Enabling Autonomic Properties in Road Transport System

F. O. Jimoh and **T. L. McCluskey** and **L. Chrpa**
Department of Informatics,
School of Computing and Engineering,
University of Huddersfield

P. Gregory
Digital Futures Institute
School of Computing
Teesside University

Abstract

Current autonomic computing systems tend to rely on reactive rather than deliberative reasoning, that is, they use a simpler form of reasoning over sets of defined rules in order to be able to work in real-time. However, technology in areas such as automated planning or constraints processing have been developing rapidly, so that now it may be possible to deploy deliberative reasoning to real-time applications. In this paper, we introduce the problem of self-management of a road traffic network as a temporal planning problem. We design a road traffic model, and use it with domain independent planners to consider the feasibility of introducing it into traffic management applications.

Introduction

Autonomic control systems are an important class of control systems, because of the desirable properties that they offer: the ability to self-manage, self-configure, self-protect and self-optimize. As a consequence, they need to plan and act effectively, in order to follow these behavioral properties. Thus, creating generic technology that enables control systems to automatically reason with knowledge of their controls, in order to generate plans and schedules to manage themselves would be a major breakthrough in the realization of autonomic properties in such systems.

The Automated Planning community has evidenced a need to extend planning algorithms to be able to input increasingly complex domain models which closely approximate real problems(Hoffmann and Edelkamp 2005; Fox and Long 2006; Haslum and Geffner 2001). The existence of such powerful planners increases the motivation for knowledge engineers to incorporate the technique of automated planning into the realization of the properties of autonomic computing. The potential role of AI Planning in autonomic computing was originally made by Srivastrava (Srivastava and Kambhampati 2005), here we look to extending this vision to the more general area of *autonomic systems*. As part of this effort, we have designed a traffic control model to optimize traffic flow and have tested it on domain independent planners.

The long term goal of this work is to exploit a traditional control system architecture, situated in the area of traffic control, and embed it with situational awareness, together with declarative representation of goals, actions and states of its environment, and explore the possibility of using planning engines to support deliberative reasoning within this system. This paper explores the feasibility of this approach in a road traffic domain model where the task is to effectively navigate cars through a road network. We demonstrates some preliminary results in enabling autonomic properties in a road traffic management domain by diverting the flow of regular traffic during an unplanned circumstance - self-optimization.

This model will be embedded into a road traffic virtual environment in our future work. We will evaluate it by comparing its behavior to a traditional system architecture, and assessing the effort and challenges required to embody such symbolic reasoning within a real time environment.

Automated Planning

AI Planning deals with the problem of finding a sequence or partially ordered set of actions whose execution leads from a particular initial state to a state in which a goal condition is satisfied. Actions in plans are ordered in such a way that executability of every action is guaranteed (Fox and Long 2003). Hence, an agent is able to transform the environment from an initial state into a desired goal state (Gupta, Nau, and Regli 1998; Garrido, Onaindia, and Barber 2001). A planning problem thus involves deciding "what" actions to do, and "when" to do them. The "when" part of the problem refers to "scheduling" (Gerevini et al. 2009; Hoffmann and Nebel 2001). In this work we are concerned particularly with temporal domain modeling. For our purpose STRIPS representation does not provide enough expressiveness because we have to consider resources (e.g. road use) and time. A *Planning Domain* consists of a set of predicates and numeric fluents describing the environment and a set of planning operators. A *Planning Operator* is specified via its precondition, effects and duration. A *precondition* of a planning operator is represented by a set of relational and/or

logical expressions. Every expression has assigned its validity range which is one of the following: 'at start' (the expression must be valid at time the operator is executed), 'at end' (the expression must be valid at time the operator finishes its execution) or 'over all' (the expression must be valid at the time interval operator is being executed). *Effects* are represented by a set of assignments and/or set of literals (predicates or their negations). Every assignment or literal has specified when it takes effect which is one of the following: 'at start' or 'at end'. *Duration* of a planning operator is represented by a number which represent the time needed for executing the operator. A *Planning Problem* consists of a set of concrete objects, an initial state and a goal situation. An *initial state* is represented by a set of assignments and predicates. Moreover, timed-literals can be included in the initial state as well which represent in which time-stamp a predicate becomes true or false. A *goal situation* is represented be a set of relational and/or logical expressions. A *plan* is a list of couples ⟨time-stamp,action⟩ (an action is an instance of a planning operator). A plan is a *solution* of a given planning problem if and only if every action is applicable in the given time-stamp (its precondition is fulfilled) and after all action are executed all expressions specified in the goal situation are satisfied.

Planning Domain Definition Language (PDDL) PDDL(McDermott D. et al. 1998) was developed with the aim of being a neutral specification of planning models and later improved upon and used at every International Planning Competition. It is widely used in the planning community due to it compatibility with most planning engine with neutral meaning that it does not favor any particular planning system. This has made it to be accepted as a standard for the representation and exchange of domain model within the planning community.

Autonomic Computing Paradigm

Human breathing, heartbeat, temperature, immune system, repair mechanisms are all to a great extent controlled by our body without our conscious management. For instance, when we are anxious, frightened, ill or injured, all our bodily functions evolves to react appropriately. The autonomic nervous system has all the organs of the body connected to the body central nervous system which takes decisions in order to optimize the effective functioning of other organs in the body. Autonomic Computing is motivated by a metaphorical parallel between computing systems and the human autonomic nervous system, and aims at realizing computing systems and applications capable of managing themselves with minimum human intervention.

The AC initiative was introduced in 2001 with the goal of developing self-managing systems. IBM defines four main properties for an AC: self-configuring, self-healing, self-optimizing and self protecting (IBM 2003; 2005; 2006; Ganek and Corbi 2003). AC is meant to improve the wide-range of usability and manageability of systems which will benefit all road transport system at long run. AC systems should have the ability to learn process pattern from the past and adopt, discard or generate new plans to improve process control. The ability to identify the task is the most important aspect of any AC element, this enables an AC to decide on the appropriate action when healing, optimizing, configuring or protecting itself.(Hariri et al. 2006).

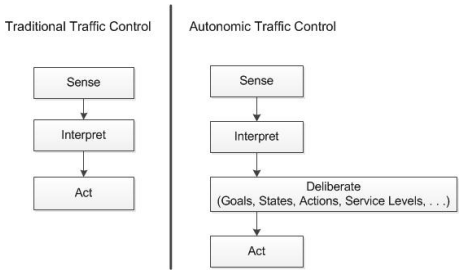

Figure 1: Illustration of AC architecture in road transport system.

Strategy of an Autonomic System

We use the term "autonomic system" rather than autonomic computing to emphasize the idea that we are dealing with a heterogeneous system containing hardware and software. Sensors and effectors are the main component of this type of autonomic system architecture (Ganek and Corbi 2003). AC needs sensors to sense the environment and executes actions through effectors. In most cases, a control loop is created: the system processes information retrieved from the sensors in order to be aware of its effect and its environment; it takes necessary decisions using its existing knowledge from its domain, generates effective plans and executes those plans using effectors. Autonomic systems typically execute a cycle of monitoring, analyzing, planning and execution (Lightstone 2007).

Most system architecture elements are self-managed by monitoring and analyzing behaviors and using the response to plan and executes new actions that takes or keeps the system in desired state. The planning processes inherent in autonomic systems could be achieved by AI planning or through the use of comprehensive learning and adaptive algorithms. Overall self-management is a means for a system to do self-assessment, protection, healing, optimization, maintenance and other overlapping terms (IBM 2006). Any system that is meant to satisfy these above objectives will need to have the following attributes:

- Self-awareness of both internal and external features, processes resources, and constrains
- Self-monitoring it exiting state and processes and
- Self-adjustment and control of it-self to the desirable/required state

- Heterogeneity across numerous hardware and software architectures

Requirements for Self-Management in Road Traffic Support System

In this paper we consider the implementation of autonomic properties of specific areas of Road Traffic Management, in line with the EU COST ARTS initiative [1]. Most existing traffic signal control methods are based on feed-back algorithms. They make use of traffic-demand data varying from several years of storage to a couple of minutes prior to usage. Current traffic control systems often operate on the basis of adaptive green phases and flexible co-ordination in (sub) networks based on measured traffic conditions (an example is SCOOT, UTOPIA-spot)(Roozemond 2001; De Oliveira and Bazzan 2009).

However, these approaches are still not optimal during unforseen situations such as road incident, road reconstruction, car breakdown and when traffic demand changes rapidly within a short time interval. In this situation, there arises the need for self-managing systems that can take pro-active decision using some set of well developed meta-rules to alter the behavior of the state of any road traffic situation based on achievable goals. This pro-active nature of traffic control is the heart on which our research work is build-upon.

Ideally, the following abilities would be incorporated into a road traffic model within the process of implementing this research work.

- Ability to retrieve information on the current state of traffic based on information from road sensors
- Ability to detect current traffic problems by evaluating current traffic trends with the use of traffic control rules on an existing ideal model of the surrounds
- Ability to estimate the next optimal cycle mathematically
- Ability to take decision and operate the entire traffic control signals based on the generated optimal plan

Resources and Constraints: In every system design, there are resources available for execution. These resources come with associated limitations (constraints) which must be identified and optimized for effective implementation of the system. A list of resources and constraints that are considered for a potential model are discussed in this section.

1. Resources from the component of road traffic system which includes: road intersections, traffic lights, variable message signs, ramp metering and state switching at the intersection.
2. Resources from road sensors which includes: status of roads and intersections, capacity of the road and length of queues.

[1] www.cost-arts.org

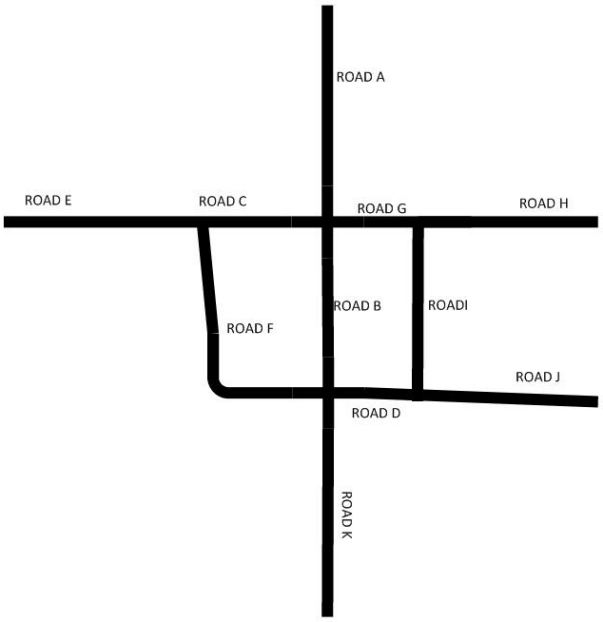

Figure 2: Layout of the road route showing the relationship between junctions

3. Constraints in this model could be further broken down into two parts:

   Resource Constraint The execution of activity needs certain resources which must be optimized at every resulting state from/during and after every action within the domain.

   Temporal Constraint Every actions have their own duration which must be valid within any action in the execution frame work. For instance, the duration of allowed action for vehicle movement is according to the road's queue length in relation to the road capacity of neighboring routes.

The Design of a Road Traffic Model(RTM)

This section describes the design of a Road Traffic Model (RTM) in order to explore the role of automated planning in enabling some of the requirements described in the section above. The model was tested with different domain independent planners and the resulting output is evaluated.

RTM Design The model architecture is divided into four part:

- The properties of the road represented by road layout segment.
- Controlled intersections are represented by the road-junctions relationship within the road network
- The road users properties represented by the road capacity and queue length

- Route properties that span several adjoining road segments.

RTM Specification

A Road Network can be represented by directed graph, where edges stand for roads and vertices stand for either junctions or entry or exit points. Entry points are points where cars enter the network, while exit points are point where cars exit the network. In junctions we must be aware of conflicting ways on which cars cannot go simultaneously. Every road has its own length and capacity (i.e. a maximum number of cars it can serve). The network model can be enhanced by considering time intervals when a road is closed for maintenance.

Definition 1 *Let (V, E) be a directed graph such that $\forall v \in V : (indeg(v) = 0 \rightarrow outdeg(v) = 1) \wedge (outdeg(v) = 0 \rightarrow indeg(v) = 1)$. Edges in E represent one-way roads. A vertex $v \in V$ represents:*

- *entry point if $indeg(v) = 0$*
- *exit point if $outdeg(v) = 0$*
- *junction otherwise*

Let $\mathcal{M} = \{M_1, \ldots, M_n\}$ be a set of sets of conflicting ways such that M_i is defined as a set of triples in form (e_{i_x}, v_i, e_{i_y}) such that a junction v_i is a tail of e_{i_x} and a head of e_{i_y}.
Let $C : E \rightarrow \Re$ be a function representing road capacity and $l : E \rightarrow \Re$ be a function representing road length.
Let T be a mapping from edges (E) to sets of time intervals representing road blockage. Then $\mathcal{N} = \langle V, E, \mathcal{M}, C, l, T \rangle$ is a road network.

A Road Planning Problem addresses the problem of efficient navigation of cars through a given Road Network from entry points to exit points. Initially, it is given the number of cars in each entry point and frequency of their releasing. This can be represented by a set time-stamps in which the entry points are 'opened'. The goal situation is determined by numbers of cars in exit points. Hence, we consider two actions (planning operators) which are defined as follows:

RELEASE-CARS — In a given time-stamp, the action releases a pre-defined number cars if the capacity of the road adjacent to the corresponding entry point would not be exceeded. After executing the action the cars are present on the road adjacent to the corresponding entry point.

FLOW — The action navigates cars through a junction if the capacity of the road on which cars are being navigated would not be exceeded. After executing the action the cars are 'relocated' from one road (leading towards the junction) to another (leading from the junction). Note that the actions cannot be executed simultaneously for the same junction.

RTM PDDL Code

Modeling an RTM problem in PDDL holds several challenges. The most critical of these is that in the real-world drivers have true agency and are therefore not strictly under our control. The flow of traffic is therefore best modeled as a complex hybrid process. However, this process is difficult to define and would lead to a problem too challenging for contemporary planners. Therefore we have taken the approach of modeling the processes as closely as possible using PDDL 2.1. In all considered models, the goal of the planning problem is to route a certain number of vehicles through the network to the exit points, whilst minimizing the makespan of the plan.

The modeling of a simple RTM problem requires a method of periodically introducing vehicles at the entry points, and also of restricting how the vehicles can move around the network. We look at these two issues in turn, starting with periodic entry into the network. Each road has a *use* fluent, representing the number of cars currently on the road. One approach that we explored was to use timed initial literals in order to define particular times when these *use* fluents are refreshed.

Therefore, several statements of the form:

```
(at  0 (= (use a) 10))
(at 10 (= (use a) 10))
       ....
```

could be added to the initial state. We believed that this was an attractive approach as it ensures the *flow* of traffic into the system in a staggered way, and this models the real-world problem better than simply adding all vehicles at time 0. Unfortunately, the use of fluent assignments in timed initial literals is not supported in PDDL (or at least by the planners we experimented with), and so another approach was necessary.

The final model is shown in the *RELEASE-CARS* action in Figure 3. We use the unary predicate *ready* to denote that an entry point is ready to release cars. For the duration of a *RELEASE-CARS* action, this is deleted to ensure that only a single instance of the ground action can be executed at any one time. At the end of the action, the entry point is set to ready again. The duration is set to the amount of time successive waves of vehicles can enter the network. A weakness of this approach is that a planner is free to schedule *RELEASE-CARS* actions as far apart as it chooses. In this respect the domain does not adequately model the fact that the vehicles should enter the network at regular intervals, regardless of when the planner chooses to release them. We do not know of a remedy to this situation other than the use of timed initial literals; and as previously discussed, this proves impractical with current planners. Pragmatically this problem is reduced by the fact that for the makespan to be optimized it is sensible to release the cars as early as possible.

The *FLOW* action (again, shown in Figure 3) is the action that deals with moving cars from one road to another. This action moves a single car from one road to

```
(define (domain transport)
  (:requirements :typing :durative-actions :fluents)

  (:types road)

  (:predicates
    (ready ?road - road)
    (operational ?road - road)
    (connected ?road1 - road ?road2 - road)
  )

  (:functions
    (flowrate ?road - road)
    (capacity ?road - road)
    (use ?road - road)
  )

  (:durative-action RELEASE-CARS
   :parameters (?r - road)
   :duration (= ?duration 10)
   :condition (and (at start (ready ?r)) (at end (< (use ?r) (capacity ?r))))
   :effect (and (at end (increase (use ?r) 5))
                (at start (not (ready ?r))) (at end (ready ?r)))
  )

  (:durative-action FLOW
   :parameters (?road1 ?road2 - road)
   :duration (= ?duration (/ 1 (flowrate ?road1)))
   :condition
    (and
      (at start (connected ?road1 ?road2))
      (at start (>= (use ?road1) 1))
      (at start (< (use ?road2) (capacity ?road2)))
      (at start (operational ?road2))
      (at start (operational ?road1))
    )
   :effect
    (and
      (at start (not (operational ?road1)))
      (at end (operational ?road1))
      (at start (decrease (use ?road1) 1))
      (at end (increase (use ?road2) 1))
    )
  )
)
```

Figure 3: A domain model for the Road Traffic Management problem.

```
(define (problem problem1)
 (:domain transport)
 (:objects
   a b c d e f g h i j k - road
 )
 (:init
   (connected a g)
   (connected b g)
   (connected c g)
   (connected d b)
   (connected d k)
   (connected e c)
   (connected e f)
   (connected f d)
   (connected f k)
   (connected g h)
   (connected g i)
   (connected i j)
   (connected i d)
   (connected g i)
   (connected g i)
   (= (flowrate a) 5)
       ...
   (= (flowrate k) 4)
   (= (capacity a) 100)
       ...
   (= (capacity k) 100)
   (= (use a) 0)
       ...
   (= (use k) 0)
   (operational a)
       ...
   (operational k)

   (ready a)
   (ready e)
 )
 (:goal
   (and
    (>= (use j) 0)
    (>= (use k) 0)
    (>= (use h) 5)
   )
 )
 (:metric minimize (total-time))
)
```

Figure 4: An example problem file modeling the transport network from Figure 2. In this problem, five cars must be routed to exit point h

another providing they are connected and that there are sufficient cars and capacity to support the operation. In the example that we are using, the road-network is a one-way system in which each junction has a maximum of two different choices for the traffic lights. Thus, each flow action prevents the first road from being used simultaneously to feed more than one road.

RTM Analysis The model shows a road layout of five junctions and eleven connected roads. A declarative representation of the road layout and the traffic switching rules and policies is modeled in PDDL. The initial road queue length of each roads and the desire queue length are given as the initial and goal state respectively. The maximum number of vehicles and the traffic flow rate limitations of each roads are part of the resource constraints in the problem file. Vehicle are allow to enter and leave the system at appropriate intervals.

RTM Result

Figure 5 shows the plan generated by LPG for the instance defined in Figure 4. This is a simple problem where 5 cars are routed from one entry point to a single exit point. Table 1 shows the outcome of solving five different problems of varying difficulty. Three planners were used in this analysis: LPG (Gerevini, Saetti, and Serina 2006), Crikey (Coles et al. 2009) and Optic (Benton, Coles, and Coles 2012). In the examples, Optic performs the best on the example instances that we have provided. LPG takes less time to generate plans. Within an autonomic framework, planning will occur in a control-loop and so there will be a practical trade-off between quality and speed, depending on size of problem and time available.

Both Crikey and Optic are capable of solving problems with required concurrency and durational inequalities and Optic supports time-dependent costs. We envision that more realistic models of the RTM problem will require these capabilities.

Discussion

In reality, it is not possible to directly control the vehicles in a road traffic network. We do not expect that the plan generated is to be executed in the traditional sense. Therefore, the results of the planning process should be seen as a guide (or a kind of heuristic) saying what traffic is going to do under idealized conditions.

```
; States evaluated: 23
; Cost: 13.510
; Time 0.10
0.000: (release-cars a)  [10.000]
10.001: (flow a g)  [0.200]
10.202: (flow g h)  [0.500]
10.703: (flow a g)  [0.200]
10.904: (flow g h)  [0.500]
11.405: (flow a g)  [0.200]
11.606: (flow g h)  [0.500]
12.107: (flow a g)  [0.200]
12.308: (flow g h)  [0.500]
12.809: (flow a g)  [0.200]
13.010: (flow g h)  [0.500]
```

Figure 5: The plan generated by the Optic planner for the problem instance defined in Figure 4

CARS	LPG	CRIKEY3	OPTIC
5	13	13	13
10	40	26	13
15	37	38	24
30	55	63	38
45	83	89	54

Table 1: Quality of plans generated by several planners for instances with varying number of cars in the network. The underlying network in all examples is that defined in Figure 2. Optic tends to produce better plans in the domain.

There are situations in which a planner is useful in answering questions for a traffic support system. One particular example is during road closures, either expected or unexpected. During road maintenance traffic flow rates may change due to slower moving traffic. Whenever there is a road block due to car break down, road maintenance or an accident, the traffic support system automatically generate plans which ca divert traffic from such routes. Road closures are modeled with the use of timed initial literals. Modeling changes in traffic flow at specified times is more problematic, though using dummy actions it is possible within PDDL 2.1.

Conclusion

In this work we have demonstrated some preliminary results in enabling autonomic properties in a road traffic management domain by diverting the flow of regular traffic during an unplanned circumstance - self-optimization. In a real world scenario where road queues and road capacity lengths are uploaded in real time from road sensors, we hope to possibly sense and divert road traffic with little or no human intervention. Other factors such as a change in weather conditions, priority vehicles and road construction interruption can also be taken into account while generating plans in order to facilitate pro-active decisions.

In this paper we designed a road traffic domain model which allows to navigate cars throughout the road network. We plan to embed our model into a road traffic virtual environment in our future work. We plan to evaluate it by comparing its behavior to a traditional system architecture, and assessing the effort and challenges required to embody the model within a real time environment. Other aspects of improvement would include: incorporating the actual road policies; increase interaction with road users; build a knowledge base from generated (optimal) plans and consider various speed limits.

References

Benton, J.; Coles, A. J.; and Coles, A. 2012. Temporal planning with preferences and time-dependent continuous costs. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *ICAPS*. AAAI.

Coles, A.; Fox, M.; Halsey, K.; Long, D.; and Smith, A. 2009. Managing concurrency in temporal planning using planner-scheduler interaction. *Artif. Intell.* 173(1):1–44.

De Oliveira, D., and Bazzan, A. L. C. 2009. *Multiagent Learning on Traffic Lights Control: Effects of Using Shared Information*. 307–322.

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

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

Ganek, A., and Corbi, T. 2003. The dawning of the autonomic computing era. *IBM Systems Journal* 42(1):5–5.

Garrido, A.; Onaindia, E.; and Barber, F. 2001. A temporal planning system for time-optimal planning. In *Progress in AI*, volume 2258 of *LNCS*.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Art. Int. (AIJ)* 173(5-6):619–668.

Gerevini, A.; Saetti, A.; and Serina, I. 2006. An approach to temporal planning and scheduling in domains with predictable exogenous events. *J. of AI Research* 25:187–231.

Gupta, S. K.; Nau, D. S.; and Regli, W. C. 1998. IMACS: A case study in real-world planning. *IEEE Expert and Intelligent Systems* 13(3):49–60.

Hariri, S.; Khargharia, B.; Chen, H.; Yang, J.; Zhang, Y.; Parashar, M.; and Liu, H. 2006. The autonomic computing paradigm. *Cluster Computing* 9(1):5–17. 1386-7857.

Haslum, P., and Geffner, H. 2001. Heuristic planning with time and resources. In *Proc. 6th European Conference on Planning (ECP'01)*, 121–132.

Hoffmann, J., and Edelkamp, S. 2005. The deterministic part of ipc-4: An overview. *J. Art. Int. Res. (JAIR)* 24:519–579.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *J. Art. Int. Res. (JAIR)* 14:253–302.

IBM. 2003. An architectural blueprint for autonomic computing,.

IBM. 2005. Problem determination using self-managing autonomic technology.

IBM. 2006. An architectural blueprint for autonomic computing.

Lightstone, S. 2007. Seven software engineering principles for autonomic computing development. *ISSE* 3(1):71–74.

McDermott D. et al. 1998. PDDL–the planning domain definition language. Technical report, Available at: `www.cs.yale.edu/homes/dvm`.

Roozemond, D. A. 2001. Using intelligent agents for pro-active, real-time urban intersection control. *European Journal of Operational Research* 131(2):293–301.

Srivastava, B., and Kambhampati, S. 2005. The case for automated planning in autonomic computing. In *Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on*, 331–332.