# ARTS-COST Roadmap

ARTS COST members

October 3, 2015

# Contents

# Foundations of Autonomic Systems

Chapter 1

# An Artificial Intelligence Perspective

Daniel Borrajo, Arturo González-Ferrer and Lee McCluskey
Universidad Carlos III de Madrid & University of Huddersfield

## 1.1 Introduction

In this chapter we briefly review some of the fundamental ideas in Artificial Intelligence (AI) and their relation to the transportation area, describing some recent developments that potentially have great benefit to ARTS ("State of the Art"), and which have not as yet been fully exploited by the Transport Studies community. Under "Challenges" we record the developments that need to be made to be able to apply these new innovations, and under "Actions for the Community" the steps needs (research projects) that would result in the overcoming of these challenges.

## 1.2 State of the art

The study of Artificial Intelligence (AI) can be carried out considering only a single agent. Here communications and interactions between agents are not considered - the single agent is to perform intelligent functions in a "centralised" fashion, hence we can refer to this as centralised AI, to distinguish it from areas such as multi-agent systems, or more generally "distributed AI" [126]. Of course, many of the ideas developed here are also relevant when we generalise to distributed AI. In this section we will describe the state of the art in Artificial Intelligence (AI) techniques that are useful for Autonomic Computing (AC) from a centralised perspective. We will aim at answering what the major innovations from this area are that ARTS can use.

### 1.2.1 Foundational Concepts in AI

AI techniques have been widely used for all kinds of applications, and, in particular, for Road Transport Systems (RTS) tasks. AI includes many different subfields, so we will summarize relevant work on AI applied to RTS, classified by some of the subareas of AI. In particular, we will summarize what type of techniques can be reused for performing AC, with special emphasis on RTS domains.

#### Knowledge representation

The first step for developing AI-based software consists of modelling the domain knowledge that is relevant for solving problems. The process of acquiring, refining and integrating domain knowledge with AI processes is called Knowledge Engineering. It is a key aspect for developing some applications, specially the ones that take control of some key aspects, given that humans prefer languages to describe tasks at higher levels than those used by most programming languages. We will discuss in subsequent sections apects of modelling and engineering of knowledge. Here we consider briefly the idea of the representation of knowledge within a precise, symbolic language,

There have been many proposals for general knowledge representation formalisms. Among the most used ones, we can cite: logic (mostly propositional and predicate logics) as in Prolog, theorem provers or automated planning; frames, which have derived on object-oriented representations; semantic networks; and ontologies, which incorporate ideas of all of the previous, as frames, relations among classes/objects (as in semantic networks), or axioms, represented using some types of logic. In order a system to have some type of AC behaviour, it should have a model of the domain knowledge, as well as some internal model about itself. So, all the previously mentioned formalisms can be used to create models of the external and internal worlds, in order to be able to reason about both. Once one has defined a model of both worlds, then one can apply a wide range of AI techniques to reason about the external and internal models. We will see next several of them, as rule-based systems, heuristic search, automated planning, reasoning under uncertainty, evolutionary computation, or machine learning.

#### Heuristic Search

Search techniques provide one of the basic problem solving capabilities to any AI software. Search deals with the task of finding solutions to problems specified in a given problem space. A problem space is an abstraction of human reasoning tasks. A problem space is composed of a set of states (describing the possible configurations of the world), and a set of operators (describing how to transit from one state into another). For instance, in a route planning task, the set of states will be composed of all the places (e.g. road junctions) at which a vehicle can be, and the set of operators specifies each one of the possible ways of going from one state to another (e.g. move the vehicle from one junction to another).

Given a problem space, a problem is specified as: an initial state (e.g. the initial position of the vehicle, and the traffic conditions at each junction), and less commonly a set of initial states; and a final state (the junction where the vehicle should go), or goal description (the vehicle is in a parking, no matter which parking it is). Search is the process by which given a problem in a problem space, it returns the set of operators (also called solution) that have to be applied to transit from the initial state into the final state (or a state fulfilling the goal description). Sometimes, what we are looking for is the end state rather than the set of operators (as in constraint satisfaction problems). Since problem spaces can be represented as graphs, solving problems can be seen as searching for paths in graphs.

In case there is no information available about a problem space, we would use uninformed search. In case we have some information, we could model it as a heuristic function, and we could use heuristic search. Heuristic search has been widely used, mainly in the context of shortest path computations. However, other tasks related to traffic can be defined as search problems. For instance, urban transport planning tasks, as deciding where to place bus stops of a given bus line in a city, such that it minimizes the rupture it creates in the remaining traffic.

Heuristic search has been used quite extensively for route planning [132]; i.e. computing shortest paths (according to different measures) in road maps, as for instance in most car navigation systems, public transport information systems, or for autonomous vehicles (as the ones participating in the Urban Challenge) [55]. They are being used also for tasks as multi-modal transportation [61], multi-agent path finding [165], or ridesharing [35].

### 1.2.2 Rule-Based Systems

Rule-Based Systems (RBS) [71] emerged back in the 1980s as a means to code the know-how of human experts. The main idea is to incorporate human knowledge as a set of if-then rules, where a particular state triggers an action. Several rules can be combined in order to make a decision, and an explanation to each action proposed can be easily traced back by exploring the applied rules. Existing knowledge can be refined, typically after selection and execution of a particular rule, and new facts can be added to the knowledge base at any time in order to increase the performance of the system.

Next we mention some applications of RBS to RTS:

- Sets of rules for traffic control. A rule-based approach is presented in [56] where multi-dimensional learning programs optimize both the hierarchy of the rules and the parameters embedded in individual rules. Different measures of effectiveness can be selected as the criterion for optimization. In [42], authors describe the development and evaluation of a fully distributed, real-time, traffic-responsive model named SPPORT (Signal Priority Procedure for Optimization in Real-Time). The model explicitly considers the interference in the general traffic caused by transit vehicles stopping in the right of way to board and discharge passengers. Second, when considering priority passage for transit vehicles, the potential effects that such preferential treatment might have on other traffic is explicitly quantified. The rule-based signal optimization procedure provided delay reductions for most of the traffic conditions examined.

- Image Analysis and rule-based reasoning for traffic control: In [38] its authors present a flexible and robust approach for detecting vehicles (spatio-temporal analysis during daytime, and by morphological analysis of headlights at night) in urban traffic scenes by means of rule-based reasoning on visual data. They use a rule-based system, working on traffic data (as vehicles and their attributes - area, pattern, direction, and others) and tuned for urban traffic conditions.

### 1.2.3 Automated Planning and Scheduling

Automated Planning (AP) deals with the task of generating a sequence of actions for achieving a set of goals from an initial state [68]. Scheduling deals with the task of assigning start and end times to activities (actions), once we know the actions ordering constraints. So, the input of scheduling tasks can come from the output of a planner, or the activities to be scheduled can come directly from humans or other software. Commonly, automated planners and schedulers use heuristic search as the internal problem solving technique. One of the main differences between the work on heuristic search and AP is that problem spaces in AP are represented declaratively in separate files, while in most heuristic search applications the problem spaces are defined implicitly within the program itself.

Most planners take as input a domain (problem space) and problem descriptions, usually in two files, that comply with the standard language PDDL (Planning Domain Description Language[1]). The domain description includes information on the language to represent states (in the form of predicates), and a set of actions. Each action is specified by a set of conditions to execute the action and a set of effects that are expected after executing the action. The problem description includes information on the initial state (e.g. the current state of the traffic network), the goals (that the human users set), and, optionally, a metric to consider for optimization. The current PDDL standard includes a very rich language. It includes the ability of defining different metrics, numeric fluents, universal and existential quantifiers, or even probabilistic information (in its probabilistic version, PPDDL).

A special type of planners that have shown their usefulness in many application fields follow the Hierarchical Task Network (HTN) planning paradigm [54, 104]. In this paradigm, high-level actions (compound tasks) are refined into lower level tasks or, in the end of the hierarchy, into primitive actions. Methods define how to perform the decomposition of tasks into subtasks, and include

[1]See http://ipc.icaps-conference.org for PDDL resources as well as International Planning Competition or http://zeus.ing.unibs.it/ipc-5/bnf.pdf for the description of PDDL3.0

preconditions for their selection. The goal of the planning problem is usually a compound task that is refined by the planning algorithm into a set of primitive actions, which lay at the leaves of the task network. The intuitive idea behind HTNs is that this task network decomposition can improve the search efficiency by correctly adding expert knowledge into the task network structure.

There are several planning tasks. Among others, we can mention: STRIPS planning is the simplest form of planning where actions only use simple preconditions and effects; cost-based planning allows users to define actions costs and reason about them when planning; optimal planning takes into account those actions costs to compute optimal plans according to some cost metric; temporal planning reasons about actions that take time to execute and can be executed concurrently, so it can replace the need of scheduling after planning; planning under uncertainty generates plans in environments where actions execution or observations about the states include some uncertainty; or planning with continuous variables.

As an example of a traffic domain, suppose we want to define an scenario to handle incidents in highways. We would need to define a domain that would model how this kind of tasks are solved and a set of problems will be dynamically created and solved each time a new incident emerges. In the domain description, we would have to define first a set of types that are relevant in this domain, such as `road-segment` or `signal` (variable speed limit signal). Then, we would define a set of predicates to represent the states. For instance, we could define `next(x,y)` (a road-segment `x` is followed by another one `y`), or `signals(x,y)` (a variable-speed-limit signal `x` specifies the speed on road-segment `y`). Also, we can define functions, which are equivalent to predicates, but have a numeric value instead of a boolean value. As an example, we could have the `max-capacity(x)` (whose value will be the maximum capacity of road segment `x`), `capacity(x)` (whose value will be the current capacity of road segment `x`), or `limit(x)` (whose value will be the speed limit that appears in signal `x`). Figure 1.1 shows an example of such domain definition in PDDL.

Finally, we could define the actions that can be executed in this domain in order to solve incidents. We could define an action that sets the speed limit in a road segment, as shown in Figure 1.2.

The action specifies that if a road segment `?r1`[2] is jammed (its current capacity is equal to its max-capacity), another road segment `?r` is the one before, `?r` is not yet jammed, and there is a variable speed limit signal `?s` at `?r`, then executing this action would set the speed limit at `?r` to 50.

In the problem file, we would have to specify first the objects that appear in a given task. Suppose that we have now a part of a highway with three connected road segments R100, R101, and R102. And, also assume there is a signal in R101. We would have to define next the initial (current) state of the highway, and the goals we want to achieve (such as not having any jammed road segment. Finally, we could also specify a metric to take into account when planning, such as minimize capacity of road segments, travel time, or emissions. Figure 1.3

[2]Everything preceeded by a ? is a variable.

```
(define (domain incidents-control)
  (:requirements :typing :fluents)
  (:types road-segment signal)

  (:predicates (next ?r - road-segment ?r1 - road-segment)
               (signals ?s - signal ?r - road-segment) ...)

  (:functions (max-capacity ?r - road-segment)
              (capacity ?r - road-segment)
              (limit ?s - signal) ...)

  (:action set-limit ...)
  (:action open-ramp ...)
  ...
)
```

Figure 1.1: Example of domain file in PDDL with the name of the domain, the type of planning model we are using (we require types, and numeric fluents), some types, predicates, functions and actions.

shows an example of such problem specification in PDDL. Here, the predicate `unjammed(x)` means the road segment capacity is under its max-capacity, and `total-travel-time` would be a function that accounts for the sum of all travel times of all vehicles. Both would also have to be defined in the domain file.

Some examples of applied AP works on the transport and/or AC related domains are:

- Traffic lights control: scheduling of traffic lights, based on vision [143]
- Traffic incidents management: in [136] the authors describe different approaches to generate planning domain models related to accident management
- Applications in logistics: TIMIPlan [61] generates long-distance multi-modal transportation plans, as well as short-distance (within a city) plans [62].
- Applications in tourism: SAMAP [26] generated tourism plans according to user and similar users profiles. It was a prototype. More recently, OnDroad [27] recovers data from a real tourism portal[3] and generates plans from the most recommended places.
- Traffic management: in [92] a planning method for traffic management and control is developed based on HTN planning technology. In [8] it is

[3]www.minube.co.uk for UK or www.minube.net for USA.

```
(:action set-limit
  :parameters (?r - road-segment ?r1 - road-segment ?s - signal)
  :precondition (and (= (capacity ?r1) (max-capacity ?r1))
                     (next ?r ?r1)
                     (< (capacity ?r) (max-capacity ?r))
                     (signals ?s ?r))
  :effect (and (assign (limit ?s) 50)))
```

Figure 1.2: Example of an action definition.

proposed to transform quantitative results related to geometric properties of recorded camera images observing road traffic into textual descriptions of vehicle maneuvers and their context. The authors propose that this conceptual knowledge, expressed as a type of graph trees, could be re-cast as Hierarchical Task Networks (HTNs) in order to re-use the knowledge about the behavior of vehicles in road traffic scenes for planning purposes. In [17] an application (TMGR) to assist users in activities involving travel management is designed as an adaptive autonomous system that should ensure trustworthiness of its behaviour. The system follows an architecture that could be used in other domains, using inductive (evolutionary), deductive (planner) and abductive (reward manager) reasoning modules governed by a trustworthiness enforcer. The deductive reasoner is implemented as an extension of the SHOP2 HTN planner [104].

- Planning for autonomic computing in general: In [146] they present a general discussion on the application of automated planning to AC, focusing on four aspects of self-management and the role that planning may have: self-configuration, self-healing, self-optimizing and self-protecting.

- Planning under uncertainty: a traffic control domain was used in the International Planning Competition in the Probabilistic track (IPC'11)[4] and it has been used again in the competition in 2014.[5]

### 1.2.4 Reasoning under Uncertainty

Most transport tasks include some uncertainty, since the world is mostly non-deterministic. There are always incidents, people do not always behave as expected, or uncertain weather can complicate traffic. Thus, in order to provide autonomic behavior, management systems should be able to cope with uncertainty. This topic has been extensively studied in AI under various forms, that

[4]http://users.cecs.anu.edu.au/ ssanner/IPPC$_2$011/
[5]http://users.cecs.anu.edu.au/ ssanner/IPPC$_2$014/*index.html*

```
(define (problem incident-2-4-2014)
        (:domain ramp-metering)
        (:objects R100 R101 R102 - road-segment
                  S1 - signal)
        (:init (next R100 R101)
               (next R101 R102)
               (= (max-capacity R100) 30)
               (= (max-capacity R101) 30)
               (= (max-capacity R102) 30)
               (= (capacity R100) 20)
               (= (capacity R101) 22)
               (= (capacity R102) 30)
               (signals S1 R101)
               (= (limit S1) 80))

        (:goal (and (unjammed R100)
                    (unjammed R101)
                    (unjammed R102)))

        (:metric minimize (total-travel-time)))
```

Figure 1.3: Example of problem file in PDDL for a hypothetical highway.

range from probabilistic computation, Bayesian reasoning, Markov models to fuzzy reasoning.

Bayesian networks are a way of compactly and graphically representing the causal (in)dependence among a set of variables [111]. They consist of a graph where nodes represent variables, and edges represent the dependence of one variable on another one. Each node has a Conditional Probability Table (CPT) that summarizes the conditional probabilities of the values that a variable $V$ can take given the values of the parent variables (the ones that have an edge that goes to $V$). If a node does not have parent variables, it will have a table with its "a priori" probabilities.

As an example in the highway traffic domain, we could have a variable for each road segment $R_i$. Each of these segments variables $R_i$ influences the values of the following segment $R_{i+1}$, so there will be an edge from each $R_i$ to its corresponding $R_{i+1}$. For simplicity, let us assume that each of these variables can have three values as: jammed, dense, and fluid. We can also define a variable for the weather condition at the highway, $W$, that will have an edge to all segment variables, and will have three values again: rain, fog, and normal. Once we define all the variables, their values and the edges that connect them, we would have to define all CPTs. Those can be manually defined or using machine learning.

Markov models are stochastic models that fulfill the Markov property. The Markov property can be summarized as: the prediction of the future states of a system only depends on its current state, so it does not depend on its history. Markov models can be divided on models that cannot be controlled (such as Markov chains or Hidden Markov models) and those that can be controlled by selecting actions (such as Markov Decision Processes, MDP, or Partially Observable MDP, POMDP). Sometimes it is useful to model them as Bayesian Networks.

A variation of Rule-Based Systems are Fuzzy logic RBS, that have shown to be a good mathematical approach for dealing with subjectivity, uncertainty, and imprecision. Fuzzy if-then rules are rules whose conditions, effects or both are fuzzy rather than crisp [178]. Fuzzy logic [182] allows practitioners to go beyond classical logic, where propositions can only have a value of either true or false. Instead, fuzzy logic usually expresses truth values in the range [0-1]. It tries to represent the subjective point of view of an observer on the truth value of a concept. A typical example of fuzzy sets is to qualify a continuous variable like temperature as cold, warm or hot, where $n$ observers could define their particular view on the truth value for each of the three qualifications. This allows us to have *linguistic variables* to express rules (e.g. "if temperature is *very hot* turn on the red alert") that have an underlying associated mathematical model, i.e. the fuzzy sets definition, where individuals -evaluated temperature values- have a *grade of membership* to each fuzzy set.

We will mention here some approaches that explicitly deal with uncertainty in transport tasks.

- Fuzzy logic: ranges from early work on traffic lights control with fuzzy techniques [107] to more modern approaches [105, 169]. One of the first examples of using fuzzy rules applied to traffic control was provided in [156], by modeling a simple two-route choice problem in order to select the best one using rules as the following one:

  "IF perceived travel time along route A *is much less* than perceived travel time along route B, THEN fuzzy preference index for A *is very stronger than* B".

  In [33], they describe an approach to use local controllers whose signal timing parameters are configured looking at local traffic conditions and the parameters of other controllers at adjacent intersections. The controllers use a set of fuzzy decision rules to adjust the standard signal timing parameters: cycle time, phase split, and offset.

- Fuzzy rule-based systems that are configured through neural networks: An automated driver prototype has been developed using a self-organising fuzzy rule-based system in [110] to model and subsequently emulate human driving expertise. Authors in [119] described a novel approach to traffic flow analysis and modelling using a specific class of self-organizing fuzzy rule-based system. Experimental results have demonstrated the chief ability of their approach to extract a valid set of fuzzy rules from the training data as well as to generalize appropriately given new input data.

- Bayesian reasoning: Bayesian networks have been used for predicting traffic flow in [150, 25].
- Markov models: MDPs or POMDPs can be used to select actions to control traffic subsystems, as traffic lights control [181].
- There have also been some works in stochastic environments by using parametric optimization [88].

### 1.2.5 Domain Modelling and Knowledge Engineering

*Domain Modelling* is a concept used, perhaps with a variety of meanings, in the wide area of computer science. A domain model is often described as an abstract conceptual description of some application, and suggests it is created as an aid to the software development process where it is formed as a part of the requirements analysis - it specifies objects, actors, roles, attributes etc, independent of the bias of a software implementation, and is often represented imprecisely using diagrams, such as in the UML. The domain model is then meant for "human consumption" - for the benefit for analysts and developers interested in creating software with the domain (application area) being modelled.

The meaning of domain model for representing knowledge is much more specific. It is an abstract conceptual description of some application area, as in general software engineering, but it is encoded for a different purpose - for machine manipulation. Hence it must be encoded *with enough precision so that we can input it to a program (agent) which can use it to reason about the domain within which it acts.* The domain model is created for the purposes of a program or software agent, not for human delevopers. We want to have domain models that enable programs to reason, learn, plan, write software, or perform autonomic functions on a traffic application. In summary, the program uses the domain model to understand the domain.

Language notations for Domain Models vary, but often contain constructs such as Objects, Object Classes (hierarchy), Functions, Properties and Relations of Objects, Simple Factual Statements, and complex statements combining the above with logical quantification, logical operators or logical connectives (AND, OR, NOT, . . . ).

**Example 1:** As a deceptively simple example, let us assume that we wanted to create an agent that could not only count, but "understood" numbers enough to proves its counting was correct - for example, could prove that $1 + 1 = 2$. Consider conceptualising the "natural numbers" including zero as a domain model. Part of such a model, describing numbers in "algebraic" notation, is shown in Figure 1.4.

To prove "1+1=2", a system could us the following reasoning:

```
zero :    -> number
successsor : number -> number
add: number  number -> number
statements:
(1) For all n in numbers: add(zero, n) = n
(2) For all n, m in numbers:
      add(successor(m), n) = successor(add(m, n))
```

Figure 1.4: Model for cumputing numbers.

First, we must consider what the representation of “2” is in the domain model language. Its representation is “successor(successor(zero))”. In the same way, “1+ 1” is represented in the domain model language as “add(successor(zero), successor(zero))”.

The automated reasoning that forms the proof would then proceed along the lines in Figure 1.5.

```
add(successor(zero), successor(zero)) =
(by (2))
successor(add(zero, successor(zero))) =
(by (1))
successor(successor(zero))
```

Figure 1.5: Example of proof within model of number theory.

**Example 2:** For a complex example of domain modelling underlying intelligent systems, we refer to the work of McCluskey et al [96] in the area of air traffic control regarding oceanic separation criteria.[6]

Sources of knowledge that were used to construct the domain model were training manuals, operational manuals (the Manual of Air Traffic Services, Part II), existing software tools, existing software documentation, Air Traffic Control Officers and other personnel.

From exploring the requirements of the project, it was decided that the domain model needed to contain objects, relations, time, constraints, rules, and logical statements, but it does not need to contain state change, actors, belief, or uncertainty. Here is the paraphrase of an extract from a source manual:

*If a profile containing a segment is wholly or partly in Shanwick OCA then a segment of such a profile starts at or after the first recognised*

[6]http://helios.hud.ac.uk/scomtlm/Artform/impress.html

> *point for oceanic conflict prediction if and only if the entry time of that segment is at or later than the time of the first recognised 4D point for oceanic conflict prediction of the profile containing that segment. For subsonic aircraft, the minimum longitudinal separation between turbojet aircraft, meeting the MNPS, and operating wholly or partly in MNPS airspace, shall be 10 minutes, provided that ... [ETC]*

After much domain analysis, we decided to represent the knowledge using a hierarchical language, where the higher levels contained many-sorted first order logic statements. The low levels contained algebraic definitions of basic objects, functions, attributes etc (not unlike the algebraic specification example of numbers in Example 1 above). An example from the 100's of axioms making up the higher level, is one capturing the idea of type of conflict, as shown in Figure 1.6.

```
AXIOM 2.3.2
(Segment1 finishes_earlier_than Segment2 starts) =>
[(Segment1 and Segment2
  are_in_lateral_and_longitudinal_oceanic_conflict)
<=>
[(Segment1 and Segment2 are_subject_to_oceanic_cpr) &
not (Segment1 and Segment2
     are_deemed_to_be_laterally_separated) &
the_time_gap_Val_between(Segment1,Segment2) <
the_min_longitudinal_sep_Val_required_for(Segment1,Segment2) &
(the_entry_Linear_track_pt_of(Segment2)
   is_within_conflict_box_of
 the_exit_Linear_track_pt_of(Segment1))  &
(the_exit_Linear_track_pt_of(Segment1)
   is_within_conflict_box_of
 the_entry_Linear_track_pt_of(Segment2))]]
```

Figure 1.6: Example of model of an axiom.

As the model was being created, a range of reasoning tools were created both to help in validating the model, and to operationalise it once it was finished. The relevance of this example is that it illustrates how this kind of modelling could be used in road traffic support systems: here domain models could be used to represent:

- road network effecting actions that the RTS can carry out
- information seeking actions

- capabilities of system components
- policies that RTS has to follow
- safety constraints

Representing such knowledge in domain models, and reasoning with them using reasoners (such as AI planners), has the potential to enable autonomic properties. For example, an AI system that can reason with component capabilities to combine components to make or fix a service, would represent self-healing, while the enacting of information seeking actions helps an agent's situational awareness. In the same way, AI planning could then use the domain model to create plans to help in demand management and resource management, as illustrated in the previous section. The set of actions represented in PDDL, decribed in the last section, is another example of a domain model.

Clearly, statements in the domain model must model knowledge in the application accurately and adequately, so that the reasoning with the model will give results that are consitent with the application. What is and is not represented in the model is another central issue here, and to investigate this we turn next to knowledge engineering.

### Knowledge Engineering

The field of *Knowledge Engineering* arose initially from the Knowledge-Based Systems (KBSs) community, following from the developments of Expert Systems in the 1970s, such as Mycin [139]. The field's emergence had a similar goal to the rise of Software Engineering in the software industry: "turning the process of constructing KBSs from an art into an engineering discipline" [149]. Central to the methodologies or frameworks developed subsequently was one which involved creating a precise, declarative and detailed model of the area of knowledge to be engineered, with an emphasis on the lifecycle of such models and the evolution of the encoded knowledge over time. This emphasis was partly in response to the apparent failure of earlier approaches to knowledge-based system construction, which appeared to focus on the "transfer" of expertise at a more superficial level. CommonKads [134] is considered the most important of these methodologies. It emphasises the modelling of the expertise and application area, leading to the construction of the *expertise model* which contains a mix of knowledge about the problem solving strategies needed within the application, and the declarative knowledge about the application.

Often a key rational for knowledge engineering is to create declarative representations of an area to act as a formalised part of some requirements, making explicit what hitherto has been implicit in code, or explicit but only residing in informal documents. With re-use in mind, the emphasis in the KBS community has focused on the development of shared, decentralised, agreed statements of knowledge about an area - in other words "ontologies", giving the advantages of the sharing of knowledge components. Currently the acquisition, engineering, maintenance and integration of ontologies themselves consumes a large part of the knowledge engineering literature, as well as the construction of ontology engineering environments such as Protege [118].

### 1.2.6 Evolutionary Computation

Evolutionary Computation emerged in the 1960s and follows the idea that evolution could be used as an optimization tool for engineering problems. John Holland invented Genetic Algorithms (GAs) with the goal to formally study the phenomenon of natural adaptation as it occurs in nature and to import the idea into computer systems [100]. His original algorithm [74] is a method to start from a population of "chromosomes" (each one a sequence of "genes" or bits that code a possible solution to a problem) and evolve it into a new population using a "natural selection" mechanism together with crossover, mutation and inversion operators. Crossover exchanges parts (subsets of genes) between two chromosomes, mutation randomly changes a gene, and inversion reverses the order of a subsequence of the chromosome. Using a carefully designed *fitness function*, each individual can be evaluated.

Some evolutionary algorithms like Scatter Search (SS) [94] try to combine solutions from a small reference set in order to find centroids with the ultimate goal to achieve better solutions. There is no randomness in the selection process and the distribution of selected points to combine is relevant (not only quality but also diversity will be considered for being included in the reference set).

Other types of so-called evolutionary algorithms have been relevant in the literature. Particle Swarm Optimization (PSO) or Ant Colony Optimization (ACO) are well-known Swarm Intelligence algorithms, that try to imitate the collective behaviour of decentralized individual agents interacting in some manner among them to find a goal. In PSO [114], introduced in the 1990s, a set of "particles" is placed in the search space of some problem, and each evaluates the fitness function at its current location. Each particle determines its movement in the search space combining its history of current and best fitness with other particles' history and introducing some randomness in this process too. Eventually, the "swarm" moves towards the optimum of the fitness function. ACO algorithms [44] imitates the social behaviour of insects, in this case ants, that deposit pheromone on the ground for other members of the swarm to follow a path to a target goal (e.g. food). ACO was actually used in the 90s to address the vehicle routing and traveling salesman and later applied to other graph-related problems.

Harmony Search [64] is a meta-heuristic population-based algorithm that imitates the behavior of a music band in their attempt at improvising the most harmonious melody. Musical instruments are comparable to decision variables, their pitch range is the value range, the harmony is the solution vector and aesthetics is comparable to the fitness function.

- Genetic algorithms: have been used, among many other things, for optimization of traffic signals. Teklu et al. [155] consider the problem of optimizing signal green and cycle timings over an urban network, in such a way that the optimization anticipates the impact on traffic routing patterns. This is achieved by including a network equilibrium model as a constraint to the optimization. A genetic algorithm is devised for solving the resulting problem, using total travel time across the network as an illustrative fitness function. A similar approach was presented in [108], where the system performance index is defined as the sum of a weighted linear combination of delay and number of stops per unit time for all traffic streams. A genetic algorithm is devised to minimize the system performance index, using its inversion as fitness function. The results showed better performance that previous heuristics algorithms. More recently, Sánchez et al. [131] considered four variables in a multicriteria fitness function to address congestion situations: the number of vehicles that left the network after the simulation, the mean travel time to leave the network, time and state of occupancy and the global mean speed. They encode chromosomes as sequence of green, red, yellow traffic lights in all the intersections considered in the network.

If the representation of individuals is generalized we can obtain richer models, as it is the case of Genetic Programming. Some works have used it for learning rules to model traffic-related time series [183].

- Multi-objective evolutionary computation: in [72], its authors present an evolutionary computation approach for dynamic ride-sharing. In [36] the problem of routing school buses in rural areas is addressed with the objectives of minimizing 1) the transportation time that a student passes in the bus and simultaneously 2) the cost, i.e. the number of buses needed. Its authors propose to use SS, where their reference set includes good solutions (in terms of both objectives) and dominated solutions (to provide diversity). Instead of looking to maximize a unique multi-objective weighted function, they designed two heuristics functions directed to find good solutions for each objective, and then combine these solutions using a voting scheme, to be included in the reference set and continue the SS. At the end, the decision maker can choose from a small subset of solutions the best compromise of cost and quality of service.

- Harmony search: the work in [128] addresses the reconfiguration of one-way roads in a city after the occurrence of a major problem (e.g. a long-term road cut) in order to provide alternative routes that guarantee the mobility of citizens, by means of Harmony Search [65].

- Swarm intelligence: García Nieto et al. [63] propose a PSO approach to find successful cycle programs (coded as vector solutions) of traffic lights, obtaining significant profits in terms of two main indicators: the number of vehicles that reach their destinations on time and the global trip time, with respect to cycles defined by experts. In [73], authors present swarm-based traffic simulator, where vehicles modelled as ants communicate with each other by depositing and detecting pheromones, and optimize traffic light sequences by voting for their preferred traffic lights settings.

In [82], a combination of hierarchical fuzzy model and ACO is proposed to adjust efficiently the road traffic according to the real-time changes in road networks by sharing information of vehicle equipped with GPS. Three types of agents (city, road supervisors and intelligent vehicle agents) are interconnected hierarchically. Each intelligent vehicle agent has to find the best possible itinerary at each intersection. The road traffic quality is interpreted as the normalized average speed of the vehicles in that segment. It is a value between 0 (i.e. traffic jam) and 1 (i.e. free flow). The value is represented like the pheromone quantity deposited by many ants which traveled the same road. The hierarchical fuzzy model is used to get a preference index for each itinerary evaluation -at intersections-, where contextual factors as departure time, weather information, roadwork information, usual driving speed, path max. speed or driver's familiarity for the path are inputs to the model.

- Hybrid techniques: Traffic solutions based on evolutionary computation often combine multiple algorithms. For example, in [99] and [180], reinforcement learning (RL) is deployed locally on individual traffic light agents, but globally evolutionary algorithms are used to find optimal combinations of parameters for the local RL processes. In [117], evolutionary algorithms are used in combination with Learning Classifier Systems [19] to evolve traffic light parameters for a specific traffic situation and evaluate them in an offline simulation. In [12], evolutionary game theory is used to model individual traffic controllers as agents that are capable of sensing their local environment and learning optimal parameters for continually changing traffic patterns. Agents receive both local reinforcement from their local detectors, and global reinforcement based on global traffic conditions. Another example on urban topology and traffic control using meta-heuristic techniques can be found in [24]. For a survey of intelligence methods for traffic signal control, we refer the reader to [90].

### 1.2.7 Machine Learning

Machine learning (ML) deals with the task of automatically improving the behavior of a computer system based on experience with respect to a given metric [101]. There have been two main types of ML systems: those that analyze (big quantities of) data to acquire new knowledge; and those that improve an existing problem solver. The first type has generated very related fields such as data mining, data analytics, big data, text mining, or social mining among others. The second type has been less studied and related to work on learning in combination with problem solvers such as rule-based systems, search, or automated planning. In the first type, usually experience takes the form of instances, with or without a class label.

Among those of the first type, a standard classification relates to the labels assigned to instances. So, we can differentiate between: supervised learning (all instances have a class label), which can be used in turn for classification (discrete class) or regression (continuous class); semi-supervised learning (some instances have a class label, most do not); reinforcement learning [79] (where there is no class, but there is a reward every now and then); or unsupervised learning (no instance has a class label) and can be used for clustering or for learning association rules. Another dimension for classification relates to the representation paradigm of input instances. Most work assumes examples are represented as vectors, where each position corresponds to the value of a given attribute. But, there has also been extensive work on techniques that take as input relational representation of examples, such as FOIL [120]. In those cases, the techniques are usually called relational learning or Inductive Logic Programming (ILP), given that the output can be seen as logic programs, as in Prolog.

Summarizing all relevant ML techniques goes beyond the scope of this roadmap, as it is a field that has had a huge increase in publications in the past decades. To mention just a few well-known techniques, we can cite:

- Supervised learning: decision trees and rules [121], Bayesian techniques [20], neural networks [172], support vector machines (SVM) [37], or instance-based techniques [57].
- Unsupervised learning: $k$-means [91] or EM [40]
- Reinforcement learning: Q-learning [166]
- Ensambles of learning techniques, that combine several techniques to improve performance of individual techniques: bagging [18], boosting [133], or stacking [175].

ML techniques are very relevant for Transport management, since they can be applied for all kinds of tasks, such as adaptive traffic lights control, traffic forecasting, early incidents detection, or vehicles tools as traffic signals detection or collision avoidance. ML is also a key component of autonomic systems, since one of their main features is self-adaptation, which can be realized by using ML.

In supervised learning, instances are labeled with a class value, either finite (classification) or infinite (regression). We will see next some examples of the use of these techniques in the Transport area.

- Classification: using relational learning (where instances are represented using predicates), there has been work that learns to predict traffic incidents with general ILP techniques [50] and with nFOIL, an ILP technique based on FOIL [93]. When compared with neural networks and support vector machines, the approach was competitive. A current trend in vision consists of using ML techniques for detecting some specific targets. For instance, classifiers have been used for vehicle recognition and tracking using various approaches as SVMs or ensambles of classifiers, as Adaboost (boosting), in [141]. A neural-network approach was used for the same purpose in [78].

- ML Regression: in [29], the authors compare several techniques for traffic prediction (SVM for regression, Bayesian networks, neural networks, instance-based, ...). A survey can be found in [34]. Neural network based controllers of traffic lights [144]. Also, learning traffic light controllers with neural networks and logistics regression from humans have been used in [15]. SVMs for incident detection were defined in [32], as well as ensambles of SVMs [31] or decision trees [30].

Other ML approaches that have been used in RTS are:

- Reinforcement learning (RL). RL deals with tasks that require making decisions (taking actions) in a sequence, mostly in the presence of uncertainty and without knowledge of the environment. Thus, RL systems are on-line systems that can be used for continously learning and acting. Examples of their use in traffic lights control for only one junction are [116, 5]. Another section of this roadmap will cover multi-agent RL.

- Case-Based Reasoning (CBR) [86, 3]. CBR sees problem solving as the accumulation of past problem episodes and their solution in a case base, and then solving new problems by reusing and adapting previous solutions to similar past problems. Cases can use any representation formalism. An example of their use in the ARTS context is in personalized multi-modal transportation [14]. Another example consists of computing routes within city maps [70].

- Learning in problem solving [77, 184]. There has been quite a number of approaches for learning in the context of general problem solving as applied to search, games, or planning. As far as we know, there have been yet no applications to areas related to ARTS, so they present a good opportunity to approach self-adaptation in this kind of systems.

- Comparison among several techniques. For instance, the work presented in [7] presents a comparison of Q-learning, neural networks and fuzzy systems to the task of controlling traffic signals.

### 1.2.8 Architectures for AI

In order to build complex applications, we need to consider how the different components will interact. There has been also plenty of works describing generic architectures for integrating different problem solving capabilities. Most of them include at least two autonomic computing properties.

- Generic problem solving: some work in the past focused on integrated architectures. Examples of such were SOAR [87], Prodigy [163], ACT [6], or Theo [102]. Those included a generic problem solving technique (as rules, frames or planning) and a machine learning component.

- Planning & Scheduling: there have been plenty of architectures that range from low-level control to integration of a deliberative and reactive components. Most of these architectures have been used for the control of robots, though the type of problems are common with RTS tasks (such as uncertainty, scalability). Examples are APSI [28], PRS [66], IxTeT [67], IDEA [10], TCA [140], PELEA [69], T-Rex [97], or ABLE [145]. An interesting recent direction that can help on the autonomic behavior of RTS is the idea of Goal-Driven Autonomy [103]. It consists of planning and execution architectures that reflect on the execution failures to detect when the goals of the system should change according to unexpected perceived states.

- Event-driven: some new approaches applied to traffic management rely on event-driven architectures [47].

## 1.3 Challenges

- Knowledege representation: what formalisms are better suited for AC and RTS applications? Currently, most people tend to use ontologies, as flexible ways of defining data bases. Most AI techniques would work fine with an ontology-based representation. Ontologies have the main characteristic of being shared by many different organizations and people. Therefore, there is an standardization effort to be performed, so that they can efficiently be used, providing interoperability.

- Planning:

  – Representation: one of the key issues in building intelligent systems consists on defining the right model. In this case, we will probably need two (or more) levels of representation. Also, there is a need of setting up the mapping in both directions. From the sensory information, to a high-level representation in predicate logic, and from a high-level representation of actions back to low-level or primitive actions. An open question is how to maintain in parallel both levels when plans are in execution.

  – Domain model: related to the previous aspect, one has to define the right level of abstraction for actions, such that planning makes sense (it is a non-trivial process), while making it tractable (it returns a solution within a sensible time).

  – Planning model: there are also decisions to be made with respect to the kind of planning paradigm to use: STRIPS, numerical, hierarchical, temporal, probabilistic, .... This decision greatly influences expressivity of the domains that can be used and the computational complexity of the planning process (time and space to solve the problems).

  - Goals: autonomous systems should be able to maintain the set of current goals and change that set dynamically. This aspect has not been studied in depth, and requires to take into account models of goals costs, benefits, priorities, and also taking into account temporal models of these values decrease or increase (e.g. a goal can have a high priority now, but its priority might decrease/increase over time).
  - Multi-agent planning: current traffic management systems can be perceived as multi-agent tasks, given that multiple users can make decisions on those systems or interconnected systems. Thus, there is a need to extend this kind of architectures to a multi-agent setting.
  - Computational complexity: given the size of many of the traffic related planning tasks, most planners will not be able to handle even small-size tasks. Thus, current planners will have to be enhanced with techniques that are able to: decompose the problems in sub-problems without much loss in optimality; filter out unimportant parts of the problems; reconsider previous solutions; or remove symmetric parts of the problems, among other solutions.
  - Planning and execution. Traffic tasks are highly dynamic and uncertain. Thus, planning architectures should be able to integrate planning, execution, monitoring, learning, and, probably, goal-driven autonomy ideas.

- Machine learning: there are plenty of open research questions related to data analysis. Currently, there is huge amounts of historical data, streaming data (data coming incrementally and on-line), or multi-modal data (videos, sound, highway sensors, ...). Besides, the quantity of data is becoming to be overwhelming (Big Data), so there is a need of techniques that can handle such quantities in real-time for various Transport related tasks.

- Architectures: how to integrate different components (blackboard, fixed architecture, three layers architectures, ...). Independently of the chosen method for the integration, there is plenty of room for standardization for a clear specification of the APIs among components. Some fields do have some clear APIs, as for instance the PDDL language for automated planning. But, even in those cases, not all planners can use the full standard, which is quite rich.

## 1.4 Actions

We propose here some actions that relate to the challenges presented before.

- Standardization of various aspects, so that big and complex autonomous systems can be developed and (self) maintained.

- Pilot studies that would show the benefits of using the AI technology.

- Definition of benchmarks that would foster the comparison among techniques in tasks as route planning, traffic lights control, or incidents management.
- Develop new planning techniques that scale well in traffic domains.
- Develop techniques that can work at different levels of abstraction that are common in RTS tasks.
- Integration of planning (or problem solving in general) with execution monitoring and GDA components for continuous autonomous behavior.
- Use of huge amounts of data arriving in real-time to improve decision-making.
- Definition of architectures that can integrate different AI and non-AI components.

# Chapter 2

# Multi-Agent Systems

Ivana Dusparic (Trinity College Dublin) and Franziska Klugl (Orebro university)

## 2.1 Introduction

In order to enable the implementation of self-* behaviours, the autonomic computing field draws on and extends research and technology from multiple scientific disciplines such as artificial intelligence, distributed systems, network management, fault-tolerant computing, requirements engineering, statistical modeling, trusted computing and others [22, 95, 147, 148]. In particular, concepts from control theory [4] and multi-agent systems (MAS) [158] have been used in the design of autonomic elements and the interaction between them.

Modeling an autonomic system manager as a closed control loop is inspired by control theory [41], where a controller observes the performance of the entity that it is controlling and determines control signals that optimize a given performance criterion [85]. Often, performance needs to be optimized for multiple criteria, requiring the use of multi-objective optimization techniques. Traditionally, multi-objective optimization techniques in control theory include vector optimization, nonlinear multi-objective programming, goal programming, fuzzy multi-objective linear programming, and evolutionary algorithms [52]. However, use of control theory relies on the existence of precise and correct models of a system and accurate estimates of the effects of its operating environment [4], which are complex and time-consuming to develop [157]. To overcome this difficulty, techniques and ideas developed in the MAS community, which do not rely on complex models for their performance, such as self-organization and emergent behaviour, multi-agent learning, and agent coordination, are being utilized in the engineering of decentralized autonomic systems [158].

This section of the Roadmap provides an introduction to intelligent agents, agent learning and multi-agent systems, concentrating on aspects and techniques that have been shown to be interesting for traffic and transportation in general

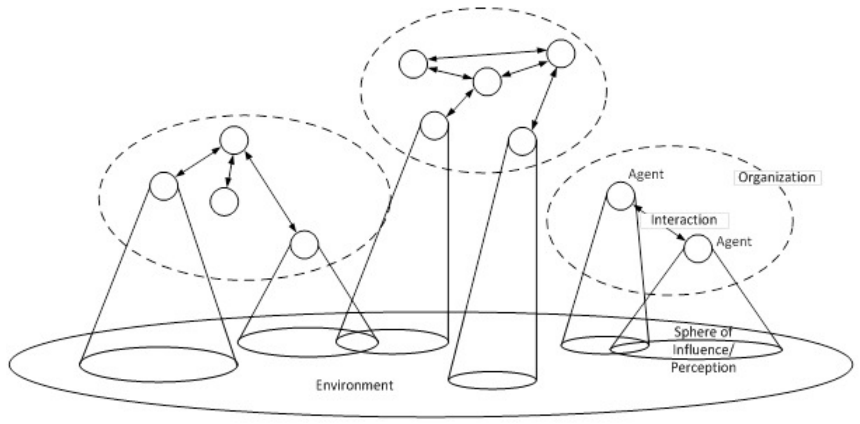

Figure 2.1: Elements of a Multiagent Systems [176].

and we foresee to be useful for autonomic computing in road transportation. For a more thorough introduction to multiagent systems in general, a number of good text books have been published in the last years, such as [176], [171] or [137].

## 2.2 From an Agent to a Multiagent System

An agent is defined as "a computational entity that is perceiving and acting upon its environment, and that is autonomous in that its behaviour, at least partially, depends on its own experience" [170]. An intelligent agent is such an agent that acts towards pursing its goals [170]. Additionally, agents can have the capability to learn how to achieve their goals, i.e., they can be learning agents [127]. The learning process carried out by an agent refers to all activities executed with the intention of meeting the particular goal [135]. As part of this process, agents can learn from other agents, or learn about other agents, if that information can be used to enable better local decision making [135].

A MAS is defined as "a loosely coupled network of agents that interact to solve problems that are beyond the individual capabilities or knowledge of each agent" [152]. Multiagent systems arise when multiple agents are put together interacting in the same environment. Figure 2.1 illustrates the overall picture: agents are interacting with each other and a shared environment. Each agent just has local perception and a restricted sphere of influence. The effect of its actions there might be perceivable by other agents. Interactions may be structured in organizations that may not only group the agents but also exert some form of control over its members. Hereby, all elements may be dynamic. Multiagent systems may take very different forms from few agents in a highly structured and fixed organization to high numbers of agents interacting by perceiving the others' changes and without pre-defined form of organization.

As discussed in a recent review on agents in traffic and transportation [13], agent-based approaches suit very well for solving traffic-related problems given the geographical, functional, and temporal distribution of data and control, as well as the frequent and flexible interaction between the participants and their environment. In traffic applications, different forms of multiagent systems can be identified, for example in agent-based traffic light control the problem is to coordinate the signal plans of traffic-light agents, hereby the spatial distribution is predefined and thus the neighbours with whom the agent need to communicate for potential coordination. An example for the other extreme can be found in agent-based route choice simulation for testing the effect of drivers' reaction on information. Hereby the interaction between drivers is restricted to taking choices in a shared environment that have an effect on all agents' travel times.

## 2.3 Interaction, Self-Organization and Emergent Phenomena

One of the main characteristics of MAS-based techniques relevant to autonomic computing is that they can exhibit self-organizing behaviour, i.e., MAS do not require a central management component or a global view of the system. All learning, actions and interactions can be performed locally by the agents, while global behaviour emerges from these actions and interactions. One may identify two basic types of interaction between agents: direct interaction, often in form of message passing that we will discuss below, or indirect interaction where actions of one agent have an effect in a shared environment that can be perceived by other agents and thus have an effect on the others' behavior. These so-called stigmergic interactions form the basis for many self-organizing and emergent phenomena that are analysed using agent-based simulation. Some works also use multiagent system approaches to engineer Self-organization and Emergence. Emergence and self-organization are the phenomena that occur in complex adaptive systems [58]. One can find several slightly different definitions of these terms, we will refer to self-organization as a property of the system that allows it to function without external control, and emergence as a property of the system that enables global behaviour to arise from purely local actions and interactions (following the views of [174]). A classical example for self-organization can be found in situations in which commuters adapt their routes based on their experiences ending up in a balanced load distribution – described as Wardrop Equilibrium. Moving congested areas from high density of vehicles is an emergent phenomenon: the congested area moves upstream, opposite to the cars of which it consists. Agent-based traffic simulations are used to capture such phenomena, they may serve as mobility simulations (as e.g. reviewed in [83]) with the emergence of traffic jams or capture the full decision making based on fully elaborated daily plans containing typical activities, departure times, route and mode choices such as MATSIM [11] that when repeated and optimized end up in some distribution near to an equilibrium. In all these systems, there is no direct communication between traffic participants or drivers, interaction happens because agents share the environment and each agents' choices affect the outcome for the others.

## 2.4 Learning in Multi-Agent Systems

Learning is defined as an acquisition of new knowledge and the incorporation of that knowledge into future activities, provided it leads to an improvement in performance [135]. Agents in a MAS can learn independently or interactively, and interactive learning agents can differ in the type of information they exchange [154]. The purpose of interaction could be to ensure locally optimal behaviours are also globally good (e.g., in [45]) or to improve the quality or speed of the local learning process (e.g., in [154]). Artificial Intelligence subsection of this Roadmap provides an introduction to various machine learning categories and techniques, while [135] provides a detailed overview of multi-agent learning. In this section we focus on reinforcement learning, as an example of how can a machine learning technique be applied in multi-agent learning, as it has been widely applied in intelligent urban traffic control and in autonomic computing in general.

### 2.4.1 Multi-Agent Reinforcement Learning

RL [151] is a learning technique based on trial and error that has been researched and applied in control theory, machine learning and artificial intelligence problems, as well as non-computer science domains such as psychology. Q-learning [167], a model-free implementation of RL, is considered particularly suitable for optimization in situations where no pre-specified model of the environment is available [5], as is likely in complex large-scale decentralized systems. In Q-learning, an agent learns to associate an action with the expected long-term reward of taking that particular action in a particular state.

Q-learning is a single-agent, single-policy learning technique. A number of multi-policy extensions have been implemented, as agents often have to simultaneously meet multiple policies. Multiple policies can be encoded into a single Q-learning process, with a larger state space which is a cross product of individual policy states (approach which is not scalable if the number of policies is large), or multiple policies can be implemented as separate Q-learning processes, with an arbitrator deciding or learning which policy should win at each time step [48].

Reinforcement learning can also be used in multi-agent scenarios, where a group of agents is used to break down a complex large task into a group of smaller ones to be solved in parallel in a decentralized manner, or if the problem requires separate individual entities (i.e., agents) with their own individual goals to cooperate or collaborate to solve a global optimization problem. Agents in multi-agent reinforcement learning (MARL) can be addressing the individual tasks without communicating with each other (so called independent learners) or can collaborate to solve the global problem or to help each other solve individual tasks (so called joint learners). For a comprehensive review of multi-agent reinforcement learning please refer to[21].

Reinforcement learning has been used in variety of autonomic systems, for example in resource grid scheduling[113], load balancing [46], and autonomic network repair [89]. The next section focuses on its use in autonomic road transport systems, specifically to urban traffic control implementations.

### 2.4.2 MAS in ARTS Use Case: Applying Reinforcement Learning to Urban Traffic Control

RL algorithms have been widely applied in the optimization of urban traffic control (UTC) systems. It is considered particularly suitable for self-optimization in large-scale heterogeneous environments such as UTC, as it does not require a model of the environment, which, due to the scale and complexity of such systems, is time-consuming and complex to construct[5]. A recent comprehensive review of RL-based approaches in UTC is provided in [13], but in this section we provide a brief summary of the types of approaches that can be used depending on the problem characteristics.

RL-based approaches to UTC differ in whether the optimization problem is solved using a single agent or a group of agents, and whether agents on which RL is deployed collaborate with each other or not. The decision which approach to use depends on the scale and complexity of the problem.

For example, if traffic flow optimization needs to be addressed on a single isolated junction (possibly in a non-urban area), a single agent Q-learning approach is used [5]. If a scale of the traffic network is sufficiently small, traffic flow can be optimized using a single RL agent to optimize the flow on multiple junctions in a centralized manner (e.g., in [115]). Most often, however, optimization on multiple junctions is done using multiple agents. In some approaches (e.g., in [23]), agents optimize their local traffic flow independently without communicating, cooperating or coordinating with each other. Most frequently though, as the performance of one junction directly influences the performance of its neighbouring junctions, the agents controlling individual junctions cooperate in some manner. In [122] agents communicate with their immediate neighbours (i.e., agents controlling the first downstream and upstream junctions), in order to use neighbours' traffic counts to anticipate their own traffic flows. In [130] agents periodically exchange information on their performance (their accumulated rewards) with their immediate neighbours and incorporate the information received from neighbours into their own reward, therefore being rewarded for their neighbours performing well, motivating them to take their neighbours' performance into account when making local decisions. In [49] agents directly learn how their decisions influence the performance of neighbouring junctions and take the preferred actions of the neighbours into account when making local decisions. [53] simplifies the local interactions by having each agent learn joint performance policy with only one of its neighbours at a time.

RL-based approaches to UTC also differ in what it is exactly that they are learning, i.e., how is the problem defined and how is the optimization of the overall system achieved. In [5] the state space encodes information about the queue length on all junction approaches, and the reward is inversely proportional to sum of delay for all vehicles on all of the junction's approaches. In [23] the reward is inversely proportional to the number of vehicles waiting at an intersection, while in [122] a reward is based on the number of vehicles each phase allows to proceed. [130] combines the above approaches and in their work an agent is rewarded proportionally to the number of cars that pass through an intersection during the phase, and inversely proportional to the number of cars still waiting at an intersection. [49] reward the agents if action selected cleared more traffic than it has arrived at the junction in the meantime.

The actions available to the agents also differ. In [5] an agent can either continue with the current phase or switch to the next one (where the sequence of the phases is fixed and predefined). In [130] the sequence of the phases is also fixed and predefined, but an agent can choose the duration of the next phase, including a duration of 0, effectively allowing it to skip a phase. In [49] agents can at each decision step choose any phase to use out of a predefined set of phases, and can use them in any order, allowing it to effectively learn to favour a several most optimal phases and never use those that they learn to be less optimal.

## 2.5 Challenges and Actions for the Community

Although multi-agent systems have shown to be suitable for addressing urban traffic optimization problems, a number of open challenges remain to be addressed before their implementations in the real-world conditions is feasible.

As MAS implementations often rely on agents finding suitable solutions through iteration and learning, arriving at an acceptable solution takes considerable amount of time. Therefore, most of the approaches propose that learning is done offline (in a simulation) and algorithms deployed only once acceptable performance has been achieved. If there are a few different sets of conditions, performance for each is learnt, and context information can be used to switch between the pre-learnt behaviours (e.g., [39]), or agents can learn to detect that the conditions have changed and tweak their learning parameters to adapt to the new conditions (e.g., [129]). However, if the whole new set of conditions is encountered in the system (e.g., a traffic accident at a place where it hasn't occurred before), learning the behaviours for the new set of conditions must occur online with no loss of performance. Addressing this problem requires either further investigation in MAS algorithms or combining MAS with a different technique to address it.

Existing MAS implementations of UTC only address some of the self-* properties required for the full autonomic performance. For example, self-configuration and self-optimization are achieved through learning, but further investigation is needed into others, such as self-healing and self-protecting properties.

Finally, all of the proposed solutions have so far been evaluated only in the simulation. The simulations in some cases have been very detailed and realistic, so the algorithms are very mature, however the next substantial step is to evaluate them in the real-life trials.

# Chapter 3

# Game Theory concepts for autonomic RTS

Luis A. Garcia (1) Vicente R. Tomas (1) László Z. Varga (2)
(1) A$i$A Research Group
Dept. Engineering and Science of Computers.
University Jaume I, 12071 Castellon, Spain

(2) ELTE Regional Knowledge Centre,
Irányi Dániel u. 4., Székesfehérvár, H-8000 Hungary

## 3.1 Summary

This section exposes the state of the art in the field of game theory of those elements in the classical game theory and other inherently related areas, algorithmic game theory and computational game theory, we consider closer to the action COST-ARTS: non cooperative and cooperative games. Then, problems and challenges about their application in practical environments are described.

## 3.2 Foundations

Traffic and transport scenarios are populated by several independent and self-interested agents. The actions performed by each agent may affect the behavior of, at least, some of the other agents. For example, as more vehicles follow the same route more congested this route will be and less-happier drivers on this route will be. Organize and control these scenarios is a difficult task that is currently addressed by defining and implementing new centralized and distributed approaches. Current advances in information technologies and the large capacity for modern devices for wireless communicating between them is promoting putting forward novel systems in traffic and transport that may benefit from modern communications paradigms as V2V (Vehicle to Vehicle), V2I (Vehicle to Infrastructure) and I2I (Infrastructure to Infrastructure) [1][2]. One of the available mathematical models to analyze and study the interactions between these communication devices is game theory. The rest of the paper is devoted to explain some of the currently used terms and results, but this is by far, not an exhaustive or complete description of the field of game theory.

Game theory studies the interactions between self-interested players. The result of this interaction relies on the decisions taken individually or collective by these players. A game involves the description of the players, the actions they can play in each movement, and the benefits, in some sense, of the performed actions. Each player chooses its own path of performed actions, that is known as its strategy. Players compute and apply strategies based on the information they can observe or it is already known in trying to maximize each individual benefit, i.e. players act as *rational* entities.

This general description of concepts gives rise to different types of games when decisions are taken about information availability (*complete* or *incomplete*), information completeness (*perfect* or *imperfect*) and rules of the encounters.

One main issue in all these kind of games, from the point of view of classical game theory, is to define how to express the benefits for all the involved players with each possible action set and how each player has to build its strategy in order to act rationally. The benefits for each player are usually expressed by using *utility functions*. These functions assign a real number to the setting produced by the execution of each particular action for each agent.

There are two kinds of strategies: pure strategies are those performed on *complete* information settings publicly available for the players and *mixed* strategies are those in which the players choose the action to be performed based on a probability distribution. There are several solution concepts used to build these strategies. The concept of equilibrium aims the solution concept in which every agent does not want to deviate from the course of its actions that gives rise to the equilibrium solution. In other words, in an equilibrium every agent receives more utility with his current action than the utility he could receive performing another different action.The more used concept of equilibrium is the *Nash equilibrium*: it is the overall profile of actions that for each agent selects the best action he can play subject to the best response actions played by the other agents. However, there are several difficulties when dealing with equilibrium concepts: an equilibrium may not exist for a particular game, there can be multiple equilibrium and the players can have serious problems trying to calculate these equilibrium or even be aware of their existence. More importantly, there are scenarios in which reaching an equilibrium is not necessarily the optimal solution for the game. An example of this situation in the field of networks happens in the Braess' paradox [51].

Games are usually represented as strategic games where players manage information about the benefits of each action being simultaneously performed. Each player in this representation receives its benefit depending on the actions performed by the other players. This interaction can be analyzed from the point of view of single-shot or repetitive-shot encounters. Another useful representation is the extensive game representation in which the players act on turns, so each player can observe, at least, the last performed action. These games are usually represented as a tree, where its leafs are the potentially reachable solutions of the game, and the inner nodes represent decision steps in which players have to choose which action to play.

There are two broad types of games: non cooperative and cooperative. Non cooperative games are those in which players act in a selfish way. Each player in a non cooperative game tries to maximize its individual value for utility. In a cooperative game, players propose binding contracts, between some of them, that can be signed prior to starting the game. This situation promotes coalitions of players that compete with each other. Once the interaction finishes, the utility obtained for each coalition must be distributed among its members.

Algorithmic game theory studies networks with source routing (section 18 in [106]), in which end users simultaneously choose a full route to their traffic and the traffic is routed in a congestion sensitive manner. Two models are used: nonatomic selfish routing and atomic selfish routing. Nonatomic routing models the case when there are very many actors, each controlling a very small fraction of the overall traffic. Atomic routing models the case when each actor controls a considerable amount of traffic. Both models are studied in detail and showed similar properties. The main difference is that different techniques are required for their analysis, because the nonatomic model basically has continuous functions having unique extreme values, while the atomic model has discrete functions approximating the extreme values at several points.

### 3.2.1 Non cooperative games

In the strategic or normal form representation of these games, the players simultaneously perform their actions, or, seeing it in another way, each player perform its own actions without knowledge about the actions performed by the other players. This representation is usually presented in a matrix form. The following description deals with games of two players, but it can be generalized to more players. There are two players, the column player and the row player. The labels for the rows and columns are the actions available for each respective player. A cell $(i, j)$ of the matrix contains the real numbers provided by the utility function of player 1 performing the action labelled in row $i$ and the utility function of player 2 performing the action labeled in column $j$.

A extensive game representation of a non cooperative game is a detailed description of the sequential structure of the encounters between players. Each player acts in turns. They are usually represented as trees. The essential part of this representation is not the tree, for itself[1], but the specification of the sequence of moves, the strategies available for the players, the information each player holds and observes in each movement and the payoffs received by each player in the deployment of each possible strategy.

[1]Sometimes is not possible to represent all the available actions as branches of the tree in a discrete way, for example, a bid in the continuous range [0,1]

| | Traffic Police | |
|---|---|---|
| Driver | Survey | Do not survey |
| Do not comply traffic norms | $5, 7$ | $8, 4$ |
| Comply traffic norms | $10, 4$ | $6, 7$ |

Figure 3.1: The Tsabelis game. A two row, two column strategic game between a driver and a traffic police unit. The numbers inside cells express the utility real values obtained by each player playing each possible action. For example, If traffic police plays *Survey* and driver plays *Comply traffic norms*, traffic police obtains utility 4 and the driver obtains 10.

In games at this form of representation, the Nash equilibrium concept can give to an absurd equilibrium [159] because the strategies in Nash equilibrium are just held at the beginning of the game and they are not guaranteed in subsequent stages. The concept of *sub-game perfect equilibrium* is usually applied when analyzing extensive games. A sub-game is a part of the game that is a game in itself, i.e., it contains the future consequences of the action that is the root of the sub-game tree and it includes every information needed to play this part of the game. An strategy of a extensive game holds with the sub-game perfect equilibrium if for each one of the deployed strategies in each decision node is a Nash equilibrium.

It is important to note that every extensive form game has an equivalent game in normal form. The transformation to normal form from extensive form may result in an exponential blowup in the size of the matrix representation, therefore making this transformation computationally infeasible [138]. Note also, that a extensive form game can capture simultaneous moves by the players with *imperfect information*.

## 3.2.2 Cooperative games

A cooperative game is a game where groups of $N$ players are promoted to joint forces. Hence the game is a competition between coalitions of players instead of between individual players. The initial set of players is called the *grand coalition* and the *characteristic function* $v$ provides the payoffs for every possible coalition. Almost all cooperative games state that the characteristic function of the empty coalition provides zero value.

There are four important subclasses of cooperative games depending on features of their characteristic function. Monotone games are those that adding a player to an existing coalition can only perform better that without its inclusion, i.e. $v(C) \leq v(D)$ for every pair of coalitions $C, D \subseteq N$ such that $C \subseteq D$. In a super-additive game, players never loose by joining two existing coalitions: given coalitions $S, T$ with no players in common $v(S \cup T) \geq v(S) + v(T)$. Therefore, players usually form the grand coalition. A game is convex if and only if for every pair of coalitions $T, S$ such that $T \subset S$ and every player $i \in N \setminus S$ it holds that $v(S \cup \{i\}) - v(S) \geq v(T \cup \{i\}) - v(T)$, Note that any convex game is necessarily super-additive. Finally, a game is simple if it is monotone and its characteristic function only takes values 1 or 0, i.e. “winning” or “losing”.

One important result in cooperative games is how to express the solution concepts. These concepts can be addressed by using two sets of criteria, those inspired in how well each player’s payoff reflects its contribution, the *fairness*; and those regarding with the incentives for the players to stay in the coalition, the *stability*.

### 3.2.3 Routing games

Algorithmic game theory studies networks with source routing (section 18 in [106]), in which end users simultaneously choose a full route to their traffic and the traffic is routed in a congestion sensitive manner. The algorithmic game theory model of the routing problem is the triple $(G, r, c)$, where

- $G$ is the road network given by a directed graph $G = (V, E)$ with vertex set $V$ and edge set $E$;
- $r$ is the total traffic flow given by a vector of traffic flows with $r_i$ denoting the traffic flow on the trip $P_i$ which is from source vertex $s_i$ of $G$ to target vertex $t_i$ of $G$; and
- $c$ is the throughput characteristic of the road network given by a cost function with $c_e : R^+ \to R^+$ for each edge $e$ of $G$ mapping the total traffic on edge $e$ to the travel time on that edge.

In this model the $G$ graph may contain parallel edges; the cost functions are nonnegative, continuous and nondecreasing; the traffic flow $r_i$ on the trip $P_i$ is routed somehow on the paths leading from $s_i$ to $t_i$; the cost of a path is the sum of the costs of the edges in the path at a given flow; and the cost of a traffic flow $r_i$ on the trip $P_i$ is the sum of the cost of all the paths in $P_i$. The allowed distribution of the traffic flow is different in the nonatomic and the atomic routing problem. In the case of a nonatomic routing problem the traffic flow $r_i$ on the trip $P_i$ may be divided arbitrarily among several paths leading from $s_i$ to $t_i$, while in the case of an atomic routing problem the traffic flow $r_i$ on the trip $P_i$ can be sent on one single path leading from $s_i$ to $t_i$. We assume that in the routing problem each actor is interested in a traffic flow $r_i$ on a trip $P_i$, therefore we will use the term ”actor”, ”agent” and ”traffic flow $r_i$” interchangeably.

A flow distribution is optimal if it minimizes the cost of the total traffic flow over all possible flow distributions. A flow distribution is an equilibrium flow distribution if none of the actors can change its traffic flow distribution among its possible paths to decrease its cost. The equilibrium flow distribution is a rational choice for every autonomic actor, because deviating from the equilibrium would increase the cost for the actor.

It is proven (section 18 in [106]) that every nonatomic routing problem has at least one equilibrium flow distribution and all equilibrium flow distributions have the same total cost. The price of anarchy is the ratio between the cost of an equilibrium flow distribution and the optimal flow distribution. It is proven (section 18 in [106]) that if the cost functions are of the form $a \times x + b$, then the price of anarchy in any nonatomic routing problem is not more than $4 \div 3$. If the cost functions can be nonlinear, then one can create cost functions to exceed any given bound on the price of anarchy of nonatomic routing problems.

In atomic routing problems the existence of equilibrium flow distribution is not always guaranteed. Atomic routing problems have equilibrium flow distribution if every traffic flow $r_i$ has the same value or if the cost functions are of the form $a \times x + b$. If there are more than one equilibrium flow distributions, then their total costs may be different. If the cost functions of an atomic routing problem are of the form $a \times x + b$, then the price of anarchy is at most $(3 + \sqrt{5}) \div 2$. If the cost functions of an atomic routing problem are of the form $a \times x + b$ and in addition every traffic flow $r_i$ has the same value, then the price of anarchy is at most $5 \div 2$.

It is known (section 18 in [106]) that if the routing problem has an equilibrium and the actors try to minimize their own cost (best-response), then the traffic flow distribution converges to an equilibrium.

## 3.3 State of the art in RTS

Anti-coordination games, or the congestion game version for multi-player settings, are those in which several players have to simultaneously choose different strategies. They are usually applied to the analysis of traffic congestion. El Farol bar anti-coordination game [9] states how individually each player decide to go to a same place for all players where too much players are not welcome. There are several variants of this problem considering, for example, to allow communications between the players, prior to perform the action, in which the players are not committed to tell the truth. Lastly, the Minority game is also a variant of El Farol problem in which all the players have to choose between two options, the ones that fall in the minority group win. These games have been used in models to optimize the road traffic assuming that drivers take their individual actions guided by them (see, for example, [59]).

The *Wardrop equilibrium* [142], very related to the Nash equilibrium, is frequently used in transport scenarios as the equilibrium for the analysis of the solutions provided. In this equilibrium, drivers may not lower their transportation cost or time by individually changing their paths subject to the behavior of the other drivers. However, in the definition of this equilibrium drivers have to know in advance the mean travel times of the available paths. An extension of this setting in which the travel times are inherently variables and it uses a Wardrop equilibrium to analyze the solution is proposed in [168].

A Stackelberg game is the one in which a player, the *leader*, first commits to a strategy, and then the other player, the *follower*, selfishly optimizes its own reward about his strategy once the leader has moved. [112] shows a model for analyzing traffic equilibrium when drivers can adopt one of two behavior of driving, passive or aggressive. Interactions between passive drivers show that a Nash equilibrium can be reached. Interactions between an aggressive and a passive driver are modeled by a Stackelberg game, while interactions between aggressive drivers do not reach an equilibrium. [161] presents a game for agreeing the signal control settings among urban roads, a ring road and drivers. The urban road player wants to minimize the total time spent by drivers in its subnetwork. The ring road player wants to maximize the average speed on its ring road, whilst each driver wants to minimize its driving time. Many approaches are possible to model these interactions, mainly as variations of Stackelberg games where drivers follow what the ring road and urban road players have played. Also, Stackelberg games have been used to model the interactions between drivers and parking places in order to attract people to a urban area but also promoting the use of public transportation [75]. The leader of the Stackelberg game is the authority player in charge of the parking places, the other player is the set of drivers who wants driving up to the urban center. The leader player must balanced between not to be very strict in offering too few parking places, because then drivers would choose driving to another place instead of the city center, and in providing enough parking places for avoiding traffic congestion or too much time driving for locating a free parking place. This balance must be guided by the maximization of the use of public transportation.

More recently in [53] a system for setting traffic signals in an adaptive way is exposed. In this system each signalized intersection plays a game with all its adjacent intersections. The payoffs of the game are used to adjust a reinforcement learning approach trying to reach an equilibrium solution for all the intersections in repetitive shot encounters. This systems has been proved in Toronto city for managing up to 60 intersections providing very successful results.

In the field of designing policies traffic, the Tsebelis game (see Figure 1) [160] has been used to analyze their design and results. The Tsebelis game expresses the dichotomy of deploying more police units for surveying traffic with respect to the accomplishment of the traffic norms by the drivers. The Tsebelis game shows how hard are the interpretations of mixes strategies for reaching an equilibrium. This is due because a change in the payment of a player does not affect its strategy mixed, but to the mixed strategy of the other player.

The algorithmic game theory approach of the routing problem has been a hot topic recently. It is well known that individually self-optimizing travel routes does not necessarily result in optimal traffic (optimal for example in term of the sum of the travel times) and each participant may have longer travel time than with central planning. This is known as the "price of anarchy" which was explored by 2012 Gödel prize winners Roughgarden and Tardos. In their paper [125] they investigated the old conundrum in transportation science known as the "Braess's paradox" [16]. The algorithmic game theory investigations revealed important properties of routing games, however the algorithmic game theory approach includes assumptions which do not handle the dynamic online information environment where autonomous individual agents try to self-adapt based on real-time information about the current situation of the traffic.

## 3.4 Challenges for Autonomic RTS

The concepts and models of game theory suitable to be applied to autonomic road traffic and transport systems mainly fall into the Analyze part of the Autonomic Control Loop [43]. The challenges related to these concepts are very similar to those within classical game theory. Some of these problems are related to the prohibitive cost of computing some kind of equilibrium [124]. The fields of *Algorithmic Game Theory* and *Computational Game Theory* deal with these problems by stating new complexity problem classes to classify the hardness of computing the equilibrium in different situations. In this way, approximation algorithms are proposed to solve efficiently a good enough practical solution. There are several constraints that are commonly used in these fields. One of the more frequently used is to suppose that players have *bounded rationality*, i.e., they have limitations about the amount of allowed computation steps for computing strategies, or they only are capable of partially observing the scenario or they have inherent noises in the perception of the environment.

Other problems arise when an equilibrium solution is not optimal or it takes a very long run time for reaching it. An example of this kind of scenario happens in the paradox of Braess. This paradox presents the following scenario. There are two not direct paths connecting two points in a road network. One path is preferable to the other not only based on the quality of the path (road), but also on the density of traffic flow in each path. Therefore, each driver takes the path that looks more favorable to his interest. If a new extension of the road network is proposed without cost (or a very low one) connecting two intermediate points that enables to appear a third path, the overall result can be worst than before because longer individual driving times than before are being produced. This happens because the Nash equilibrium of the initial system no longer holds, because drivers in the extended road scenario have an incentive to change their routes, and the new Nash equilibrium is worse than the previous one, producing a reduction in the overall performance.

Another example [173] is the Tsebelis game. Its equilibrium suggests that an increase in fines for speeding is not a viable policy tool for decreasing driver's speed. However, this game theory result is inconsistent with human common sense and the way the world works. Drivers will slow down the speed proportionally to the increase of amount paying for speeding fines. Using a systems dynamics model for a mixed-strategy game and assuming expected utility maximization of the part of both police and drivers, Kim and Kim [84] show that it takes a very long time for reaching the Tsebelis equilibrium. Thus, the choosing of actions by drivers do not depend on the equilibrium. By this way in terms of traffic policy, the increasing of speeding fines promotes, in the short run,that a majority of drivers decrease their speed.

Another set of problems arises because game theory supposes that human behavior is of a perfect and infallible rationality in the execution of perfectly calculated strategies about perfectly observable scenarios via human cognition. Traffic human actors are far from this theoretical perspective. The field of *Behavioral Economics* [80] studies interactions between real human adversaries exposing new methods and theories more centered in real human capabilities. It is crucial to define and to introduce more realistic models of human decision-making to model the behavior of traffic actors (car drivers, truck drivers, pedestrians, traffic operators, etc.) in autonomic RTS.

Two frequently used theories in this new field are the *Prospect Theory*[81] and the *Quantal Response Equilibrium*[98]. Prospect theory provides a descriptive framework for decision making under uncertainty that accounts on risky situations (mainly a loss aversion attitude in which losses for a player count more than equivalent gains) and variations in how humans interpret probabilities through a weighted function. Quantal response equilibrium assumes that humans more frequently choose most beneficial actions by following a stochastic choice probability model influenced by some noise in the decision making process. Milind Tambe et al. [109][153][76][179] are deploying practical systems for security resource allocation purposes by modeling and analyzing Bayesian Stackelberg Games, an Stackelberg game in which the actions of both players follow a probability distribution, by using both previously described theories.

Another important challenge is the availability of real-time online information. Five ongoing trends became more and more accomplished in the history of computing: ubiquity, interconnection, intelligence, delegation and human-orientation [177]. The current wave of this progress is marked by the widespread availability of online real-time data which open new possibilities in several application areas like intelligent road transport systems. The most challenging applications are those where autonomous agents have access to online real-time data and create plans how to achieve their goals in an environment where they jointly utilize resources that become more costly as more agents use them. In these applications agents are dynamically arriving and departing when they complete their plans which are created by exploiting online data that describes the current status and the current cost of the resources. There is uncertainty about the feasible decision of an agent, because the cost of the resources will change by the time the agent reaches that point in its plan when it starts to use them: departing agents will release the resources as they complete their plans, agents simultaneously creating their plans will influence each other's costs, and agents arriving later may also influence the costs of the resources used by agents already executing their plans.

The algorithmic game theory investigations of the routing game revealed important basic properties, however the following assumptions of the algorithmic game theory approach do not handle the online information environment of real-time data based navigation systems where the agents autonomously self-adapt:

1. the throughput characteristic of the network does not change with time and the drivers can compute or learn this characteristic by repeatedly passing the road network;

2. the drivers simultaneously and selfishly decide their optimal route; and

3. the outcome travel time for a given driver depends on the choice of the other drivers and the characteristic of the network, but not on the schedule

of the trip of the drivers.

In the case of autonomous self-adapting routing problem

1. the throughput characteristic of the network changes with time and the drivers cannot compute or learn this characteristic by repeatedly passing the road network, because there may be an accident on a road and the road becomes susceptible to congestion, and then later when the accident is cleared the road is less susceptible to congestion;

2. the drivers do not decide their route at the same time simultaneously, because drivers continuously enter the road network and decide their optimal route only when they enter the road network and the decision is based on the live information about the current situation of the road network; and

3. the outcome travel time for a given driver depends not only on the current characteristic of the network and the route choice of the other drivers currently entering the road network, but also on the trip schedule of other drivers that entered the network previously, are currently entering the network or will enter the network later.

Because game theory is an appropriate foundation for multi-agent systems [123], this type of applications are called as *online joint resource utilization games* [162]. Note that these games are different from resource allocation or minority games [60] which are simultaneous one shot or repeated simultaneous games where there might be some coordination among some of the agents, while online joint resource utilization games are continuous and non-cooperative games exploiting real-time online data.

A well-known online joint resource utilization game is *real-time data based navigation.* In real-time data based navigation traffic participants generate and use online traffic information to create a self-optimized plan for their route which contain road sections jointly utilized with other agents. Two well-known real-time data based navigation systems are Google Maps and Waze. Real-time data based navigation is a special case of online joint resource utilization games, because the possible order of the resources in the plan of the agents is restricted by the structure of the road network. From theoretical point of view we call real-time data based navigation applications as *online routing games* [162] which need to be investigated in the future.

Although it is widely believed and intuitively we might think that traffic route planning results in shorter travel time if we take into account the current traffic information (for example congestions), but the classical theoretical studies do not have definite answer if real-time data based navigation produces globally better traffic or not. The challenge for COST ARTS is to define and analyze the model of online routing games in order to be able to determine and prove the benefits of online real-time data.

## 3.5 Actions for Autonomic RTS

The concept of equilibrium plays a crucial role in game theory based applications, and therefore they are also central for autonomic road traffic and transport systems. The following actions must be performed in ARTS:

- Identifying what concept of equilibrium is more adequate to a given traffic scenario.
- Relaxing the notion of equilibrium by approaching it to practical real traffic scenarios. For example, by means of evaluating the constraints more suited to model traffic scenarios by using the concept of bounded rationality.
- Introducing equilibrium concepts for guiding another approaches in modeling the behavior of traffic. These approaches may belong to other fields, as for example, the reinforcement learning process used for self-manage traffic light settings discussed in [53].

Another set of actions must be devoted to model the behaviors of traffic actors (car drivers, truck drivers, pedestrians, traffic operators, etc.) that might appears in Autonomic RTS.

- Developing general and specific behavior models for sets of the previously described traffic actors.
- Defining and evaluating principles and heuristics that can be attained to them.
- The study and proposal of new models based on prospect theory and quantal response equilibrium.
- Analyzing the tradeoff between addressing the behavior of the actors as a whole and addressing them individually.

There are very few models that use cooperative games in traffic and transport settings. Therefore, the analysis of the behavior of platoon vehicles should also be addressed by the following actions:

- Proposing cooperative games for describing them.
- Proposing new incentives for traffic actors to form a coalition.
- Evaluating the tradeoffs between the behaviors several different platoon of traffic actors for analyse the behavior of traffic.

Information and communication technologies allow modern navigation systems to use live online information about the current status of traffic flow in order to optimize the route planning of autonomic vehicles and make the traffic as a whole self-adapt to the current situation. Several authors investigated with simulation tools [164] how the traffic would behave if the majority of autonomic vehicles based their route planning on such navigation systems and concluded that online data has to be used carefully in traffic scenarios. In order to be able to measure and prove properties of autonomous self-adapting traffic routing, the formal model of online routing games should be investigated. Based on the formal model, we have to analyse routing scenarios and discover the different aspects of the benefit of online data. We have to define how we can measure the benefit of online data. We foresee different classes of online routing games. We have to find formal methods to prove the properties of online routing games.

The novel model of online routing games should be studied further in order to be able to exploit online data in autonomous self-adapting navigation systems. These should be addressed by the following actions:

- The research in the field of online routing games should identify the major features that currently produce undesirable phenomena.

- The research should focus on new online routing game decision strategies that eliminate the undesirable phenomena and have benefits of online data, or the research should prove that it is not possible to develop such strategies.

- If online routing game decision strategies that have benefits of online data are possible, then we expect that the application of these new strategies will be individually rational choice and the decision strategies can be implemented in the navigation devices themselves instead of the centralized planning approaches, because some users are reluctant to provide private data for the centralized approach.

# Bibliography

[1] Cooperative vehicle infrastructure systems. ist-2004-027293 2006-2010. Co-Founded by EC Sixth Framework program. http://www.cvisproject.org.

[2] Drive c2x - accelerate cooperative mobility. Co-Founded by the European Commission DG CONNECT in the FP7. http://www.drive-2x.eu.

[3] Agnar Aamodt and Enric Plaza. Case based reasoning: Foundational issues, methodological variations and system approaches. *AI Communications*, 7(1):39–59, 1994.

[4] Sherif Abdelwahed and Nagarajan Kandasamy. *Autonomic Computing: Concepts, Infrastructure, and Applications*, chapter A Control-Based Approach to Autonomic Performance Management in Computing Systems. CRC Press, 2007.

[5] B. Abdulhai, R. Pringle, and G Karakoulas. Reinforcement learning for the true adaptive traffic signal control. *Journal of Transportation Engineering*, 129(3):278–285, May/June 2003.

[6] John R. Anderson. *The Architecture of Cognition*. Harvard University Press, Cambridge, Mass, 1983.

[7] Sahar Araghi, Abbas Khosravi, and Douglas Creighton. A review on computational intelligence methods for controlling traffic signal timing. *Expert Systems with Applications*, 42(3):1538 – 1550, 2015.

[8] Michael Arens and Hans-Hellmut Nagel. Representation of behavioral knowledge for planning and plan-recognition in a cognitive vision system. In *KI 2002: Advances in Artificial Intelligence*, pages 268–282. Springer, 2002.

[9] W Brian Arthur. Inductive reasoning and bounded rationality. *The American economic review*, pages 406–411, 1994.

[10] P. Aschwanden, V. Baskaran, S. Bernardini, M. Moreno C. Fry, N. Muscettola, C. Plaunt, D. Rijsman, and P. Tompkins. Model-unified planning and execution for distributed autonomous system control. In *Workshop*

*on Spacecraft Autonomy: Using AI to Expand Human Space Exploration.* AAAI Press, 2006.

[11] Michael Balmer, Marcel Rieser, Konrad Meister, David Charypar, Nicolas Lefebvre, and Kai Nagel. Matsim-t: Architecture and simulation times. In Ana L. C. Bazzan and Franziska Klugl, editors, *Multi-Agent Systems for Traffic and Transportation Engineering.* 2009.

[12] Ana L. Bazzan. A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multi-Agent Systems*, 10(1):131–164, 2005.

[13] Ana L.C. Bazzan and Franziska Klugl. A review of agent-based tehcnology for traffic and transportation. *The Knowledge Engineering Rview.*

[14] Amna Bouhana, Afef Fekih, Mourad Abed, and Habib Chabchoub. An integrated case-based reasoning approach for personalized itinerary search in multimodal transportation systems. *Transportation Research Part C*, 31:30–50, 2013.

[15] Simon Box and Ben Waterson. An automated signalized junction controller that learns strategies from a human expert. *Engineering Applications of Artificial Intelligence*, 25(1):107–118, 2012.

[16] Priv-Doz Dr D Braess. Über ein paradoxon aus der verkehrsplanung. *Unternehmensforschung*, 12(1):258–268, 1968.

[17] George Brancovici. Towards trustworthy intelligence on the road: A flexible architecture for safe, adaptive, autonomous applications. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 4230–4237. IEEE, 2007.

[18] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[19] Larry Bull. Learning classifier systems: A brief introduction. In *Applications of Learning Classifier Systems.* Springer, 2004.

[20] Way L. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

[21] L. Busoniu, B. De Schutter, and R. Babuska. Learning and coordination in dynamic multiagent systems. Technical Report 05-019, Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands, October 2005.

[22] David W. Bustard and Roy Sterritt. *Autonomic Computing: Concepts, Infrastructure, and Applications*, chapter A Requirements Engineering Perspective on Autonomic Systems Development. CRC Press, 2007.

[23] Eduardo Camponogara and Werner Kraus. Distributed learning agents in urban traffic control. In *Portuguese Conference on Artificial Intelligence (EPIA)*, pages 324–335, 2003.

[24] G.E. Cantarella, G. Pavone, and A. Vitetta. Heuristics for urban road network design: Lane layout and signal settings. *European Journal of Operational Research*, 175(3):1682–1695, 2006.

[25] Enrique Castillo, José María Menéndez, and Santos Sánchez-Cambronero. Predicting traffic flow using bayesian networks. *Transportation Research Part B: Methodological*, 42(5):482 – 509, 2008.

[26] Luis Castillo, Eva Armengol, Eva Onaindía, Laura Sebastiá, Jesús González-Boticario, Antonio Rodríguez, Susana Fernández, Juan David Arias, and Daniel Borrajo. SAMAP. A user-oriented adaptive system for planning tourist visits. *Expert Systems with Applications*, 34(2):1318–1332, February 2008. ISSN: 0957-4174.

[27] Isabel Cenamor, Tomás de la Rosa, and Daniel Borrajo. Ondroad planner: Building tourist plans using traveling social network information. In *Proceedings of Conference on Human Computation & Crowdsourcing (HCOMP'13). Works-in-Progress & Demonstrations*, Palm Springs (EEUU), 2013.

[28] A. Cesta, G. Cortellessa, S. Fratini, and A. Oddi. Developing an End-to-End Planning Application from a Timeline Representation Framework. In *IAAI-09. Proceedings of the 21$^{st}$ Innovative Applications of Artificial Intelligence Conference, Pasadena, CA, USA*, 2009.

[29] Chenyi Chen, Yin Wang, Li Li, Jianming Hu, and Zuo Zhang. The retrieval of intra-day trend and its influence on traffic prediction. *Transportation Research Part C*, 22:103–118, 2012.

[30] S. Chen and W. Wang. Decision tree learning for freeway automatic incident detection. *Expert Systems with Applications*, 36(2):4101–4105, 2009.

[31] S. Chen, W. Wang, and H. van Zuylen. Construct support vector machine ensemble to detect incident. *Expert Systems with Applications*, 36:10976–10986, 2009.

[32] R.L. Cheu and D. Srinivasan T.E. Teh. Support vector machine models for freeway incident detection. In *Proceedings of Intelligent Transportation Systems*, volume 1, pages 238–243, 2003.

[33] Stephen Chiu and Sujeet Chand. Self-organizing traffic control via fuzzy logic. In *Decision and Control, 1993., Proceedings of the 32nd IEEE Conference on*, pages 1897–1902. IEEE, 1993.

[34] R. Chrobok, O. Kaumann, J. Wahle, and M. Schreckenberg. Different methods of traffic forecast based on real data. *European Journal of Operational Research*, 155(3):558–568, 2004.

[35] Brian J. Coltin and Manuela Veloso. Towards ridesharing with passenger transfers. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 1299–1300, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[36] Angel Corberán, Elena Fernández, Manuel Laguna, and Rafael Marti. Heuristic solutions to the problem of routing school buses with multiple objectives. *Journal of the Operational Research Society*, 53(4):427–435, 2002.

[37] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, pages 273–297, 1995.

[38] Rita Cucchiara, Massimo Piccardi, and Paola Mello. Image analysis and rule-based reasoning for a traffic monitoring system. *Intelligent Transportation Systems, IEEE Transactions on*, 1(2):119–130, 2000.

[39] Denise de Oliveira, Ana L. C. Bazzan, Bruno Castro da Silva, Eduardo W. Basso, and Luís Nunes. Reinforcement learning based control of traffic lights in non-stationary environments: A case study in a microscopic simulator. In *EUMAS*, 2006.

[40] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[41] Yixin Diao, Joseph L. Hellerstein, Sujay Parekh, Rean Griffith, Gail Kaiser, and Dan Phung. Self-managing systems: A control theory foundation. In *ECBS '05: Proceedings of the 12th IEEE International Conference and Workshops on Engineering of Computer-Based Systems*, pages 441–448, Washington, DC, USA, 2005. IEEE Computer Society.

[42] Francois Dion and Bruce Hellinga. A rule-based real-time traffic responsive signal control system with transit priority: application to an isolated intersection. *Transportation Research Part B: Methodological*, 36(4):325–343, 2002.

[43] Simon Dobson, Spyros Denazis, Antonio Fernández, Dominique Gaïti, Erol Gelenbe, Fabio Massacci, Paddy Nixon, Fabrice Saffre, Nikita Schmidt, and Franco Zambonelli. A survey of autonomic communications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(2):223–259, 2006.

[44] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *Computational Intelligence Magazine, IEEE*, 1(4):28–39, 2006.

[45] Jim Dowling. *The Decentralised Coordination of Self-Adaptive Components for Autonomic Distributed Systems.* PhD thesis, Trinity College Dublin, 2005.

[46] Jim Dowling, Raymond Cunningham, Eoin Curran, and Vinny Cahill. Building autonomic systems using collaborative reinforcement learning. *Knowledge Engineering Review*, 21(3):231–238, 2006.

[47] Jürgen Dunkel, Alberto Fernández, Rubén Ortiz, and Sascha Ossowski. Event-driven architecture for decision support in traffic management systems. *Expert Systems with Applications*, 38:6530–6539, 2011.

[48] Ivana Dusparic and Vinny Cahill. Using reinforcement learning for multi-policy optimization in decentralized autonomic systems - an experimental evaluation. In Wolfgang Reif, Guojun Wang, and Jadwiga Indulska, editors, *Proceedings of the 6th International Conference on Autonomic and Trusted Computing*, volume 5586 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 2009.

[49] Ivana Dusparic and Vinny Cahill. Autonomic multi-policy optimization in pervasive systems: Overview and evaluation. *ACM Transactions on Autonomous and Adaptive Systems*, 7(1), 2012.

[50] S. Dzeroski, N. Jacobs, M. Molina, C. Moure, S. Muggleton, and W. Van Laer. Detecting traffic problems with ilp. In *Proceedings of the 8th International Conference on Inductive Logic Programming*, volume 1446 of *Lecture notes in Artificial Intelligence*, pages 281–290. Springer, 1998.

[51] David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world.* Cambridge University Press, 2010.

[52] M. Ehrgott and X. Gandibleux. *Multiple Criteria Optimization. State of the art annotated bibliographic surveys.* Kluwer Academic, Dordrecht, 2002.

[53] Samah El-Tantawy, Baher Abdulhai, and Hossam Abdelgawad. Multi-agent reinforcement learning for integrated network of adaptive traffic signal controllers marlin-atsc: Methodology and large-scale application on downtown toronto. *EEE Transactions on Intelligent Transportation Systems*, 14(3), 2013.

[54] Kutluhan Erol, James Hendler, and Dana S Nau. Htn planning: Complexity and expressivity. In *AAAI*, volume 94, pages 1123–1128, 1994.

[55] Dave Ferguson, Thomas Howard, and Maxim Likhachev. Motion planning in urban environments. *Journal of Field Robotics*, 25(11-12):939–960, 2008.

[56] N. Findler and J. Stapp. A distributed approach to optimized control of street traffic signals. *Journal of Transportation Engineering*, 118(1):99–110, 1992.

[57] E. Fix and J. L. Hodges. Discriminatory analysis, nonparametric discrimination. Technical Report Project 21-49-004, Report 4, Contract AF41(128)-31, USAF School of Aviation Medicine, Randolph Field, Texas, February 1951.

[58] Gary W. Flake. *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation.* The MIT Press, January 2000.

[59] Syed Md Galib and Irene Moser. Road traffic optimisation using an evolutionary game. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, pages 519–526. ACM, 2011.

[60] Aram Galstyan, Shashikiran Kolar, and Kristina Lerman. Resource allocation games with changing resource capacities. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 145–152. ACM, 2003.

[61] Javier García, José E. Flórez, Álvaro Torralba, Daniel Borrajo, Carlos Linares López, Ángel García-Olaya, and Juan Sáenz. Combining linear programming and automated planning to solve multimodal transportation problems. *European Journal of Operations Research*, 227:216–226, 2013.

[62] Javier García, Álvaro Torralba, Ángel García-Olaya, Daniel Borrajo, and José E. Flórez. Solving multi-modal and uni-modal transportation problems through timiplan. In *Proceedings of the IFAC Symposium on Control in Transportation Systems*, pages 231–236, Sofia (Bulgaria), September 2012.

[63] J. Garcíia-Nieto, E. Alba, and A. Carolina Olivera. Swarm intelligence for traffic light scheduling: Application to real urban areas. *Engineering Applications of Artificial Intelligence*, 25(2):274–283, 2012.

[64] Zong Woo Geem, Joong Hoon Kim, and GV Loganathan. A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2):60–68, 2001.

[65] Zong Woo Geem, Joong Hoon Kim, and GV Loganathan. A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2):60–68, 2001.

[66] Michael P. Georgeff and Amy L. Lansky. Reactive reasoning and planning. In *Proceedings of AAAI-87 Sixth National Conference on Artificial Intelligence*, pages 677–682, Seattle, WA (USA), July 1987.

[67] M. Ghallab and H. Laruelle. Representation and control in IxTeT, a temporal planner. In *Proceedings of the 2nd International Conference on AI Planning Systems*, 1994.

[68] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning. Theory & Practice.* Morgan Kaufmann, 2004.

[69] César Guzmán, Vidal Alcázar, David Prior, Eva Onaindía, Daniel Borrajo, Juan Fdez-Olivares, and Ezequiel Quintero. Pelea: a domain-independent architecture for planning, execution and learning. In *Proceedings of ICAPS'12 Scheduling and Planning Applications woRKshop (SPARK)*, pages 38–45, Atibaia (Brazil), 2012. AAAI Press.

[70] Karen Zita Haigh and Manuela M. Veloso. Route planning by analogy. In *Case-Based Reasoning Research and Development, Proceedings of ICCBR-95*, pages 169–180. Springer-Verlag, October 1995.

[71] Frederick Hayes-Roth. Rule-based systems. *Communications of the ACM*, 28(9):921–932, 1985.

[72] Wesam Herbawi and Michael Weber. Evolutionary multiobjective route planning in dynamic multi-hop ridesharing. In Peter Merz and Jin-Kao Hao, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 6622 of *Lecture Notes in Computer Science*, pages 84–95. Springer Berlin Heidelberg, 2011.

[73] R. Hoar, J. Penner, and C. Jacob. Evolutionary swarm traffic: if ant roads had traffic lights. In *CEC '02: Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress*, pages 1910–1915, Washington, DC, USA, 2002. IEEE Computer Society.

[74] John H Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[75] Yaron Hollander and Joseph N Prashker. The applicability of non-cooperative game theory in transport analysis. *Transportation*, 33(5):481–496, 2006.

[76] Manish Jain, Dmytro Korzhyk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, and Milind Tambe. A double oracle algorithm for zero-sum security games on graphs. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 327–334. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[77] Sergio Jiménez, Tomás de la Rosa, Susana Fernández, Fernando Fernández, and Daniel Borrajo. A review of machine learning for automated planning. *The Knowledge Engineering Review*, 27(4):433–467, December 2012.

[78] O. L. Junior and U. Nunes. Improving the generalization properties of neural networks: An application to vehicle detection. In *Proceedings of*

*the IEEE Conference on Intelligent Transportation Systems*, pages 310–315, October 2008.

[79] Leslie Pack Kaebling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *International Journal of Artificial Intelligence Research*, pages 237–285, 1996.

[80] Daniel Kahneman. Maps of bounded rationality: Psychology for behavioral economics. *American economic review*, pages 1449–1475, 2003.

[81] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. *Econometrica: Journal of the Econometric Society*, pages 263–291, 1979.

[82] Habib M Kammoun, Ilhem Kallel, Jorge Casillas, Ajith Abraham, and Adel M Alimi. Adapt-traf: An adaptive multiagent road traffic management system based on hybrid ant-hierarchical fuzzy model. *Transportation Research Part C: Emerging Technologies*, 42:147–167, 2014.

[83] Arne Kesting, Martin Treiber, and Dirk Helbing. Agents for traffic simulation. In Adelinde M. Uhrmacher and Danny Weyns, editors, *Multi-Agent Systems: Simulation and Applications*. 2009.

[84] Dong-Hwan Kim and Doa Hoon Kim. System dynamics model for a mixed strategy game between police and driver. *System Dynamics Review*, 13(1):33–52, 1997.

[85] Donald E. Kirk. *Optimal Control Theory: An Introduction*. Dover Publications, April 2004.

[86] Janet Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 1993.

[87] John E. Laird, Paul S. Rosenbloom, and Allen Newell. Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning*, 1:11–46, 1986.

[88] Sejoon Lim, Hari Balakrishnan, David Gifford, Samuel Madden, and Daniela Rus. Stochastic motion planning and applications to traffic. In Gregory S. Chirikjian et al., editor, *Algorithmic Foundation of Robotics VIII*, volume 57 of *Springer Tracts in Advanced Robotics*, pages 483–500. Springer, 2009.

[89] Michael L. Littman, Nishkam Ravi, Eitan Fenson, and Rich Howard. Reinforcement learning for autonomic network repair. In *ICAC '04: Proceedings of the First International Conference on Autonomic Computing*, pages 284–285, Washington, DC, USA, 2004. IEEE Computer Society.

[90] Zhiyong Liu. A survey of intelligence methods in urban traffic signal control. *IJCSNS International Journal of Computer Science and Network Security*, 7(7):105–112, 2007.

[91] S. P. Lloyd. Least squares quantization in *pcm*. Unpublished Bell Laboratories Technical Note. Portions presented at the Institute of Mathematical Statistics Meeting Atlantic City, New Jersey, September 1957. Published in the March 1982 special issue on quantization of the IEEE Transactions on Information Theory, 1957.

[92] Chao Lu and Haibo Chen. Hierarchical planning for agent-based traffic management and control. *Control in Transportation Systems*, 13(1):256–261, 2012.

[93] Jian Lu, Shuyan Chen, Wei Wang, and Bin Ran. Automatic traffic incident detection based on nfoil. *Expert Systems with Applications*, 39:6547–6556, 2012.

[94] Rafael Marti, Manuel Laguna, and Fred Glover. Principles of scatter search. *European Journal of Operational Research*, 169(2):359–372, 2006.

[95] Julie A. McCann and Markus C. Huebscher. Evaluation issues in autonomic computing. In *Grid and Cooperative Computing Workshops*, pages 597–608, 2004.

[96] T. L. McCluskey, J. M. Porteous, Y. Naik, C. N. Taylor, and S. Jones. A requirements capture method and its use in an air traffic control application. *Software - Practice and Experience*, 25(1):47–71, 1995.

[97] C. McGann, F. Py, K. Rajan, H. Thomas, R. Henthorn, and R. McEwen. T-REX: A Deliberative System for AUV Control. In *3rd Workshop on Planning and Plan Execution for Real-World Systems, ICAPS*, Providence, RI, 2007.

[98] Richard D McKelvey and Thomas R Palfrey. Quantal response equilibria for normal form games. *Games and economic behavior*, 10(1):6–38, 1995.

[99] Sadayoshi Mikami and Yukinori Kakazu. Genetic reinforcement learning for cooperative traffic signal control. In *International Conference on Evolutionary Computation*, pages 223–228, 1994.

[100] Melanie Mitchell. *An introduction to genetic algorithms.* MIT press, 1998.

[101] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[102] Tom M. Mitchell, John Allen, Prasad Chalasani, John Cheng, Oren Etzioni, Marc Ringuette, and Jeffrey C. Schlimmer. Theo: A framework for self-improving systems. In K. VanLehn, editor, *Architectures for Intelligence*. Erlbaum, Hillsdale, NJ, 1990.

[103] Matthew Molineaux, Matthew Klenk, and David W. Aha. Goal-driven autonomy in a navy strategy simulation. In Maria Fox and David Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2010.

[104] Dana S Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J William Murdock, Dan Wu, and Fusun Yaman. Shop2: An htn planning system. *J. Artif. Intell. Res.(JAIR)*, 20:379–404, 2003.

[105] J. Niittymaki and M. Pursula. Signal control using fuzzy logic. *Fuzzy Sets Systems*, 116(1):11–22, 2000.

[106] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*. Cambridge University Press, 2007.

[107] C. P. Pappis and Ebrahim H. Mamdani. A fuzzy logic controller for a traffic junction. *IEEE Transactions on Systems, Man and Cybernetics*, 7(10):707–717, Oct 1977.

[108] Byungkyu " Brian" Park, Carroll J Messer, and Thomas Urbanik II. Enhanced genetic algorithm for signal-timing optimization of oversaturated intersections. *Transportation Research Record: Journal of the Transportation Research Board*, 1727(1):32–41, 2000.

[109] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Playing games for security: an efficient exact algorithm for solving bayesian stackelberg games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 895–902. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[110] Michel Pasquier, Chai Quek, and Mary Toh. Fuzzylot: a novel self-organising fuzzy-neural rule-based pilot system for automated vehicles. *Neural Networks*, 14(8):1099–1112, 2001.

[111] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[112] Pål Andreas Pedersen. Moral hazard in traffic games. *Journal of Transport Economics and Policy (JTEP)*, 37(1):47–68, 2003.

[113] Julien Perez, Cecile Germain-Renaud, Balazs Kegl, and Charles Loomis. Grid differentiated services: A reinforcement learning approach. In *CCGRID '08: Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid*, pages 287–294, Washington, DC, USA, 2008. IEEE Computer Society.

[114] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

[115] L.A. Prashanth and S. Bhatnagar. Reinforcement learning with function approximation for traffic signal control. *Intelligent Transportation Systems, IEEE Transactions on*, 12(2):412–421, June 2011.

[116] L.A. Prashanth and Bhatnagar Shalabh. Reinforcement learning with function approximation for traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):412–421, 2011.

[117] Holger Prothmann, Fabian Rochner, Sven Tomforde, Jürgen Branke, Christian Müller-Schloer, and Hartmut Schmeck. Organic control of traffic lights. In *ATC '08: Proceedings of the 5th international conference on Autonomic and Trusted Computing*, pages 219–233, Berlin, Heidelberg, 2008. Springer-Verlag.

[118] Angel Puerta, John Egar, Samson Tu, and Mark Musen. A multiple-method knowledge-acquisition shell for the automatic generation of knowledge-acquisition tools. *Knowledge Acquisition*, 4:171–196, 1992.

[119] Chai Quek, Michel Pasquier, and B Lim. A novel self-organizing fuzzy rule-based system for modelling traffic flow behaviour. *Expert Systems with applications*, 36(10):12167–12178, 2009.

[120] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, August 1990.

[121] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[122] Silvia Richter. Learning traffic control - towards practical traffic control using policy gradients. Technical report, Albert-Ludwigs-Universitat Freiburg, 2006.

[123] Jeffrey S Rosenschein. Multiagent systems, and the search for appropriate foundations. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 5–6. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[124] Tim Roughgarden. Computing equilibria: a computational complexity perspective. *Economic Theory*, 42(1):193–236, 2010.

[125] Tim Roughgarden and Éva Tardos. How bad is selfish routing? *Journal of the ACM (JACM)*, 49(2):236–259, 2002.

[126] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[127] Stuart Russell and Peter Norvig. *Aritifical Intelligence - A Modern Approach.* Prentice Hall, 2003.

[128] S. Salcedo-Sanz, D. Manjarrós, A. Pastor-Sánchez, J. Del Ser, J.A. Portilla-Figueras, and S. Gil-López. One-way urban traffic reconfiguration using a multi-objective harmony search approach. *Expert Systems with Applications*, 40:3341–3350, 2013.

[129] Asad Salkham and Vinny Cahill. Soilse: A decentralized approach to optimization of fluctuating urban traffic using reinforcement learning. In *13th International IEEE Annual Conference on Intelligent Transportation Systems*, 2010.

[130] Asad Salkham, Raymond Cunningham, Anurag Garg, and Vinny Cahill. A collaborative reinforcement learning approach to urban traffic control optimization. In *International Conference on Intelligent Agent Technology*, December 2008.

[131] J. Sánchez, M. Galń, and E. Rubio. Traffic signal optimization in "la almozara" district in saragossa under congestion conditions, using genetic algorithms, traffic microsimulation, and cluster computing. *IEEE Trans. Intell. Transp. Syst.*, 11:132–141, 2010.

[132] Peter Sanders and Dominik Schultes. Engineering fast route planning algorithms. In Camil Demetrescu, editor, *Experimental Algorithms*, volume 4525 of *Lecture Notes in Computer Science*, pages 23–36. Springer Berlin Heidelberg, 2007.

[133] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[134] Guus Schreiber, Hans Akkermans, Anjo Anjewierden, Robert de Hoog, Nigel Shadbolt, Walter Van de Velde, and Bob J. Wielinga. *Knowledge Engineering and Management: The CommonKADS Methodology.* MIT Press, Cambridge, Mass., 2nd ed. edition, 1999.

[135] Sandip Sen and Gerhard Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, chapter Learning in Multiagent Systems, pages 259–298. MIT Press, Cambridge, MA, USA, 1999.

[136] Mohammad M. Shah, Lukas Chrpa, Diane Kitchin, Thomas L. McCluskey, and Mauro Vallati. Exploring knowledge engineering strategies in designing and modelling a road traffic accident management domain. In *Proceedings of IJCAI*, 2013.

[137] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations.* Cambridge University Press, 2008.

[138] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations.* Cambridge University Press, 2008.

[139] E.H. Shortliffe. *MYCIN: A rule-based computer program for advising physicians regarding antimicrobial therapy selection.* PhD thesis, Stanford University, 1974.

[140] Reid Simmons. Concurrent planning and execution for autonomous robots. In *IEEE International Conference on Robotics and Automation*, pages 46–50, 1992.

[141] S. Sivaraman and M.M. Trivedi. A general active-learning framework for on-road vehicle recognition and tracking. *IEEE Transactions on Intelligent Transportation Systems*, 11(2):267–276, June 2010.

[142] MJ Smith. The existence, uniqueness and stability of traffic equilibria. *Transportation Research Part B: Methodological*, 13(4):295–304, 1979.

[143] Stephen F. Smith, Gregory J. Barlow, Xiao-Feng Xie, and Zachary B. Rubinstein. Smart urban signal networks: Initial application of the surtrac adaptive traffic signal control system. In *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling*, pages 434–442. AAAI Press, 2014.

[144] J. Spall and D. Chin. Traffic-responsive signal timing for systemwide traffic control. *Transp. Res. Part C: Emerging Technol.*, 5(3/4):153–163, 1997.

[145] Biplav Srivastava, Joseph P Bigus, and Don A Schlosnagle. Bringing planning to autonomic applications with able. In *Autonomic Computing, 2004. Proceedings. International Conference on*, pages 154–161. IEEE, 2004.

[146] Biplav Srivastava and Subbarao Kambhampati. The case for automated planning in autonomic computing. In *Proc. of the ICAC 2005. IEEE Computer*, pages 331–332. Society Press, 2005.

[147] Roy Sterritt. Autonomic computing. *Innovations in Systems and Software Engineering*, pages 79–88, 2005.

[148] Roy Sterritt, Manish Parashar, Huaglory Tianfield, and Rainer Unland. A concise introduction to autonomic computing. *Advanced Engineering Informatics*, 19(3):181–187, July 2005.

[149] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Knowledge Engineering: Principles and Methods. *Data and Knowledge Engineering*, 25(1-2):161–197, March 1998.

[150] Shiliang Sun, Changshui Zhang, Guoqiang Yu, Naijiang Lu, and Fei Xiao. Bayesian network methods for traffic flow forecasting with incomplete data. In *Proceedings of ECML 2004*, volume 3201 of *Lecture notes in Artificial Intelligence*, pages 419–428. Springer, 2004.

[151] Richard S. Suton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* A Bradford Book. The MIT Press, Cambridge, Massachusetts, 1998.

[152] Katia Sycara. Multiagent systems. *AI Magazine*, 19(2), 1998.

[153] Milind Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.

[154] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337. Morgan Kaufmann, 1993.

[155] Fitsum Teklu, Agachai Sumalee, and David Watling. A genetic algorithm approach for optimizing traffic control signals considering routing. *Computer-Aided Civil and Infrastructure Engineering*, 22(1):31–43, 2007.

[156] Dusan Teodorovic and Shinya Kikuchi. Transportation route choice model using fuzzy inference technique. In *Uncertainty Modeling and Analysis, 1990. Proceedings., First International Symposium on*, pages 140–145. IEEE, 1990.

[157] G. Tesauro. Reinforcement learning in autonomic computing: A manifesto and case studies. *IEEE Internet Computing*, 11(1):22–30, 2007.

[158] Gerald Tesauro, David M. Chess, William E. Walsh, Rajarshi Das, Alla Segal, Ian Whalley, Jeffrey O. Kephart, and Steve R. White. A multi-agent systems approach to autonomic computing. *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 464–471, 2004.

[159] Jean Tirole. *The theory of industrial organization.* MIT press, 1988.

[160] George Tsebelis. The abuse of probability in political analysis: The robinson crusoe fallacy. *The American political science review*, pages 77–91, 1989.

[161] Henk J van Zuylen and Henk Taale. Urban networks with ring roads: two-level, three-player game. *Transportation Research Record: Journal of the Transportation Research Board*, 1894(1):180–187, 2004.

[162] László Z Varga. Online routing games and the benefit of online data.

[163] Manuela Veloso, Jaime Carbonell, Alicia Pérez, Daniel Borrajo, Eugene Fink, and Jim Blythe. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical AI*, 7:81–120, 1995.

[164] Joachim Wahle, Ana Lúcia C Bazzan, Franziska Klügl, and Michael Schreckenberg. Decision dynamics in a traffic scenario. *Physica A: Statistical Mechanics and its Applications*, 287(3):669–681, 2000.

[165] Ko-Hsin Cindy Wang and Adi Botea. Tractable multi-agent path planning on grid maps. In *Proceedings of IJCAI'09*, pages 1870–1875, 2009.

[166] C. J. C. H. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3/4):279–292, May 1992.

[167] Christopher J. C. H. Watkins and Peter Dayan. Technical note: Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

[168] David Watling. User equilibrium traffic network assignment with stochastic travel times and late arrival penalty. *European journal of operational research*, 175(3):1539–1556, 2006.

[169] W. Wei and Y. Zhang. Fl-fn based traffic signal control. In *Proceedings of the 2002 IEEE International Conference on Fuzzy Systems*, pages 296–300, 2002.

[170] Gerhard Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, 1999.

[171] Gerhard Weiss, editor. *Multiagent Systems*. MIT Press, 2013.

[172] Paul J. Werbos. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[173] Michael E Wetzstein. *Microeconomic theory: Concepts and connections*. Routledge, 2013.

[174] Tom De Wolf and Tom Holvoet. Emergence and self-organisation: a statement of similarities and differences. In *Lecture Notes in Artificial Intelligence*, pages 96–110. Springer Verlag, 2004.

[175] D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[176] M. Wooldridge. *An Introduction to Multi-Agent Systems*. John Wiley and Sons, 2012.

[177] Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[178] Ronald R Yager and Lotfi Asker Zadeh. *An introduction to fuzzy logic applications in intelligent systems*. Springer, 1992.

[179] Rong Yang, Milind Tambe, Manish Jain, Jun-young Kwak, James Pita, and Zhengyu Yin. Game theory and human behavior: challenges in security and sustainability. In *Algorithmic Decision Theory*, pages 320–330. Springer, 2011.

[180] Zhao-sheng Yang, Xin Chen, Yang-shan Tang, and Jian-ping Sun. Intelligent cooperation control of urban traffic networks. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, pages 1482 – 1486, 2005.

[181] X. Yu and W. Recker. Stochastic adaptive control model for traffic signal systems. *Transp. Res. Part C: Emerging Technol.*, 14(4):263–282, 2006.

[182] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[183] Huiyu Zhou and Kotaro Hirasawa. Traffic conduction analysis model with time series rule mining. *Expert Systems with Applications*, 41(14):6524 – 6535, 2014.

[184] Terry Zimmerman and Subbarao Kambhampati. Learning-assisted automated planning: Looking back, taking stock, going forward. *AI Magazine*, 24(2):73–96, Summer 2003.