# Towards The Integration of Model Predictive Control into an AI Planning Framework

**Falilat Jimoh** and **Thomas L. McCluskey**
PARK Research group
University of Huddersfield

## Abstract

This paper describes a framework for a hybrid algorithm that combines both AI Planning and Model Predictive Control approaches to reason with processes and events within a domain. This effectively utilises the strengths of search-based and model-simulation-based methods. We explore this control approach and show how it can be embedded into existing, modern AI Planning technology. This preserves the many advantages of the AI Planning approach, to do with domain independence through declarative modelling, and explicit reasoning, while leveraging the capability of MPC to deal with continuous processes computation within such domains. The developed technique is tested on an urban traffic control application and the results demonstrate the potential in utilising MPC as a heurisic to guide planning search.

## Introduction

The area of domain independent planning involves modelling a system in a knowledge-based way, with declarative data structures representing goals, states, resources, actions to mention a few, and creating tools that can reason about them logically. Plans are created as output to achieve goal conditions in a future state. As well as the flexibility of input language, a characteristic of this approach is that the human user can understand and inspect the output plan, to help validate the approach, and to promote a mixed-initiative interaction with the system operators. Automated planning and scheduling is increasingly been used to solve real-world planning problems. Current limitations in state-of-the-art planning algorithms present significant limitations when trying to plan for domains which contains continuous numeric change. Few planning engines can input models described by continuous processes, and the computational complexity of the planning problem can be prohibitive without an appropriate planning approach or strong heuristics.

Control Engineering researchers, on the other hand, have developed techniques to control a continuous process by the varying of a control variable over a fixed time horizon. The advantages of Control Engineering approaches in this area, such as embedded in the popular "Model Predictive Control"(MPC) technique, is that they can work with dynamical systems described as continuously changing processes, and output controls that take into account future events. On the other hand, these approaches are not as flexible as the AIP & S approaches.

This work aims to inegrate Control Engineering and AI Planning techniques by formulating plan generation algorithms which combine the advantages of both approaches into one unified approach(Jimoh 2015). The algorithms should be able to produce readable plans to achieve goal conditions where the dynamical system model includes continuous processes. In this paper, a hybrid approach is presented where the Model Predictive Control (MPC) approach is used in conjunction with a traditional state-based planning system. This allows for the effective planning of both logical and numerical change in a problem specification.

The resulting approach, Model predictive control Approach to Continuous Planning (MACOP), is able to generate plans in domains containing actions, events and processes; discrete, real-valued, and interval-valued fluents; and continuous change to quantities. While there has been a growing amount of interest in seeking to unite work in control engineering and AI hybrid planning (e.g. (Löhr et al. 2012), MACOPs novel contribution is to use the control output from an MPC routine as a heuristic within an AI planner - that is to guide forward search planning using a discretized problem and domain model.

This paper describes an implementation of the planner used to control the light signals in the application area of urban traffic conntrol using a flow model of traffic. We are targeting those applications which require a plan to be generated a priori (rather than planning in a tight plan-execution loop), such as in Urban Traffic Control (UTC). While given the uncertainty in the UTC area, a plan - re-planning loop is often necessary, but there is also a requirement to validate such automated strategies by inspection prior to execution (e.g the problem to be solved might be planning for a known road closure or predicted saturated road conditions). This paper shows the feasibility of using this kind of hybrid approach to generate plans for such problems involving hybrid states. While the paper describes a domain dependent planner, we propose that the framework could be used with an existing planner as the basis for a domain independent version.

## Background

Predictive controls are a branch of Control Engineering that are used in adapting and forecasting the future trend of control processes in other to manipulates it inputs for a desirable result in a future time. There exist different types of predictive controls, for example, the Receding Horizon Predictive Control(RHC); the Generalised Predictive Control (GPC) and the Model Predictive Controls (MPC). MPC is used to predict the future behaviour of processes or output of a system over a period of time in the future. This is achieved by computing the future input variables at each step while minimising a cost function under disparity constraints on the manipulated controls and controlled variable. MPC applies only the first set of control variables on the controlled system and repeats the previous step with new measured numeric variables (Veselý, Rosinov, and Foltin 2010). MPC has attracted notable attention in the control of dynamic systems and has gained an important role in process control. It was developed in the industrial area as an alternative algorithm control to the conventional Proportional Integrate Derivative (Bennett 1993) (PID) due to it ability to reason with the *model* of a system under consideration. The MPC formulation integrates optimal control, stochastic control, control of processes with dead time, multi-variable control and future references when available (Camacho and Bordons 1999). There are various MPC algorithms, these algorithms has been constantly improved and refined to increase its robustness for real time processes (Tay 2007; Al-Gherwi, Budman, and Elkamel 2011).

Automated Planning with continuous processes allows us to model actions, process and events as part of domain operators. This extends temporal planning by considering constantly changing state variables with respect to time. Automated planning in domains which are represented with rich notations has long been a great challenge for AI (Bresina et al. 2013). For instance, changes occurring due to fuel consumption, continuous movement, or environmental conditions may not be adequately modelled through instantaneous or even durative actions; rather these require modelling as continuously changing processes. The combination of time dependent problems and numeric optimisation problem create a more challenging and hard task of time-dependent metric fluents.

One of the earliest works that involves planning with continuous processes includes the Zeno system (Penberthy and Weld 1994). In this case, processes are described using differential equations rather than as continuous update effects, so that simultaneous equations must be consistent with one another rather than accumulating additive effects. McDermotts OPTOP system (McDermott 2003) is another early planner to handle continuous processes. A forward search planner that avoids grounding the representation using a unique approach to generate heuristics(relaxed plan estimates of the number of actions required to achieve the Goal) from a given state.

More recently, the syntax and semantics of a hybrid language was brought into the PDDL family in the form of PDDL+ (Fox and Long 2006). TM-LPSAT (Shin and Davis 2005) was built upon the earlier LPSAT (Shin and Davis 2005). It was the first planner to implement the *PDDL+* Semantics and in addition had the capability of handling variable durative actions. This includes durative actions with continuous effects and duration-dependent end-effects. It uses PDDL+ semantics to compile a collection of SAT formulas from a horizon bounded continuous planning problem, together with an associated set of linear metric constraints over numeric variables. The compiled formulation is passed to a SAT-based arithmetic constraint solver, LPSAT (Audemard et al. 2002). The SAT-solver parses triggered constraints to the LP-solver, if there is no solution the horizon is increased and the process repeats, otherwise the solution is decoded into a plan. The novelty of TM-LPSAT lies in the compilation of the PDDL+ semantics and decoding of the SAT solver into a plan, since both solvers are well-established systems. Kongming (Li and Williams 2008; **?**)is another domain dependent continuous planner that solves a class of control planning problems with continuous dynamics. The language used is a version of PDDL2.1 extended to enable dynamics to be encoded. It is based on the building of fact and action layers of flow tubes, using the iterative plan graph structure of Graphplan algorithm (Blum and Furst 1995). As the graph is expands, every action produces a flow tube which contains the valid trajectories as they develop over a period of time. Reachable states at any time can be computed using the state equations of the system starting from a feasible region, and applying actions whose preconditions intersect with the feasible region. Kongming translates a planning problem into Mixed Logical-Quadratic Program (MLQP) using the plan-graph encoding with the continuous dynamics of the system. The planners metric objectives function can be defined in terms of quadratic function of state variable. Time is discretised to support state update within the plan - successive layers of the graph are separated by a constant and uniform time increment.

UPMurphi is another planner that reasons with continuous processes (Penna et al. 2010). It alternatively refines a discretisation of continuous changes until the solution to the discretised problem validates against the original problem specification. UPMurphi starts by discretising the continuous representation of the problem. Specific values within feasible ranges are taken as actions, giving rise to several version of each action. The current discretisation is then used to explicitly construct and explore the state space. Plans are constructed in the form of planning-as-model-checking paradigm (Cimatti et al. 1997) with no heuristic to guide the search (users can insert their own). When a plan is found, it is validated against the original continuous model, using the plan validator (Fox, Howey, and Long 2005). If it fails to find a plan at one discretisation, it iterates again at a finer grained discretisation. Successive refinements lead to ever denser feasible regions, which might be increasingly complex to construct.

COLIN (Coles et al. 2012) is a forward-chaining heuristic search planner is capable of reasoning with continuous linear numeric change. It combines forward chaining search of FF to prune state, with the use of a Linear Program (LP) to check the consistency of the interacting temporal and numeric constraints at each state. The Temporal Relaxed Plan-

ning Graph heuristic of CRIKEY3 (Coles et al. 2008) is also extended to support reasoning with continuous change. A mix integer programming is used for post processes to optimise the timestamps of the actions in the plan.

As the diversity of potential planning applications has increased, so has the complexity of the continuous domain knowledge. In order to circumvent this limitation, some researchers are currently *relaxing* the complexity of the domain by discretising continuous change into discrete profiles of linear change (Piacentini et al. 2013). in a similar vein, in Domain Predictive Control (Löhr et al. 2012), a discrete domain model is derived from the equations governing dynamics in the application domain, and AI planning is used to generate plans (using durative planning with PDDL 2.1). The application area is continuous (re-)planning in "switched hybrid systems". The idea is to start with the dynamical equations, and generate a discrete domain model from that; this contrasts with the work in this paper, which assumes that we create a symbolic domain model containing processes, events and actions, then use MPC derived from a model of dynamical equations as a heuristic to control forward search in a symbolic planning search space. While in the DPC work the emphasis is on applications requiring real time control, in our work we are more interested in creating a complete readable plan before execution.

## The MACOP Framework

In overview, MACOP inputs a planning domain model and problem, and searches through a space of nodes using a best-first heuristic to find a complete plan. A node is a point in a search space at which search frontiers or pathways intersect or branch. Operators' preconditions are checked against propositions and numeric fluents at each node, if one is satisfied, the operator effect is applied and the new state becomes the current state. The model-based numeric optimisation problem within the domain model is solved at specific nodes during node exploration. The search proceeds by applying each applicable operator to the current states in a receding horizon until a goal state is found or the node set is empty. The following subsection gives a description of the details of the algorithm.

### MACOP Algorithm Preliminaries

This section contains definitions that are fundamental to the design of the planner algorithm.

**Definition 1 (State)** *A state S is a pair $\langle P,R\rangle$, where P is the set of atomic propositions and R is an assignment of numeric variables to values. A state describes what is true of some world at a snapshot of time assuming a Closed World Assumption on S.*

**Definition 2 (Initial State)** *An Initial State is a state $\langle P,R\rangle$ that is true at the start of some planning problem.*

**Definition 3 (Goal Condition)** *A Goal Condition is a condition $G = \langle Q,N\rangle$, where Q is a set of atomic propositions, and N is a set of conditions on numeric variables. For a goal to be achieved in some state $(P,R)$, Q must be contained in P, and the values of numeric variables in R must satisfy the conditions in N.*

**Definition 4 (Domain Model)** *A Domain Model DM, consist of:*

- *A set of Propositions* $\{p_1,...,p_k\}$
- *A set of numeric Functions* $\{n_1,...,n_k\}$
- *A set of Resources* $\{r_1,...,r_k\}$
- *A set of Actions* $\{a_1,...,a_k\}$
- *A set of Processes* $\{c_1,...,c_k\}$
- *A set of Events* $\{e_1,...,e_k\}$

The domain description language syntax and semantics used in this implementation is similar to *PDDL+* (Fox and Long 2006). To save space, we refer the reader to the definition of PDDL+ for prelimiary definitions. The exact definitions (which are at variance a little with PDDL+) are given in Jimoh's thesis[1]. The example action, process and event definitions in Figs 1 - 3 below, taken from the flow model of urban traffic control that we will develop later in the paper, illustrate the idea.

**Algorithm 1** Top level algorithm of MACOP

**Input:**
(P,R) : initial state
G: Goal Condition
DM: Domain Model
$N_c$ : control horizon window
$N_p$ : horizon prediction window
**Output:** Plan.

1: $\Re$ := [ R ]; $S$ := [ ]; $\wp := null$;
2: $n := (P,\Re,S)$
3: **repeat**
4: $Q := \{n\}$
5: i := 1
6: $\wp := SolveMPC(n,DM,N_c,N_p,\wp)$
7: **while** $Q \neq \{\}$ and $i \leq N_c$ and $NoSolution(Q)$ **do**
8: $n := getBest(Q,\wp)$
9: $N := Expand(n)$
10: $Q$ := AddTo($N,Q$)
11: $i := i+1$
12: **end while**
13: **if** $Q \neq \{\}$ and $NoSolution(Q)$ **then**
14: $n := getBest(Q,\wp)$
15: **end if**
16: **until** $Q = \{\}$ or $SolutionFound(Q)$

### Top level algorithm of MACOP

The input to the planner is the initial state, goal condition and the domain model as defined in the preliminary definitions. As well as this, tied to the particular domain that the system is being applied to, are the fixed horizon prediction value $N_p$ (the amount of time for which the MPC component will form a guiding plan into the future), and the value

[1]"A Synthesis of Automated Planning and Model Predictive Control Techniques and its Use in Solving Urban Traffic Control Problem", F. Jimoh, 2016

```
(:action  switch_to_green
:parameters [at_junction, this_phase, from_road1, to_road2]
:precondition [(intersect at_junction this_phase from_road1 to_road2)]
                [(>(queueLenght (from_road1 0.0))
                (<(interuptLevel (to_road2 7.0))]
:effect ([(JflowActive at_junction this_phase from_road1 to_road2)])  )
```

Figure 1: Sample of an Action Declaration

```
 (:process  Jtraffic_flow
:parameters [at_junction, this_phase, from_road1, to_road2]
:precondition [(JflowActive at_junction
                this_phase from_road1 to_road2)]
:effect ([(decrease(queueLenght (from_road1 (* #t flowrate)))
        (increase(queueLenght (to_road2 (* #t flowrate)))])  )
```

Figure 2: Sample of an Process Declaration

```
 (:event  upstreamFilled
:parameters [at_junction, from_road1, to_road2]
:precondition [(>=(queueLenght (to_road2 capacity_of_road2)))]
:effect ([(assign(queueLenght (to_road2 capacity_of_road2))
           (assign(interuptLevel (to_road2 7.0))])  )
```

Figure 3: Sample of an Event Declaration

of the control horizon window $N_c$ (the number of nodes that are searched between MPC prediction episodes). Both these values are determined a priori for the domain and kinds of problems that the planner is aimed at.

A node in the search space is made up of 3 components: (1) a set of propositions as previously defined as the "P" component of a state (2) a *sequence* of the numerical variable components in the "R" component of the state, the sequence capturing their history over a fixed time horizon (3) a partial plan. Lines 1-2 initialise a node. $\wp$ is the variable that stores dynamic prediction values output from the MPC process over successive horizons, and is initially set to null.

Within the outer loop Line 4-5 initialise the search space, and in Line 6 the MPC optimization process is called. This returns numeric values for control variables which can be interpreted as a set of *predicted actions* that if executed will lead towards the objective function.

The inner loop (Line 7 - 12) expands search within a fixed horizon $N_c$. Within the loop the best open node n is chosen and removed from $Q$. This choice is informed by the output of MPC: the node which is closest to the trajectory given by the partial plan in the current $\wp$ is picked. Currently, MACOP uses no other built-in specific heuristics based on the goal condition. In Line 9 the node chosen for expansion is then expanded to return a set of successor nodes, N, which are added to the open node set. Details of the expansion algorithm is given in Algorithm 2 below. After the inner loop exits, if no solution has been found, then the best node is chosen, extracted from $Q$, and used as the start node for a new search within a new horizon. While the selection of one node creates incompleteness in the system, it limits the search, and uses the direction from the MPC component to decide which the best node is to go forward with. Line 16 repeats the search and optimisation processes from the current node until *SolutionFound* flag is true or the search cannot find a solution (the open node set is empty).

## Expanding Search Nodes

An action is an instance of an operator within the domain model. Its preconditions could be logical, or both logical and numeric inequalities and its effect are logical and/or numeric updates to the current state where the action is executed. For instance, the action 'switch to green' in Fig 1 has a logical precondition that the two roads must be intersected at the junction and must be sharing the same green phase. It also has numeric preconditions such as the interrupt level of the connected road must be less that interrupt level seven. The effect of this action changes the logical state of the phase at the junction of the two roads to be active, which subsequently starts a process at that junction in the next node.

Search space expansion of the current node $n$ occurs by applying actions that satisfy the conditions at a node, or by time passing for a unit of time. The effect of an action changes the state at a node, as shown in Algorithm 2. This makes certain assumptions on the semantics of events, such as different orders of simultaneous events make no difference.

When a process is initiated at a given node, the process will run while its precondition holds. Whenever there is a process in our domain representation there must always be a corresponding event to monitor and control the process.

A grounded process within the domain runs for a period

**Algorithm 2** Algorithm for Expand(n)

**Input:** n
**Output:** N

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

of time once it is initiated within the search node. The time is discretised into single step counts(E.g. $t = 1, 2, 3...t_n$) where $t_n$ is the duration of simulation of the process. Processes are started as a result of an action initiating the process or an event triggering the start of a process. The preconditions could be logical or numeric inequalities and their effects are also numeric update to the current state where the process is activated. An example of a process initialisation is the resulting effect of an action "switch-to-green" (Figure 1). This action effect could lead to a process flow of vehicles starting, from one road link to another, at the rate of flow of traffic through that junction for the duration of active green phase at the junction (Figure 2). The process would continue to run until the specified simulation time elapses or there is an internally generated event that halts the process.

## The MPC Solver: generating and using a dynamic prediction table

Whenever the horizon control value $N_c$ is reached, the stored past numeric fluents in node $n$ are used to generate a dynamic prediction table over the period of prediction horizon count $N_p$, via the *SolveMPC* procedure. The generated values are passed to a numeric optimisation procedure to compute the best control values, $\wp$, for the next set of alterations taking into considerations all the constraints in the domain.

The optimiser is a procedure which works as a planning as satisfiability(SAT) problem solver (originally used in (Audemard et al. 2002; Shin and Davis 2005)). The continuous variables with their corresponding constraint values are translated at a given search node into a linear programming problem. The best combination of inputs that satisfies the given numeric constraint are returned to the node.

For instance, assume $N_c$ is set to 500 node count and $N_p$ is set to 30 seconds. The planner keeps track of the node counts and retrieves past numeric fluents at every 500 node count. It will use this numeric information to generate a new dynamic prediction table of changes in numeric values over the period of prediction horizon of 30 seconds. The generated values will be passed to the optimiser to compute the best actions for the subsequent search period taking into considerations all the constraints in the domain.

How this table is generated depends on the application; below we illustrate this with a UTC domain.

## Application of MACOP to a UTC Domain

We illustrate the working of the MACOP framework with a specific UTC application. The application is extracted from a town center area in the United Kingdom.The domain model consists of both static and dynamic parts. The static part represents the road network topology, i.e., roads, their capacity, length and junctions connecting the roads. The road network is represented by a directed graph, where edges stand for road links and vertices stand for either junctions, source or sink roads. Sources are roads where vehicles enter the network, while sink road are roads where vehicles exit the network. The dynamic part represents the length of queues (relating to vehicle occupancy on a link) on each road and the flowrate of vehicles on such roads. The dynamic information changes as vehicles are moving through the road network.

For example, if a predicate (link nLSouth wDStr) is present in some state, then in this state the road nLSouth is linked to wDStr, thus allowing the flow of traffic from nLSouth to wDStr if all other constraints are satisfied. A UTC Planning Problem addresses the problem of effective navigation of vehicles through a given road network from source to sink roads while optimising traffic flow. This is equated to minimising the accumulated queues at each of the junctions in the network.

A "store-and-forward" traffic flow model is used to formulate a state space MPC system for this work. The store-and-forward traffic flow model was proposed in 1963 by Gazis and Potts with the desire of getting a balanced trade-off between control accuracy and computational efforts. The mathematical treatment of this model is covered by various works, and recently by Guo et al (Guo, Gang, and Zhang 2014), hence we will not repeat the mathematics here but refer the reader to that work. The method sets up:

1. a set of equations $x(k) = x(k+1) + K$ which relates the number of vehicles in a road link x at time step k, with the number of vehicles in the same link x at step k+1, given the effect of those flows over one unit of time. K encapsulates all the effects of flows into x from other links, all the flows out of x into other links, as well as the traffic signal condition at that step. Flow rates are estimated using values that are obtained from historical traffic databases. The equations gives the basic dynamical relations in the system: they take into account all the processes affecting link x at once (this is somewhat different to the domain model's process specification in Figure 2, which specifies one flow between two road links in isolation).
2. relations capturing all the constraints in the system: the maximum and minimum green times, the maximum number of vehicles that can occpy each link, etc.

The series of linear equations/relations making up (i) and (ii) are then input to a linear optimiser. The variables that the optimiser can change are the green light times, at each junction, and the objective to be optimised (minimised) is the sum of vehicles in queues. The output is the timings over the look-a-head period of changes to the lights in each junction, which minimises the overall occupancy. These timings equate to actions which change the lights over the horizon period, and hence form heuristics to the search process in the planner.

## Setup and Evaluation

The MACOP algorithm, and the embedded MPC solver, are implemented in Netbeans Java 8.0. The domain and problem representation (in this case the traffic domain description) were also developed in Java syntax to ease the parser and integration issues. The experiments were carried out on an Intel(R) Core(TM) i7-4702MQ CPU, 2.20GHz, with 16GB RAM.

We evaluated the performance of MACOP on the UTC domain previously introduced: the UTC problem is a tafffic network problem, comprising of 12 roads connected by junctions, 2 linked roads and 3 junctions. The number of vehicles on road links is meansured by the queue length. Each junction is designed to have more than one signal stage to test the ability of MACOP to split the green time between the two stages at a junction based on current state of the queue length associated to each road at the junction. The network model also has connected roads linking other roads without a signaled junction, in order to test the ability of MACOP to reason with the dynamic state of those connected roads that are not directly linked to a junction in the network.

The plans that MACOP generates contains a list of switch actions over time representing signal changes in the junctions in the region. The switch action is executed at a time when it anticipates a better optimum green stage than the fixed value during search space.

We evaluated the effectiveness of the embedded MPC approach in the MACOP algorithm to optimise traffic flow during changes in traffic situation. We tested the performance of the planner based on its ability at controlling the signaled junctions to accommodate for the changes in traffic situation. As no benchmark set exists yet for PDDL+ problems, we created a variation of MACOP, a version without MPC integrated (Fixed Signal) that reasons with numerics like a numeric planner (Hoffmann 2003).

Both instances used the same formulation of the given domain and problems:

**Fixed** The signal duration at the junctions are fixed from the initial state to the goal state: the planner decides what flows are turned on/off by the signals. Hence, an action could be to switch a (fixed duration) stage from red to green.

**Controlled** The signal is fully controlled by the planner. In this case, the signal is fixed at the initial state, but the planner can change the signal duration during search space using the embedded MPC approach in MACOP to optimise the green phase at a junction based on traffic demands within the network.

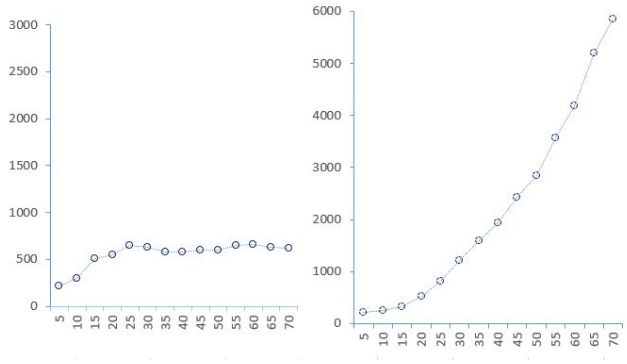


Figure 4: Run-times of MACOP with Fixed signal (right) and controlled (left). The y-axis shows run-time in microseconds (time taken to output a complete plan), the x-axis represents the size of the queue length. Thus, problems which start with an increased queue length indicate a more congested network, and hence a more complex problem.

We constructed several problems within the UTC domain with increasing complexities. These problems are useful for our evaluation as it highlights the benefits of the MPC integration (controlled signal) over the numeric AI searching (fixed signal) mechanism. Fig 4 shows the times taken by MACOP to solve problems in our test suite.

The same time discretisation is used t = 1.0, for all problems in our UTC domain.

We investigated the speed of MACOP on different volumes of traffic and bottlenecks to test the performance of MACOP during traffic congestion. We also evaluated the quality of plans generated by comparing the number of actions, processes and plan length in the fix signalled plan to the controlled signalled plan. We generated several traffic flows by increasing the percentage of queues to create heavier traffic flow in the experiment.

## Discussion

The results show that both fixed and controlled strategies perform well in lesser traffic conditions (problems with less complexities). However, there is a vast difference between the two instances when the traffic condition becomes heavier with increasing bottlenecks (i.e increasing complexities). The planning time of the controlled instance is steady while the performance of the fixed time strategy gets worse with increasing problem complexities. The total number of actions and processes in the fix instance is 45% more than the total number of actions and processes in the controlled instance. Also, the plan length of the controlled instance is less than the plan length of the fixed instance. This gives evidence that MACOP generates quality plans. In terms of coverage, both configurations are able to solve the problems in the domain, but the runtime is less in the controlled instance than the fixed instance. This shows the advantage of the more informed MACOP reaching goal conditions in less

time.

The ability to create rich representations of UTC domain makes it easier to reason with some logical constraints within the road network, in contrast to a classic MPC controller might not be able to handle them. The MPC approach, on the other hand, utilising domain-dependent knowledge, helps to dynamically control the green split which the searching mechanism might not be able to simulate. Combining the two approaches this way helps to control a signalled junction while still taking care of the logical reasoning within the network of roads.

At the moment, the planner works offline. The generation of plans is only based on the formal description of the environment. The state of the system at the time of executing the plan is assumed to be adequately modelled. Hence, a change in a state prior to plan execution is not taken into consideration. Thus, the domain model is robust enough to take care of the gaps or differences between the conceptual model and the real world. In a dynamic environment where unpredictable changes are likely to occur, however, the planner will need to have constant feedback from the effect of its actions on the domain environment. Thus, online planning will be employed in this situation.

## Conclusion

In this paper we have described a UTC dependent, but scenario and configuration independent planning system called MACOP. It supports the encoding of domains containing continuously changing processes, events and actions. We presented a new approach for such problems areas, integrating MPC and search space planning, by using MPC as a control heuristic for a discretised forward search space. We described the implementation and performance of our MACOP hybrid algorithm, when tested on a network of connected roads. While our implementation is not yet competitive with state of the art hybrid planners, we have used the application to UTC to show the feasibility and promise of this partticular type of hybrid integration. Future work includes integrating state-of-the-art heuristics into the search space planning process to improve the performance, stability and robustness of the planner. Also, a more efficient optimiser needs to be employed to improve the robustness to larger networks of constraints, rather than the simple solver that was employed in the current implementation.

## References


[Al-Gherwi, Budman, and Elkamel 2011] Al-Gherwi, W.; Budman, H.; and Elkamel, A. 2011. A robust distributed model predictive control algorithm. *Journal of Process Control* 21(8):1127–1137.

[Audemard et al. 2002] Audemard, G.; Bertoli, P.; Cimatti, A.; Kornilowicz, A.; and Sebastiani, R. 2002. A SAT-based approach for solving formulas over boolean and linear mathematical propositions. In *Proc. 18th Int. Conf. on Automated Deduction*, volume 2392, 193–208. Springer-Verlag, LNAI Series.

[Bennett 1993] Bennett, S. 1993. *A History of Control Engineering 1930-1955*. Hitchin, Herts., UK, UK: Peter Peregrinus, 1st edition.

[Blum and Furst 1995] Blum, A., and Furst, M. 1995. Fast planning through planning graph analysis. In *Proceedings of the International Joint Conference on Artificial Inteligence (IJCAI-95)*.

[Bresina et al. 2013] Bresina, J. L.; Dearden, R.; Meuleau, N.; Ramakrishnan, S.; Smith, D. E.; and Washington, R. 2013. Planning under continuous time and resource uncertainty: A challenge for ai. *CoRR* abs/1301.0559.

[Camacho and Bordons 1999] Camacho, E., and Bordons, C. 1999. *Model predictive control*. London: Springer.

[Cimatti et al. 1997] Cimatti, A.; Giunchiglia, F.; Giunchiglia, E.; and Traverso, P. 1997. Planning via model checking: A decision procedure for *r*. In Steel, S., and Alami, R., eds., *ECP*, volume 1348 of *Lecture Notes in Computer Science*, 130–142. Springer.

[Coles et al. 2008] Coles, A. I.; Fox, M.; Long, D.; and Smith, A. J. 2008. Planning with problems requiring temporal coordination. In *Proc. 23rd AAAI Conf. on Artificial Intelligence*.

[Coles et al. 2012] Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2012. Colin: Planning with continuous linear numeric change. *J. Artif. Intell. Res. (JAIR)* 44:1–96.

[Fox and Long 2006] Fox, M., and Long, D. 2006. Modelling mixed discrete-continuous domains for planning. *J. Art. Int. Res. (JAIR)* 27:235–297.

[Fox, Howey, and Long 2005] Fox, M.; Howey, R.; and Long, D. 2005. Validating plans in the context of processes and exogenous events. In Veloso, M. M., and Kambhampati, S., eds., *AAAI*, 1151–1156. AAAI Press / The MIT Press.

[Guo, Gang, and Zhang 2014] Guo, C.; Gang, X.; and Zhang, M. 2014. Model predictive control implementation and simulation for urban traffic networks. In *Proceedings of 2014 IEEE International Conference on Service Operations and Logistics, and Informatics*, 334–340.

[Hoffmann 2003] Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *J. Art. Int. Res. (JAIR)* 20:291–341.

[Jimoh 2015] Jimoh, F. 2015. A synthesis of automated planning and model predictive control techniques and its use in solving urban traffic control problem.

[Li and Williams 2008] Li, H., and Williams, B. 2008. Generative systems for hybrid planning based on flow tubes. In *Proc. 18th Int. Conf. on Aut. Planning and Scheduling (ICAPS)*.

[Löhr et al. 2012] Löhr, J.; Eyerich, P.; Keller, T.; and Nebel, B. 2012. A planning based framework for controlling hybrid systems. In *ICAPS*.

[McDermott 2003] McDermott, D. 2003. Reasoning about Autonomous Processes in an Estimated Regression Planner. In *Proc. 13th Int. Conf. on Aut. Planning and Scheduling (ICAPS)*.

[Penberthy and Weld 1994] Penberthy, S., and Weld, D. 1994. Temporal Planning with Continuous Change. In

*Proc. 12th Nat. Conf. on AI (AAAI)*, 1010–1015. AAAI/MIT Press.

[Penna et al. 2010] Penna, G. D.; Intrigila, B.; Magazzeni, D.; and Mercorio, F. 2010. A pddl+ benchmark problem: The batch chemical plant. In *ICAPS'10*, 222–225.

[Piacentini et al. 2013] Piacentini, C.; Alimisis, V.; Fox, M.; and Long, D. 2013. Combining a temporal planner with an external solver for the power balancing problem in an electricity network. In *ICAPS*.

[Shin and Davis 2005] Shin, J., and Davis, E. 2005. Processes and Continuous Change in a SAT-based Planner. *Art. Int. (AIJ)* 166:194–253.

[Tay 2007] Tay, M. 2007. Model predictive cost control. *Control Engineering* 54(8):IE9.

[Veselý, Rosinov, and Foltin 2010] Veselý, V.; Rosinov, D.; and Foltin, M. 2010. Robust model predictive control design with input constraints. *ISA Transactions* 49(1):114–120.