variable "*R*". The Goal condition "*G*" is satisfied in a state *S*, if *S* satisfies every proposition in *P* and $\exists v = c \in N$: $V_L < c < V_U$ for all v in N. Assuming c is a constant representing a value between the upper and lower bound of v. A detail component of the domain model is defined in the preliminary definitions.

The fixed horizon prediction value N_p represents the period for which the MPC component will generate a new future prediction values to guide the search space. The control horizon value N_c represents the number of nodes frontiers that are searched at every control horizon window after an MPC prediction episodes. N_p and N_c are tailored to the domain and the nature of the problem that the planner is intended to solve.

A node is initialised in Lines 1-2. There are four components that constitute a node in the search space: (a) the set of propositions "*P*" component of "*S*" (b) the numerical variable components in the "*R*" component of "*S*" (c) the variable "$\Im$" that updates and saves the dynamic prediction values over successive horizons; "$\Im$" is initially set to null (d) a partial plan.

The search space is initialised within the outer loop of Line 4. Line 5 utilises the MPC numeric optimisation and prediction process to generate numeric control variables. The output of Line 5 could be inferred as a set of predicted actions whose execution fulfills the stipulated objective function and guides the search space towards satisfying the goal condition.

Algorithm 1 UTCPLAN: Top Level Algorithm

Input:
DM: Domain Model
N_p: prediction horizon
N_c: control horizon
(P,R): initial state
G: Goal Condition
Output: Plan.
1: S := [ ]; $\Im$: = null; $\wp$:= [R]
2: n := (P, $\wp$, S, $\Im$)
3: repeat
4: Q := {n}; x: = 1
5: $\Im$:= *UtiliseMPC*(n, N_c, N_p, $\Im$, *DM*)
6: while $x \leq N_c$ and $Q \neq \{\}$ and *noSolutionFound*(Q) do
7: n := *retrieveBest*(Q, $\Im$)
8: N := *Expand*(n)
9: Q := moveTo(N, Q)
10: $x := x + 1$
11: end while
12: if $Q \neq \{\}$ and *noSolutionFound*(Q) then
13: n: = *retrieveBest*(Q, $\Im$)
14: end if
15: until *SolutionFound*(Q)*or*Q = {}

The inner loop of Line 6 11 expands the search frontiers over a fixed horizon window N_c. The selection of the best node is informed by the output of UtiliseMPC procedure. The closest node to the given trajectory specified by the partial plan in the current $\Im$ is picked as the best node "*n*" and removed from "*Q*". The selected node "*n*" is expanded in Line 8 and returns a set of successor node "*N*". Line 9 adds "N" to the open set as detailed in Algorithm 5.3. There is currently no built-in specific heuristics for pruning the search space in UTCPLAN.

Given that the goal condition is not met upon the exit of the inner loop of Line 6-11; the best node is retrieve from Q informed by $\Im$. The best node "*n*" becomes the start node for a new search for the next control horizon window. The selection of a single node might create incompleteness in the algorithm, but it restricts the search and utilise the guidance of the MPC approach to select the best node for pruning the search space. The search and optimisation procedure is repeated from the current node in Line 15 until the goal conditions are satisfied, or the open node set becomes empty.

Nodes Expansion

The current node n is expanded by selecting the appropriate operator that satisfies the condition at the node. The effect of the operator changes the state at a node from '*n*' into a new state '*N*' as explained in Algorithm 5.3. The procedure for the application of an action, initiation of a process and the triggering of an event is explained in Algorithm ?? respectively. Certain assumptions are made with regards to the event semantics. For instance, there is no different in the

orders occurrence of simultaneous events. The detailed procedure for the application of an operator, grounded process and event is explained in Jimoh (2015).

Action Application

Definition 11 (Apply Action). Given an action a and a state s, if a is applicable in s, then a new state s' is produces and denoted by $s[a]$ as shown in Algorithm 3.

Algorithm 2 Expand(n) Algorithm

Input: n
Output: N
$N := \{\}$
$E := \{e'|e'$ represent instantiation of some event $e \in DM$ and n makes e':pre true$\}$;
$n :=$ apply all events in E chronologically to n
$O: = \{o'|o'$ represent instantiation of some operator $o \in DM$ and n make o':pre true$\}$
for all $o' \in O$ do
    $n': =$ apply o' to n
    $N := N[\{(n'.I, n'.\Im, [o']{+}{+}n'.S)\}$
end for
$P := \{p'|p'$ is an instantiation of some process $p \in DM$ and n make p':pre true$\}$
for all $p \in P$ do
    $n :=$ apply p for a unit of time to n
end for
$N := N \cup \{n\}$

A action consist of logical or numeric preconditions. The effect of an action operator could be logical propositions; numeric updates of the current state after the execution of the action or both. An example is given in Fig. 4. The action 'switchGreen' has a logical precondition that 'roadA' and 'roadB' must be connected by at the same junction. The two roads are also controlled by the same signal phase. The action in Fig. 4 also indicate numeric preconditions of an interrupt level seven for the linked roads. This means that the connect roads must not be a congested road. The action effect alters the signal phase at this junction, which consequently initiates a flow process at the connected junction.

Algorithm 3 Action Application
Input: s,a
Output: s'.
1: s' is initialised to be s;
2: All propositions in eff^+ a that are not already in s are added to $P(s)$
3: All proposition in eff^- a are deleted from $P(s)$
4: All numeric fluent f where $(f, op, \text{exp}) \in eff_{num}(\delta)$ are updated
5: All state $s \in S$ obtained by a non applicable operator is undefined and does not satisfy any condition.

Simulate Process

Definition 12 (Simulate Process)

Given a ground process c and a state s, such that c is applicable in s, the application of c in s, denoted by $s[c^+]$ to simulate continuous numeric changes in s for a period of time is as shown in Algorithm 4.

Whenever processes are initiated within a given node, it will run for a period of time at a single discretisation of a step count. For instance, time t becomes t =1, 2, 3...t_n given that t_n is the duration of the process simulation. Processes are initiated as an effect of an action or event trigger. The preconditions of process simulation are logical or numeric inequalities, but its effects produces a numeric update of the current state at the node. For instance, the effect of an action "switchGreen" in Fig. 4 could initiate a vehicles flow process at the flow rate of traffic on the connected roads as depicted by Fig. 5. Once a process is initiated at a node, it will continuously run for the specified duration of time, except if it is halted by an event. The current numeric status of the process is updated at the node upon the completion or halting of the process.

Algorithm 4 Simulate Process
Input: s, c
Output: s'.
1: initialise process duration time count = dur
2: repeat
3: All numeric fluent f such that $(f, op, \text{exp}) \in eff_{num}(c)$ is updated and modified according to the defined op and exp involved
4: Time #t and other primitive numeric variables are updated
5: until event e is triggered or dur exceeded.

Event Application

Definition 13 (Apply Event)

Given a ground event e and a state s, such that e is applicable in s, represented by $s[e]$, the application of e in s lead to a new state s' as shown in Algorithm 5.

Event application share some similarities with an action operator, except that, the unique difference is the fact that an action may occur if its preconditions hold, an event, on the other hand, must occur if its precondition hold. An event in the domain could be internally triggered from within a process, or outside the control of a process. Internally triggered event are interrupts that are activated while a process is running, it preconditions are usually numeric inequalities and their effect are also numeric assignments. These numeric assignments are set as preconditions for some actions in the domain. This means that the interrupts tell the planner to execute an

emphaction that could change the emphstate of the system or flag a display.

An example of event is to manage the constraint of traffic spill-over at junctions during rush hour as shown in Fig. 6. It has a precondition to check the capacity of the connected road during the process of traffic flow at a junction. The effect of this event stops the currently running process from transferring queue to the upstream road. This is achieved by an interrupt trigger that halts the process and pushes the current state of the node to the priority queue node.

Externally triggered event are a result of interaction between domain objects. An example of such external event is the activation of connectors that link two separate roads. Once the condition for the connector is satisfied, the queue from the previous road flows to the connected routes. This is outside the control of a junction, but the ripple effect of such event (traffic flow) affects the queues at downstream of the junctions. The different between this connecting event and an action is that once the event precondition is satisfied, it has to be activated, computed and updated to the current state, however, an action might only be selected if it necessary get the state closer to the goal state.

Algorithm 5 Apply Event
Input: s,e
Output: s'.
1: $s'$is initialised to be s;
2: All proposition in $eff^{-}e$ are removed from $P_{(s)}$
3: All propositions in $eff^{+}e$ that are not already in s are added to $P_{(s)}$
4: All numeric fluent f where (f, op, exp) $\in eff_{num}(\delta)$ is updated
5: Time #t and other primitive numeric variables are updated

The UtiliseMPC(n, DM, N_c, Np, $\Im$) Procedure

Numeric fluents R are stored in the node; the stored numeric are utilised in generating a dynamic prediction table (look ahead table) for a duration of control horizon N_p window within the UtiliseMPC procedure. A numeric optimisation procedure takes into consideration all constraints in the domain *DM* and the generated values from the prediction table to compute the best control values $\Im$ within the horizon window N_c, over a period of N_p. The computed value $\Im$ is the updated at the node n and use as a guide for the next set of alterations.

The numeric optimisation procedure is implemented as Satisfiability (SAT) problem solver in AI planning, formerly used in Shin and Davis (2005); Audemard *et al.* (2002). Such that, the continuous numeric variables with their associated constraints are converted to a linear programming problem within the search node. The best combination of input satisfying the stipulated numeric constraint is returned and updated at the node. Given a domain of problem for instance, assume N_c is set at 300 node count and N_p is set at 30 sec. At every 300 node counts, the planner retrieves past numeric fluents, sent it to the UtiliseMPC procedure and update the result at the node. This means that the past numeric fluents are utilised during the generation of a new set of predicted numeric values over a prediction horizon period of 30 sec. The predicted new generated set of values serve as an input to the numeric optimiser; to obtain the best option of numeric combination that would be used during the next successive search frontiers.

Implementation Assumptions

It is assumed that the continuous approximation of numeric counts(queue length) is maintained within the network. This is obtained at different level of abstractions based on the following: Route (R) explored by the planner during search space; queue (Q) denoting the numeric value of each road object at any instance of time; Source (Sc) which represents the entering road to the networks and sink (Si) which represents the exit roads. Vehicles originate from the source, passes through roads, connectors and junctions, then end up in sink.

A road could be active or inactive at every time instance. Vehicles are assumed to move on an active road at the flow rate of unit value per seconds of time *veh*/*sec*. We assumed the flow rate of the roads were known and fixed at the initial state. The flowrate of inactive road is assume to be zero; due to no movement of vehicles on such road.

Each of the junctions has two phase (1 and 2). Traffic can move from north to south or from east to west at junctions. Two conflicting roads cannot be activated at the same time at a junction. The domain model, incorporate declarative descriptions of grounded event that monitors the movement of traffic within linking roads. The planner selects the appropriate green phase duration to controls the traffic of roads connected at a junction.

All dynamic inputs, such as turning rates are assumed constant; with an exception of the state variables ($x_z(t)$) and controlled variables ($g_{j,i}$). The flow rate of individual junctions is also assumed to be constant. The rate of flow of vehicles is represented as a unit value per seconds of time (*veh*/*sec*). We assume we cannot control drivers behaviour; thus, we only control the green split (the controlled variable). We also assumed that the traffic flow dynamics are fully defined and included in the domain file.

We consider a linearised version of the quadratic problem that simplifies real-time calculations. Linearised methods often led to suboptimal solutions and could not consider the limits of some constraints exhaustively. Therefore, exploring more complex optimisation solution that can scale better in preferred for future purpose. The main objective of this implementation is not to scale the output metrics, but to investigate the

feasibility of using our UTCPLAN approach in this domain of interest (UTC).

Evaluation

The main evaluation criterion is to show that UTCPLAN can indeed accept inputs expressive domain descriptions within urban traffic domain and output solution plans containing continuous processes, events and actions through the integration of MPC with AI search - based planning techniques. This is measured by creating an expressive description of a UTC domain with traffic flow problems of various degree to test if UTCPLAN can generate execution plans that can control and manage traffic situation base on specified traffic goals.

The experimental traffic network (domain) is designed to have more than one connected junctions in other to test the centralise reasoning of UTCPLAN to manage upstream and downstream of traffic from connected road to the junction. This also allows us to test the feasibility of junction to junction traffic relationship within the network. Each junction in the model is designed to have more than one signal phase, for the purpose of evaluating the effectiveness of UTCPLAN at splitting the green times of the signal phases within a junction. There are several connected roads without a signaled junction within the network model; for the purpose of evaluating the effectiveness of UTCPLAN at reasoning with the dynamics of traffic flow in those linked roads not directly controlled by a signaled junction.

The effectiveness of the embedded MPC approach in UTCPLAN algorithm is tested with sample traffic domains; to evaluate the performance of UTCPLAN at controlling the signaled junctions while optimising the flow of traffic within the given network, during unexpected changes to the traffic situation. To achieve this, two signaled situation were created for experimental purpose:

Fixed

Signal duration are fixed for every junction within the network. The planner cannot alter the signal duration during search space. The planner reasons with the domain and problem information to generate solution plans using the fix signal value at every junction.

Controlled

Signal control is entirely at the discretion of the planner. The signal durations are set at initial state; however, the planner alters the signal duration whenever it anticipates a better control performance during search space; utilising the embedded MPC approach.

The speed of UTCPLAN was assessed with different volume of traffic with bottlenecks to investigation the plan generation time during light and heavy traffic situation. Numerous traffic flows were generated by altering the values of queuing distance on roads to create a heavier flow of traffic in the test suite. The quality of plan generated by UTCPLAN was evaluated for both controlled and fixed signal experiment. This is achieved by computing the total number of executable actions and initiated processes within the output plans, for both fixed and controlled signal.

Evaluation Criteria

To investigate the applicability and effectiveness of UTCPLAN, we use three evaluation criteria for comparison: Total time taken to generate a plan; the average number of processes initiated and the average number of actions sequence in the output plan. Makespan is not considered in this criteria because this implementation does not include a scheduler for makespan optimisation in the plan. Thus, using makespan as a major metric would not be suitable as criteria for evaluation of the planner.

A variation of UTCPLAN was created for the purpose of comparison and experimental analysis. This variation creates a planner version without integrating MPC approach. This version produces a Fixed Signal approach; it reasons with numerics within the domain similar to a classical numeric planner Hoffmann (2003). The Fixed Signal and the Controlled Signal are tested with the same formulation of domain and problems. Several traffic problems of increasing complexities were abstracted and modelled within the UTC domain. The modelled traffic problems are suitable for UTCPLAN evaluation because it highlights the advantages of the controlled signal (with MPC integration) over the fixed signal approach. The time discretisation of $t = 1.0$, is used in the two test cases (Fixed and Controlled); and the entire task in the UTC domain. The time taken to solve problems in our experiment is shown in Fig. 7. The performance of the planner (controlled signal) is compared with fixed signal value. The results of the fixed time duration compared with the controlled approach are reported in Table 1. Given that x_2 is the new average value and x_1 is the previous average value, the percentage change in value y% is measured by Equation 10 and recorded in Table 1:

$$y(\%) = \frac{x_2 - x_1}{x_2} * \frac{100}{1} \tag{10}$$

This helps to visually illustrate the trend in plan quality of both the fixed and the controlled experiment. A decreasing ($\downarrow$) trend in the value of y implies a good quality plan while a continuous increase ($\uparrow$) in the value of y means that the planner output is affected by the complexity of the problem in the domain. The more complex the problem becomes the more the challenge to generate quality plan at a reasonable time. Moreover, when y is zero, it means the output plan is steady and stable despite an increase in problem complexity.

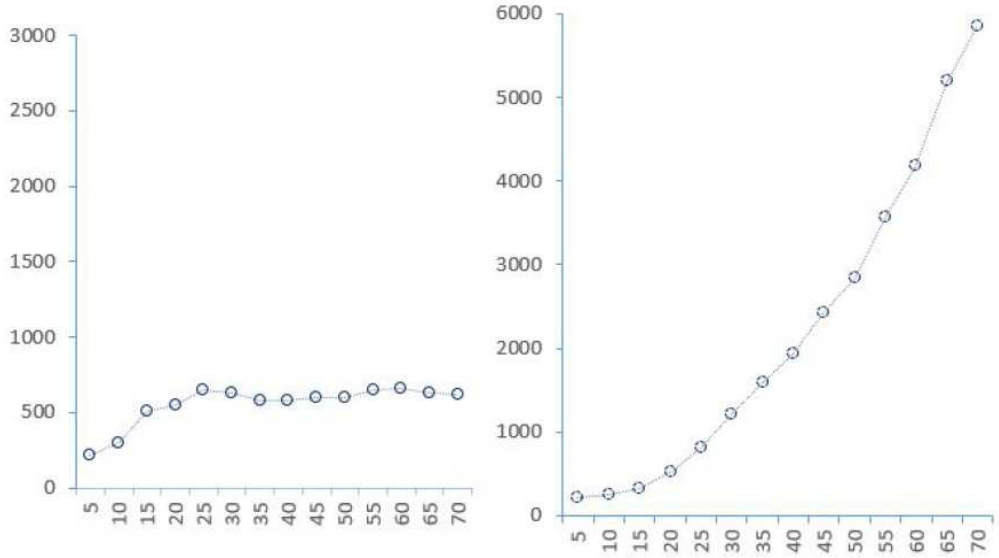

Fig. 7. The run-times for both Controlled (left) and Fixed (right) Signal. The y-axis indicate the time taken to output a complete plan (run-time) in microseconds, the queue length size is represented by the x-axis. An increasing queue length signifies a more congested network; consequently, an increasing problem complexity.

Table 1. Planner result showing the percentage increase in number of vehicles in the network and the corresponding percentage changes(effect) in the plan metrics. Fixed duration means the duration of the green split is fixed at the initial state and would be the same throughout the planning time. Controlled means that the duration is fixed at the initial state, but subject to changes during search space whenever the planner anticipate a better optimised green time than the fixed value

QueueLenght Variation	Increase in Queue Lenght(%)	Change in Avg. planning time (%)		Change in Avg. No. of processes (%)		Change in Avg. No. of actions (%)	
		Fixed	Controlled	Fixed	Controlled	Fixed	Controlled
5	↑ 100	↑ 100	↑ 100	↑ 100	↑ 100	↑ 100	↑ 100
20	↑ 75	↑ 31.3	↑ 56.5	↑ 35.0	↑ 45.8	↑ 50	↑ 58.3
40	↑ 50	↑ 61.2	↑ 22.5	↑ 41.2	0.0	↑ 50	↑ 42.9
80	↑ 50	↑ 48.1	↓ 13.2	↑ 45.2	0.0	↑ 50	0.0
60	↑ 50	↑ 34.5	↑ 3.9	↑ 25.3	0.0	↑ 34.4	0.0
200	↑ 20	↑ 42.1	↑ 8.7	↑ 29.7	↑ 4.0	↑ 29.1	0.0
300	↑ 33.3	↑ 28.5	↓ 5.7	↑ 22.9	0.0	↑ 22.5	0.0

Test Environment

The UTCPLAN algorithm is implemented in Netbeans Java 8.0 which involves the creation of a continuous planner with an embedded MPC approach. The domain and problem representation (traffic description) are also developed in Java to facilitate easy data transfer between planner and network information description. The experiment was run on Ubuntu 15.04, Intel Core i7 on a 16GB RAM at 2.20GHz.

Result

The plan contains the sequence of action operators needed to optimise traffic flow within an urban traffic network until the goal condition is satisfied. Figure 7 shows an excerpt of a sample plan generated by UTCPLAN for a controller to solve a UTC control problem instance.

Empirical Analysis

A output plan is the sequence of steps needed to get to a goal condition from an initial problem situation. The total length of a plan for a given problem varies from planner to planner. The shorter the length of the generated plan, the better the quality of the plan. The lesser the number of actions and processes needed to achieve a goal condition the better the quality of the plan for such problem domain.

The average total time taken to generate a plan is a metric that shows the efficacy and speed of the planner. The total time depends majorly on the planner algorithm. It is also dependent on some other factors such as the language used to implement the planner and the hardware configuration of the system that the planner resides on. The faster it is to achieve the goal condition the lesser the total time to generate a plan and vice versa. The total time taken to generate a plan is an essential

criterion for the evaluation of planners in AI planning. A planner is effective in a domain of problem if the total time to generate a plan for problems in that domain remains steady and stable. However, if the total time to produce a solution in a domain of problem is astronomically increasing with an increase in the complexity of the problem, it means the planner might get stuck during certain problem situation in such domain.

Table 1 presents the percentage rate of increase in queues within the network and the effect of those percentage increase on the average total time as illustrated in Fig. 7. It is observed that the average total time required to generate a plan varies with a variation in queuing distance and the green split values. The percentage change in total time increases with an increase in queue length at fixed signal. However, the percentage change in the total time of controlled signal is remarkable at a low increase rate with increase in queue length.

The trend in the percentage change in average number of processes initiated by generated plans is also shown in Table1. The percentage change in the average number of processes increases with increase in queue length at fixed signal. However, the percentage change in the average number of processes is reduced to zero percent despite an increase in queue length when the signal is controlled by UTCPLAN. It increases a little when the length of the queue reaches close to 200 m but later drop back to zero percent despite a further increase in queue length. The total number of processes initiated by the planner to achieve the goal condition increases with an increase in the congestion rate whenever the signal is fixed as shown in Fig. 8. However, the changes are minimum and often becomes steady despite the increasing queues in the network when the green split is controlled by the UTCPLAN approach within the traffic network.

Similarly, Table 1 shows the trend in percentage change in the average number of action operator within the plans. This increases with an increase in queue length at fixed signal. However, the percentage change in the average number of action operator is reduced to zero percent despite an increase in queue length when the signal is controlled by UTCPLAN. The total number of actions generated by the planner to achieve the goal condition increases with an increase in the traffic congestion rate whenever the signal is fixed. However, the changes are also minimum and often becomes steady despite the increasing queues in the network when the green split is controlled by UTCPLAN approach within the traffic network as illustrated in Fig. 9.

Discussion

The percentage change in output value gives a visual illustration of the trend in plan quality of both the fixed and the controlled experiment. A decreasing (↓) trend in the output value implies a good quality plan while a continuous increase (↑) in output value means that the planner output is affected by the complexity of the problem in the domain. The more the complexity of the problem, the higher the challenge to generate quality plan at a reasonable time. Moreover, when the percentage change in output value is zero, it means the output plan is steady and stable despite an increase in problem complexity as illustrated by Fig. 7-9.

Stability in plan metrics can not be achieved by a planner with fixed duration. It can only be achieved by a planner that can establish a unique approach to numeric fluents during search space. The stability in the controlled output plan metric is achieved through the novel integration of MPC approach with AI planning.

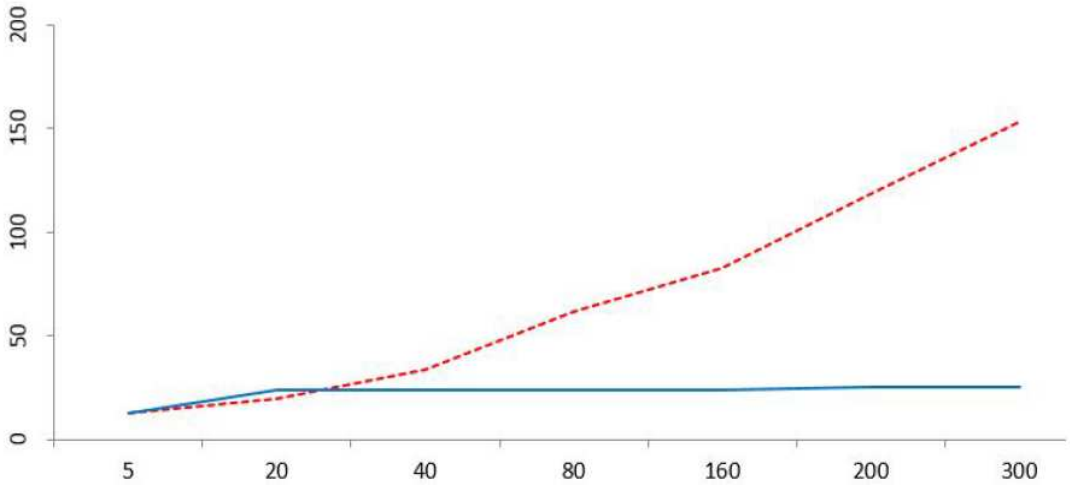

Fig. 8. Average number of processes initiated by UTCPLAN plans with fixed and controlled traffic signal. The y-axis shows the average number of processes, the x-axis represents the size of the queue length

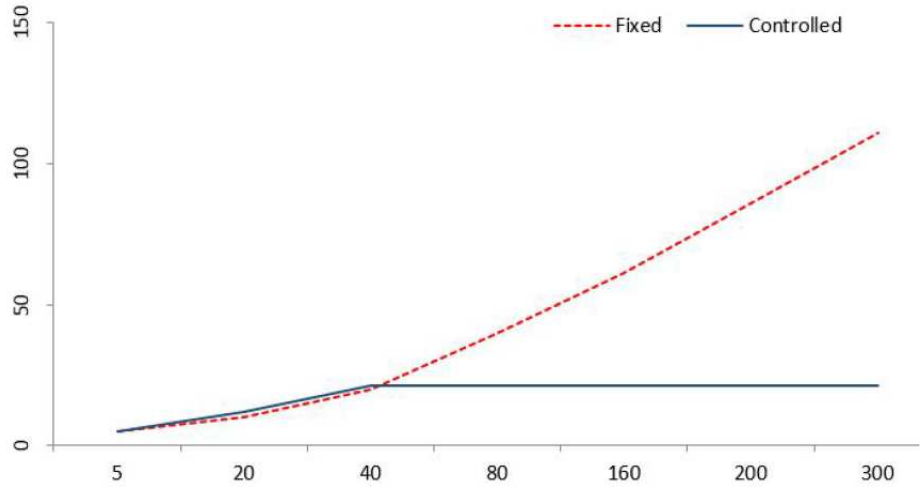

Fig. 9. Average Number of Action steps in UTCPLAN Plans with Fixed and Controlled Traffic Signal. The y-axis shows the Average Number of Actions, the x-axis represents the size of the queue length

This implies that the time to generate a valid plan, as well as the quality of plan generated, becomes stable at some point irrespective of the increase in complexity of the problem domain. For instance, the result shows that a controlled approach is required to optimise any traffic situation. The effectiveness of UTCPLAN approach at tracking and predicting numeric changes, while evaluating the effect of those changes during search space, helps to anticipate increasing or decreasing queue trends within the network. The controlled green time is always suited to the changes in the network. This helps to keep the network in a stable state despite increasing congestion.

The result indicates a favourable output in both signal test cases when planning with tasks of less complexities. It is inferred from the result that the fixed and controlled signal approach produce excellent control performance during a lesser traffic situation. However, a vast output difference is observed between the two instances when planning with tasks of higher complexities. It is inferred from the result that the run-time of controlled signal increases initially, then become steady despite an increase in traffic congestion and bottleneck. While the run-time of the fixed signal gets worse with increasing traffic congestions and bottleneck as shown in Fig. 7 (right side), because large traffic demand generates huge search space and, therefore, the solution requires more computational time especially at lower fix duration.

The total number of actions sequence and initiated simulation in the plan generated by the fixed signal is 45% above the controlled signal plan. Thus, the controlled plan is has a lesser plan length in over 80% of the tasks in the test suite compared with the fixed generated plan. This evidence confirms that UTCPLAN generates a more quality plans. Another benefit of the controlled instance is the ability to reach the goal condition in lesser time for most of the problem instances, though the domain courage is the same for both configurations (both test instances solved all the modelled problems in the domain).

The creation of a rich declarative representation of the UTC model facilitates reasoning with logical constants, variables and constraints within the model; but a classical MPC formulation might not take logical formalities into consideration. However, the MPC mathematical formulation and computation of UTC numerics within the model, facilitate dynamic control of traffic signal and vehicle routing; this might not be effectively achieved by classical AI planning search mechanism. Integrating and utilising the two approaches create an effective control of continuous numerics combined with the logical component within a model.

Scaling Difficulties

UTCPLAN currently, does not have a built-in specific heuristics for pruning the search space. Integrating advanced planning solvers into the search pattern of this implementation would boost the speed of planner. The implementation made use of a simple classical numeric solver; the use of a state-of-the-art commercial solver would enhance the robustness and scalability of UTCPLAN to deal with a larger network of constraints in future implementation.

Conclusion

We introduce UTCPLAN, a planning system that embeds model predictive approach into an AI planning search paradigm. UTCPLAN supports the analysis of domain descriptions containing continuously changing processes, events and actions. Experimental evaluation shows that our novel approach can control traffic and reduce congestion when tested on a sample road

network. The application to Urban Traffic domain is utilised to validate the practicability of this novel hybrid integration on a continuous domain with logical preferences. The result shows that UTCPLAN can reason with continuous processes in the domain and has the potential to generate control and execution plans and schedules that will keep such domain in a desirable state.

Acknowledgement

The authors acknowledge the European Commission for Science and Technology (COST) funding for the Autonomic Road Transport Supports grant (Grant Ref: COST-ARTS/TU1102).

Author's Contributions

Dr. Jimoh Falilat Olaitan: Made considerable contributions to the conception of this hybrid technology; she contributed to the original design and revamping of the algorithms. She also contributes in critical reviewing of the manuscript for significant intellectual content.

Professor Thomas Leo McCluskey: Made considerable contributions during the design of this hybrid technology. He contributed to the revamping of the algorithms. He also contributes in critical reviewing of the manuscript for significant intellectual content.

Dr. Simon Parkinson: Made considerable contribution to the drafting of this article; he contributed to the presentation and analysis of result; he made critical review for significant intellectual content and also added genuine content where applicable.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and there are no ethical issues involved.

References

Al-Gherwi, W., H. Budman and A. Elkamel, 2011. A robust distributed model predictive control algorithm. J. Process Control, 21: 1127-1137. DOI: 10.1016/j.jprocont.2011.07.002

Alanqar, A., H. Durand and P.D. Christofides, 2017. Error-triggered on-line model identification for model-based feedback control. AIChE J., 63: 949-966. DOI: 10.1002/aic.15430

Audemard, G., P. Bertoli, A. Cimatti, A. Kornilowicz and R. Sebastiani, 2002. A SAT based approach for solving formulas over boolean and linear mathematical propositions. Proceedings of the 18th International Conference on Automated Deduction, Jul. 27-30, Springer, UK., pp: 193-208. DOI: 10.1007/3-540-45620-1_17

Baldea, M., J. Du, J. Park, I. Harjunkoski, 2015. Integrated production scheduling and model predictive control of continuous processes. AIChE J., 61: 4179-4190. DOI: 10.1002/aic.14951

Bennett, S., 1993. A History of Control Engineering 1930-1955. 1st Edn., Peter Peregrinus, Hitchin, Herts., UK, UK.

Camacho, E. and C. Bordons, 1999. Model Predictive Control. 1st Edn., Springer, London, ISBN-10: 3540762418, pp; 280.

Chen, Y., L. Cheng, H. Wu, X. Zhao and J. Han, 2015. Knowledge-driven path planning for mobile robots: Relative state tree. Soft Comput., 19: 763-773. DOI: 10.1007/s00500-014-1299-4

Chu, Y. and F. You, 2015. Model-based integration of control and operations: Overview, challenges, advances and opportunities. Comput. Chem. Eng., 83: 2-20. DOI: 10.1016/j.compchemeng.2015.04.011

da Silva Fonseca, J.P., A.R. de Sousa, M.V.M. Ferreira and J.J.P.Z. de Souza Tavares, 2016. Planpas: Plc and automated planning integration. Int. J. Comput. Integrated Manufact., 29: 1200-1217. DOI: 10.1080/0951192X.2015.1067909

De Oliveira, D. and A.L.C. Bazzan, 2009. Multiagent learning on traffic lights control: Effects of using shared information.

Dinapoli, N., G. Chiloiro, G. Mattiucci, L. Azario and M. Gambacorta *et al*., 2016. Ep-1636: Clinical validation of automated planning process in rectal cancer imrt treatment. Radiotherapy Oncol., 119: S763-S764. DOI: 10.1016/s0167-8140(16)32887-0

Dusparic, I., J. Monteil and V. Cahill, 2016. Towards autonomic urban traffic control with collaborative multi-policy reinforcement learning. Proceedings of the IEEE 19th International Conference on Intelligent Transportation Systems, IEEE Xplore Press, pp: 2065-2070. DOI: 10.1109/ITSC.2016.7795890

Falugi, P., S. Olaru and D. Dumur, 2010. Robust multi-model predictive control using LMIs. Int. J. Control, Automat. Syst., 8: 169-175. DOI: 10.1007/s12555-010-0122-y

Fox, M. and D. Long, 2003. PDDL2.1: An extension of PDDL for expressing temporal planning domains. J. Art. Int. Res., 20: 61-124.

Fox, M. and D. Long, 2006. Modelling mixed discrete-continuous domains for planning. J. Art. Int. Res., 27: 235-297.

Garrido, A., E. Onaindia and F. Barber, 2001. A temporal planning system for time-optimal planning. Proceedings of the Portuguese Conference on Artificial Intelligence, (CAI' 01), Springer, Berlin, Heidelberg, pp: 379-392. DOI: 10.1007/3-540-45329-6_37

Graf Plessen, M.M. and A. Bemporad, 2017. Reference trajectory planning under constraints and path tracking using linear time-varying model predictive control for agricultural machines. Biosyst. Eng., 153: 28-41. DOI: 10.1016/j.biosystemseng.2016.10.019

Grosso, J.M., C. Ocampo-Martinez and V. Puig, 2016. Reliability–based economic model predictive control for generalised flow–based networks including actuators' health–aware capabilities. Int. J. Applied Math. Comput. Sci. 26: 641-654. DOI: 10.1515/amcs-2016-0044

Guo, C., X. Gang and M. Zhang, 2014. Model predictive control implementation and simulation for urban traffic networks. Proceedings of the IEEE International Conference on Service Operations and Logistics and Informatics, Oct. 8-10, IEEE Xplore Press, Qingdao, China, pp: 334-340. DOI: 10.1109/SOLI.2014.6960746

Gupta, S.K., D.S. Nau and W.C. Regli, 1998. IMACS: A case study in real-world planning. IEEE Intell. Syst. Applic., 13: 49-60. DOI: 10.1109/5254.683210

Heinrich, B., M. Klier, S. Zimmermann, 2015. Automated planning of process models: Design of a novel approach to construct exclusive choices. Decis. Support Syst., 78: 1-14. DOI: 10.1016/j.dss.2015.07.005

Hoffmann, J., 2003. The metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. J. Art. Int. Res., 20: 291-341. DOI: 10.1613/jair.1144

Hoogendoorn, S.P. and P.H.L. Bovy, 2001. State-of-the-art of vehicular traffic flow modelling. Delft University of Technology, Delft.

Ji, J., A. Khajepour, W.W. Melek and Y. Huang, 2017. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. IEEE Trans. Vehicular Technol., 66: 952-964. DOI: 10.1109/TVT.2016.2555853

Jimoh, F., 2015. A synthesis of automated planning and model predictive control techniques and its use in solving urban traffic control problem. PhD Thesis, University of Huddersfield.

Jimoh, F., L. Chrpa, T. McCluskey, 2014. The application of planning to urban traffic control. Proceedings of the 24th International Conference on Automated Planning and Scheduling, Jun. 21-26, Portsmouth, NH, USA.

Jimoh, F., L. Chrpa, T. McCluskey and M.M.S. Shah, 2013a. Towards application of automated planning in urban traffic control. Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems, Oct. 6-9, IEEE Xplore Press, The Hague, Netherlands, pp: 985-990. DOI: 10.1109/ITSC.2013.6728360

Jimoh, F., L. Chrpa and M. Vallati, 2013b. Autonomic system architecture: An automated planning perspective. Proceedings of the IFIP International Conference on Artificial Intelligence Applications and Innovations, Springer, Berlin, pp: 121-130. DOI: 10.1007/978-3-642-41142-7_13

Jimoh, F. and T. McCluskey, 2012. Using automated planning to enable autonomic properties in computer systems.

Jimoh, F. and T.A.A.P.P. McCluskey, 2015. Self-Management in Urban Traffic Control. 1st Edn., Autonomic Systems, Springer, Birkhuser Basel.

Jimoh, F. and T.L. McCluskey, 2016. Towards the integration of model perdictive control into an AI planning framework. Proceedings of the 34th UK Workshop on Planning and Scheduling, (WPG' 16).

Jimnez, S., F. Fernndez and D. Borrajo, 2013. Integrating planning, execution and learning to improve plan execution. Comput. Intell., 29: 1-36. DOI: 10.1111/j.1467-8640.2012.00447.x

Joos, A., C. Seiferth, L. Schmitt and W. Fichter, 2017. Parameters for nonlinear model predictive control in unmanned aerial vehicle path-planning applications. J. Guidance Control Dynam., 40: 484-492. DOI: 10.2514/1.G000311

Khoshnevis, B. and Q. Chen, 1991. Integration of process planning and scheduling functions. J. Intell. Manufact., 2: 165-175. DOI: 10.1007/BF01471363

Laguna, J.O., A.G. Olaya, D.B. Millan, 2014. Building planning action models using activity recognition. Ph.D. Thesis, Universidad Carlos III de Madrid.

Lhr, J., P. Eyerich, T. Keller and B. Nebel, 2012. A planning based framework for controlling hybrid systems. Proceedings of the Twenty-Second International Conference on International Conference on Automated Planning and Scheduling, Jun. 25-29, AAAI Press, Atibaia, São Paulo, Brazil, pp: 164-171.

Machado, L., R. Prikladnicki, F. Meneguzzi, C. de Souza and E. Carmel, 2016. Task allocation for crowdsourcing using ai planning. Proceedings of the 3rd International Workshop on CrowdSourcing in Software Engineering, May 14-22, ACM, Austin, Texas, pp: 36-40. DOI: 10.1145/2897659.2897666

Mahalingam, V. and A. Agrawal, 2016. Learning agents based intelligent transport and routing systems for autonomous vehicles and their respective vehicle control systems based on Model Predictive Control (MPC). Proceedings of the International Conference on Recent Trends in Electronics, Information and Communication Technology, May 20-21, IEEE Xplore Press, Bangalore, India, pp: 284-290. DOI: 10.1109/RTEICT.2016.7807828

Mezghiche, A., M. Moula and L. Tadj, 2015. Model predictive control of a forecasting production system with deteriorating items. Int. J. Operat. Res. Inform. Syst., 6: 19-37. DOI: 10.4018/IJORIS.2015100102

Osusky, J. and V. Vesely, 2015. Design of robust controller with input constraints. Proceedings of the 20th International Conference on Process Control, Jun. 9-12, IEEE Xplore Press, Strbske Pleso, Slovakia, pp: 261-265. DOI: 10.1109/PC.2015.7169973

Parkinson, S. and A.P. Longstaff, 2015. Multi-objective optimisation of machine tool error mapping using automated planning. Expert Syst. Applic., 42: 3005-3015. DOI: 10.1016/j.eswa.2014.11.066

Parkinson, S., A.P. Longstaff and S. Fletcher, 2014. Automated planning to minimise uncertainty of machine tool calibration. Eng. Applic. Artificial Intell., 30: 63-72. DOI: 10.1016/j.engappai.2014.02.002

Pinto, J., J. Sousa, F. Py, K. Rajan, 2012. Experiments with deliberative planning on autonomous underwater vehicles. Proceedings of the IROS Workshop on Robotics for Environmental Modeling, (REM' 12), Algarve, Portugal.

Roncoli, C., I. Papamichail and M. Papageorgiou, 2016. Hierarchical model predictive control for multi-lane motorways in presence of vehicle automation and communication systems. Transport. Res. Part C, 62: 117-132. DOI: 10.1016/j.trc.2015.11.008

Russell, S.J., P. Norvig, J.F. Canny, J.M. Malik and D.D. Edwards, 1995. Artificial Intelligence: A Modern Approach. 1st Edn., Prentice Hall, Englewood Cliffs.

Schildbach, G. and M. Morari, 2016. Scenario-based model predictive control for multi-echelon supply chain management. Eur. J. Operat. Res., 252: 540-549. DOI: 10.1016/j.ejor.2016.01.051

Shin, J. and E. Davis, 2005. Processes and continuous change in a SAT-based planner. Artificial Int., 166: 194-253. DOI: 10.1016/j.artint.2005.04.001

Tay, M., 2007. Model predictive cost control. Control Eng., 54:, IE9-IE9.

Tierney, K., A.J. Coles, A. Coles, C. Kroer and A.M. Britt *et al*., 2012. Automated planning for liner shipping fleet repositioning. Proceedings of the 22nd International Conference on International Conference on Automated Planning and Scheduling, Jun. 25-29, AAAI Press, Brazil, pp: 279-287.

Zhu, J., Q. Yang, X. Xu and J. Lu, 2015. A LPV model-based chilled water temperature controller for HVAC systems. Build. Services Eng. Res. Technol., 36: 368-385. DOI: 10.1177/0143624414555811