AI Planning-Based Service Modeling for the Internet of Things

Quentin Bahers
Abstract

AI Planning-Based Service Modeling for the Internet of Things

Quentin Bahers

It is estimated that by 2020, more than 50 billion devices will be interconnected, to form what is called the Internet of Things. Those devices range from consumer electronics to utility meters, including vehicles. Provided with sensory capabilities, those objects will be able to transmit valuable information about their environment, not only to humans, but even more importantly to other machines, which should ultimately be able to interpret and take decisions based on the information received. This “smartness” implies gifting those devices with a certain degree of automation. This Master’s Thesis investigates how recent advances in artificial intelligence planning can be helpful in building such systems. In particular, an artificial intelligence planner able to generate workflows for most of IoT-related use cases has been connected to an IoT platform. A performance study of a state-of-the planner, Fast Downward, on one of the most challenging IoT application, Smart Garbage Collection (which is similar to the Traveling Salesman Problem) has also been carried out. Eventually, different pre-processing and clustering techniques are suggested to tackle the latest AI planners’ inefficiency on quickly finding plans for the most difficult tasks.
Contents

1 Introduction 3

1.1 Motivation and Objectives ........................................ 3

1.2 Structure of the Report ........................................... 4

2 Background Theory 6

2.1 What is Planning? .................................................. 6

2.1.1 Defining AI Planning .......................................... 6

2.1.2 Main Types of Planners ........................................ 7

2.2 Conceptual Model for Planning ................................... 7

2.2.1 State-Transition Systems ..................................... 7

2.2.2 Plans and Objectives .......................................... 8

2.3 Planning and Search .............................................. 9

2.3.1 Search Problems ................................................ 9

2.3.2 Problem Formulation ........................................... 9
3 The Internet of Things and Its Applications 11
  3.1 Definition ................................................. 11
  3.2 Use Cases .................................................. 12
    3.2.1 Main Applications ..................................... 12
    3.2.2 Other Applications .................................... 18

4 Middleware for the Internet of Things 19
  4.1 Introduction ............................................... 19
  4.2 IPSO Guidelines ............................................ 20
  4.3 IoT-Framework ............................................. 21

5 First Step: Rule Engines 22
  5.1 Introduction ............................................... 22
  5.2 Motivating Examples ....................................... 23
  5.3 IFTTT-like Applications ................................... 23
  5.4 Limitations ................................................ 24

6 Second Step: Domain Specific Planning 25
  6.1 SHOP2 ..................................................... 25
  6.2 Implementation ............................................ 25
    6.2.1 Architecture Overview ................................. 25
    6.2.2 About the Planner ..................................... 27
  6.3 Limitations ................................................ 27
7 Third Step: Domain Independent Planning

7.1 Motivating Examples

7.1.1 Smart Waste Management

7.1.2 Agricultural Drones Route Optimization

7.2 Heuristic Planning

7.2.1 International Planning Competition

7.2.2 Fast Downward Stone Soup & LAMA

7.2.3 Arvand & ArvandHerd

7.3 Experiments

7.3.1 Problem Description

7.3.2 Problem Modeling

7.3.3 Planner Used

7.3.4 Results

7.4 Limitations

8 Scalability

8.1 Scaling with Preprocessing

8.2 Scaling with Clustering

8.2.1 Kd Tree Space Partitioning Approach

8.2.2 Minimum Spanning Tree Approach

8.2.3 Computer Networks Analogy
9 Conclusion

9.1 Summary of Master’s Thesis Achievements . . . . . . . . . . . . . . . 40

9.2 Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . 40

A Waste Management: Domain File 42

B Waste Management: Problem File 44

C Waste Management: Plan Produced 48

Bibliography 49
List of Figures

6.1 Architecture .................................................. 26

7.1 Impact of the number of bins on the search time ..................... 35

7.2 Impact of the number of bins on the peak memory .................... 35

List of Tables

7.1 Evolution of the peak memory and total time in relation to the num-
ber of bins .................................................... 34
Acronyms

AI Artificial Intelligence. 2–4, 6, 22–24, 26–30, 32, 33, 36–39, 41, 42
HTN Hierarchical Task Network. 4, 25, 27, 28, 41, 42
IoT Internet of Things. 2–4, 11, 18, 22–24, 42
IPC International Planning Competition. 25, 30, 31, 34
IPSO IP for Smart Objects. 19, 20, 42
IPv6 Internet Protocol version 6. 12
M2M Machine to Machine. 23
MOOC Massive Open Online Course. 6, 7
NFC Near Field Communication. 18
PDDL Planning Domain Definition Language. 4, 33
QR code Quick Response Code. 18
REST Representational State Transfer. 20
RFID Radio-frequency identification. 11, 18, 19
URL Uniform Resource Locator. 20
Chapter 1

Introduction

I conducted my thesis at Ericsson, one of the world leaders in the field of communications technology [1]. Since the Internet of Things (IoT) is seen by many experts as the next upcoming revolution after the advent of the Internet and Mobile Networks, Ericsson is putting a lot of effort into it, its vision being to build a “Networked Society”, where people and objects are interconnected. Its full potential will be reached when this interconnection will come with some sort of intelligence, so that our interaction with objects becomes more powerful, hence the need to explore how Artificial Intelligence (AI) and in particular AI planning, can help in achieving this goal.

1.1 Motivation and Objectives

This master’s thesis aims at investigating how AI planning techniques can be used in the context of IoT to generate workflows given a set of tasks to achieve.

An example of a task to perform could be waste collection in a smart city. Sensors inside bins would indicate their fill level and send this information to an AI planner which would produce an optimized path for collecting full bins.

The Internet of Things is a vision where every object, including ovens and carpets, will be connected to the Internet. This will push further human-machine communication, as well as machine to machine communication. For instance, one could be notified by its umbrella before leaving home that it is likely to rain today.
Chapter 1. Introduction

In the state of the art, there has been some work done regarding how to apply AI planning algorithms to Web services, but in our knowledge there is little to no work done when it comes to using them in the context of IoT, which is unique in that it mainly deals with (a large number of) sensors.

1.2 Structure of the Report

The rest of the report is structured as follows:

Chapter 2 provides some background information about AI planning. More specifically, we give a definition of AI planning, plans and objectives. We also explain how to model a search problem and briefly talk about the different types of planners to solve them.

Chapter 3 defines what is meant by the “Internet of Things”. It also details numerous use cases related to IoT, so that later on in the report, we can look at which ones could benefit from AI planning.

Chapter 4 explains why a middleware for the Internet of Things is needed and how the IoT-Framework is a solution to this issue.

The next three chapters explore three different solutions to solve some representative IoT use cases depending on their complexity.

Chapter 5 details what are the existing solutions for easy-to-solve use cases.

Chapter 6 presents my implementation of a Hierarchical Task Network (HTN) AI planner that I connected to the IoT-Framework, which is able to solve most of the typical IoT use cases.

Chapter 7 evaluates the performance of one of the state-of-the-art AI planner, Fast Downward, on a difficult-to-solve IoT use case (waste collection) and shows its limitations.

Chapter 8 provides some hints on how to overcome state-of-the-art AI planners scaling issues for some real world use cases, such as waste collection, using pre-processing and clustering techniques.
Chapter 9 concludes the report and gives some directions on how this Master’s Thesis could be used as a starting point for future work.

In addition, some additional information can be found in the following three appendices:

Appendix A provides the code used to define the waste collection problem (in the Planning Domain Definition Language (PDDL) language).

Appendix B displays the code specifying the initial state, the goal state, as well as the numerical values specifying the waste collection problem.

Appendix C shows the plan produced by the Fast Downward planner, when provided the files available in Appendix A and Appendix B.
Chapter 2

Background Theory

2.1 What is Planning?

2.1.1 Defining AI Planning

Here are some definitions on planning found in literature to give you an idea of what planning is about:

In “Artificial Intelligence, A Modern Approach” (second edition) by Stuart Russel and Peter Norvig, one of the reference textbooks in the field of artificial intelligence, and in automated planning in particular, planning is concisely defined as “the task of coming up with a sequence of actions that will achieve a goal” [2].

Austin Tate, in “The MIT Encyclopedia of the Cognitive Sciences”, defines planning as follows: “Planning is the process of generating (possibly partial) representations of future behaviour prior to the use of such plans to constrain or control that behavior. The outcome is usually a set of actions, with temporal and other constraints on them, for execution by some agent or agents. As a core aspect of human intelligence, planning has been studied since the earliest days of AI and cognitive science. Planning research has led to many useful tools for real-world applications, and has yielded significant insights into the organization of behavior and the nature of reasoning about actions” [3].

In their “Artificial Intelligence Planning” Massive Open Online Course (MOOC)
available on Coursera, Dr. Gerhard Wickler and Prof. Austin Tate defined planning as being "an explicit deliberation process that chooses and organizes actions by anticipating their outcomes, which aims at achieving some pre-stated objectives" [4].

AI (Artificial Intelligence) planning is the tentative to understand the human process of planning, to be able to define a theoretical model describing it, and later on to transpose this model to a computational model.

The end goal to the study of AI planning is to be able to develop autonomous intelligent machines, able to achieve tasks by themselves, without any human intervention, depending on their surrounding environment.

2.1.2 Main Types of Planners

There are essentially two types of planners: domain-specific planners and domain-independent planners. On the one hand, domain-specific planners use specific knowledge about the problem to solve to find a solution efficiently. They are especially tuned for a specific problem. This implies that if the problem to solve is different and unrelated to the planning domain, those kind of planners may not be able to find a solution, or at least are likely to be very inefficient at finding one [5]. In general, domain-specific planning is a good option if we are looking for very efficient solutions. Domain-independent planners, on the other hand, only use generic techniques, without any knowledge of the planning domain. The advantage is that you do not have to write a new planner for every problem, as it is supposed to work for any planning domain. The disadvantage is that it is in theory a bit slower than domain-specific planning (however, this is not always the case in practice).

2.2 Conceptual Model for Planning

2.2.1 State-Transition Systems

A state-transition system is a 4-tuple \((S, A, E, \gamma)\), where:

- \(S\) is a finite or infinite set of states;
Chapter 2. Background Theory

- A is a finite or infinite set of actions;
- E is a finite infinite set of events;
- $\gamma$ is a state transition function.

The first component $S$ represents the different states the system can be in. The second component $A$ is the set of actions. An action is a thing an agent can do to modify the state the system is currently in. The third component is the set of events. An event is different from an action in that it can happen in an unpredictable way, without the will of the external agent. However, similarly to actions, events also change the current state of the system. The last component $\gamma$ is the state-transition function. A state-transition function takes a state, as well as either an action or an event as an input, and returns all the possible states resulting in applying the action or event in question. A state-transition system is a useful way to synthesize the different possibilities a system can evolve in.

2.2.2 Plans and Objectives

2.2.2.1 Plans

In the “Artificial Intelligence Planning” MOOC available on Coursera, a plan is described as “a structure that gives appropriate actions to apply in order to achieve some objective when starting from a given state”. If the system is deterministic, a plan gives us a sequential schedule of actions that we have to follow to complete a task.

2.2.2.2 Objectives

There are really two ways to consider the notion of objective. The first way to see an objective is as a goal state or a set of goal states we want to end in. An example of such an objective could be “go to the beach”. Here, the goal state is different for the initial state (we could be initially at home for instance).

Another way to see an objective is as a task to be achieved. An example of such an objective could be “go on holiday”. This type of objective is quite different, since in the end, we go back in the initial state. It is just that we do something in the meantime.
2.3 Planning and Search

2.3.1 Search Problems

A search problem is characterized by:

- An initial state
- A set of possible actions/applicability conditions
- A goal
- A path cost function

The first component is the initial state. The initial state is the state where we are at when we begin our search. The second component is the set of possible actions. An action is only applicable to a subset of the world states, which is most of the time reduced to a single state. This implies that actions have to come together with applicability conditions, to know for which states they are applicable. The third component is the goal. A goal can be represented by either a single goal state or a set of goal states. A solution to a search problem is a path starting from an initial state, going through a number of intermediate states using only applicable actions, and ending in a goal state. The last component is the path cost function. The past cost function attributes a numerical value to a path, its cost. Usually, the lower the numerical value, the better.

2.3.2 Problem Formulation

2.3.2.1 Assumptions About The Environment

We make the assumption that the environment is:

- finite and discrete
- fully observable
- deterministic
- static

Making the assumption that the environment is finite and discrete means that the number of world states is finite. This is possible if the parameters governing the
environment are discrete, and not continuous, as continuous variables would generate an infinite number of world states. The second assumption is that the environment is fully observable. This simply means that the planner can see and know all there is to know about the environment. The third assumption is that the environment is deterministic. It means that each action has only one outcome. The last assumption is that the environment is static, that is to say that there are no events, but only actions.

### 2.3.2.2 Other Assumptions

The other assumptions we make are the following:

- restricted goals
- sequential plans
- implicit time
- offline planning

Making the assumption that we have restricted goals means that the search problem either has a unique goal state or a set of goal states. The second assumption is that the plans are sequential, i.e. plans are a linearly ordered sequence of actions. The third assumption is that time is implicit, which means that there are no time duration; actions are performed instantaneously, without any waiting time between them. The last assumption is that we considering that the planning process is done offline, that is to say, the planner does not know the execution status.

The type of planning for which the eight assumptions we have just listed are fulfilled is called “classical planning” [5]. Classical planning is the type of planning we will restrict to in this report.
Chapter 3

The Internet of Things and Its Applications

3.1 Definition

The Internet of Things (often shortened to IoT) is a concept where potentially all physical objects, or “things”, will be interconnected via the Internet. The “things” in question can range from cars to glasses, including grocery products, street lamps, ovens, and many more. This is made possible by embedding small electronic devices in objects, providing them with an access to the Internet. It becomes even more interesting when those embedded devices also come with some sensory capabilities, so that objects can have a partial view of their surroundings, as well as some actuation capabilities, giving them the ability to interact with their environment.

The Internet of Things term was first coined by Kevin Ashton in 1999, during a presentation he gave at Procter & Gamble, which was about applying the Radio-frequency identification (RFID) technology to their supply chain [6]. Ten years later, in an article published in the RFID journal, he described more thoroughly what he actually meant by the “Internet of Things”, and what his vision back then was:

“Today computers—and, therefore, the Internet—are almost wholly dependent on human beings for information. [...] The problem is, people have limited time, attention and accuracy—all of which means they are
not very good at capturing data about things in the real world. [...] If we had computers that knew everything there was to know about things—using data they gathered without any help from us—we would be able to track and count everything, and greatly reduce waste, loss and cost.”

It is estimated that by 2020, 50 billions devices will be connected to the Internet, more than 6 times the number of human beings on Earth [7]. One may ask, however, whether it will be possible to uniquely identify such a huge amount of connected devices. That should actually be the case, thanks to the advent of the Internet Protocol version 6 (IPv6) protocol, which can provide approximately 100 Internet addresses for every atom on the surface of the Earth [8].

There has been a lot of hype around the Internet of Things lately, and rightly so, according to Kevin Ashton, presenting it as the next revolution: “The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so.”

### 3.2 Use Cases

#### 3.2.1 Main Applications

The Internet of Things use cases revolve around six main themes [9]:

- Smart cities
- Smart environment
- Smart water
- Smart agriculture
- Domotic & home automation
- eHealth

Below are briefly described some common applications for each one of those themes. The list provided is deliberately a bit long, the purpose being to give an idea of the wide and heterogeneous range of services that become possible thanks to the Internet of Things, and how it has the potential to revolutionize our modern society in many ways.
3.2.1.1 Smart Cities

Hitachi defines a smart city as a city that “seeks to satisfy the desires and values of its residents, with the use of advanced IT to improve energy efficiency and concern for the global environment as prerequisites” [10]. This entails easing the traffic flow in city centers, reducing the electric consumption of public infrastructures and facilities, or optimizing the way we collect garbage, just to name a few examples.

Traffic Congestion

The idea is to record the amount of traffic and detect traffic jams as well as accidents in order to regulate the traffic by encouraging drivers to use less congested roads [9]. At a later stage, one can imagine this information being sent to self-driving cars so that they can decide which path to take, which will lead to a self-regulated traffic.

Smart Parking

By detecting the presence of cars parked using for instance magnetic or ultrasound sensors [11], a smart parking application could help drivers find the closest parking space available, sending this information to their smartphones or navigation systems.

Smart Street Lighting

Cities are over-illuminated at night, partly because street lamps are always turned on, whether there are cars and pedestrians on the street or not. By gathering real-time data about the current traffic, we could manage more effectively when to turn on the street lights, and when to turn them off. Doing this would help to reduce our growing energy usage. In addition, we could use those street lamps to highlight dangerous places where a traffic accident has just occurred for instance [7].

Smart Waste Management

By setting up sensors inside bins and containers to measure their fill level, we can compute an optimized path for collecting nearly full bins, and then send the route
to truck’s on board electronics [7]. One can also imagine taking into account other parameters, such as the temperature for example, and collect bins more often when it is warm outside, to help reduce smelling issues.

3.2.1.2 Smart Environment

With the growing number of people living on Earth, more and more waste is produced everyday, and natural resources are becoming increasingly scarce. Thus, having a better grasp on our planet to take better care of it appears to be even more important than before.

Forest Fire Detection

By detecting combustion gases (using CO$_2$ detection sensors for instance) and by collecting various other parameters such as vegetation humidity or wind direction, fire services can get notified when a forest fire begins, and immediately get all the information they need to plan their intervention [9].

Air Pollution

If we manage to control and forecast the CO$_2$ emissions of factories and vehicles as well as weather conditions (heat and sunlight being catalysts for the reactions that create ozone [12]), we can predict which days ozone levels will exceed the threshold after which air pollution can be noxious. To prevent it from happening, local authorities can then decide ahead of time to partially close some areas to traffic and promote the use of public transports.

Snow Level Monitoring

The idea behind a snow level monitoring system is simply to be able to know in real time snow levels on a mountain slope as well as the snowpack structure [9]. That way, we can estimate the snowpack mechanical stability and the likelihood of an avalanche to occur.
3.2. Use Cases

3.2.1.3 Smart Water

“There is a water crisis today. But the crisis is not about having too little water to satisfy our needs. It is a crisis of managing water so badly that billions of people—and the environment—suffer badly.” — World Water Vision Report [13]

Water Leakages

The idea there is simply to put pressure sensors inside the underground pipes of water supply networks so that we can immediately detect water leakages and automatically close relevant valves to avoid water waste [9].

Chemical Leakage Detection in Rivers

This one is quite self-explanatory. By systematically checking pollution levels in rivers, citizens will eventually know which factories are environmentally friendly, and which ones are not...This will also help local governments to punish them accordingly [9].

Potable Water Monitoring

A potable water monitoring system can provide a real-time monitoring of tap water in cities. Its quality could be displayed on local authorities’ websites so that citizens can feel safe to drink it or use it for cooking for instance [9].

3.2.1.4 Smart Agriculture

Cutting down forests to replace them with agricultural lands, extensively using fertilizers and pesticides, as well as irrigating without any limit is not sustainable. We need to somehow be able to increase yields to feed the planet while respecting our environment at the same time. Applying the Internet of Things to agriculture could help us achieve that goal.
Wine Quality Enhancing

By monitoring the temperature, humidity and leaf wetness in vineyards using Wireless Sensor Networks, and by fetching weather forecasts from weather stations, it is possible to predict the appearance of various plagues such as mildium, oidium and botritis in the next days thanks to statistical models [14]. That way, wine-growers can take appropriate measures to prevent or at least limit plagues’ impact, which will eventually lead to higher quality grapes.

Golf Courses

The first use case that comes to mind when thinking about smart irrigation systems is to use them in farms to augment crops productivity and reduce water consumption. However, golf courses could also be interested in this solution, to selectively irrigate the dry zones of the green, since they are one of the biggest consumers of water [15].

3.2.1.5 Domotic & Home Automation

Home automation is a topic which got a lot of attention lately. All the giant technology companies have begun to work on their own “smart home” solution: Apple, Google (which bought Nest Labs, mostly known for developing a smart thermostat, for 3.2 billion dollars), Microsoft, Samsung, to name a few [16].

Energy and Water Use

We do not always use our home appliances in the most economical way. As an example, we tend to turn on our washing machine and our water heater in the middle of the day, when the electricity price is at its highest. Instead of managing our appliances manually, we should let a home control system take smart decisions for us. It would look at when the pricing of gas, water and electricity are the cheapest and would turn on our devices accordingly. It could also manage our heating system: after analysing data such as the indoor and outdoor temperature, and checking the weather reports provided on the Internet, the home control system would decide to
trigger the heating system or not. That way, our electricity consumption would be based on what we really need. This would also help us to save money [7].

_Intrusion Detection Systems_

Intrusion detection systems can detect windows and doors break downs and alert the closest police station or send a notification to the house owner’s smartphone.

3.2.1.6 _eHealth_

In our ageing society, eHealth could play an important role in improving people’s life. According to a European Commission, among the 3.4 billion people worldwide who will own a smartphone in 2017, 50% of them will use health apps [17]. This figure is even more impressive when we think that this is without taking into account the emergence of smart watches or other wearable devices.

_Fall Detection_

If elderly or disabled people were to wear portable devices at home capable of detecting when they fall on the ground and immediately notify their neighbors or children that a problem occurred, they could feel secure to stay longer in their own home and avoid having to go to a nursing home [7].

_Patients Surveillance_

Visit the doctor can sometimes be an issue for elderly people living in remote areas or disabled people unable to drive. What’s more, those visits are most of the time routine checkups [18]. All of this could be easily avoided if they were to use devices allowing their general practitioner to remotely check their health status. A couple of companies, like Withings, have already started to commercialize some smart objects related to health, such as a smart scale [19] or a wireless blood pressure monitoring system [20]. And this is only the beginning...
3.2.2 Other Applications

Some other fields can also benefit from the Internet of Things, among which [9]:

- Smart metering
- Security & emergencies
- Retail
- Logistics
- Industrial control
- Smart animal farming

In particular, IoT could revolutionise the retail and logistics sectors. In retail, RFID tags could be put on products to ensure a better traceability. Numerous mobile applications could also emerge to assist customers in buying the product they really want: by scanning the product’s Near Field Communication (NFC) tag or Quick Response Code (QR code) they could get a lot of useful information, like the presence of substances they are allergic to or how many times they bought that product before. RFID tags could also be profitably used in the logistic sector, to help locating items in warehouses (e.g. Amazon’s warehouses) or harbours.
Chapter 4

Middleware for the Internet of Things

4.1 Introduction

One of the issues we have to face when dealing with the Internet of Things is that devices can be very heterogeneous. They can range from smartphones, RFID tags, including mere temperature sensors plugged on an Arduino. This implies the data they produce can be very different, which is an issue because smart systems won’t have any way to know a priori what the format of the data is going to be. Thus, all the data coming from those various devices should somehow be sent using the same format.

Moreover, another issue is that smart systems “often rely on the services available at other devices in the surroundings. Not only does this require selecting the most suitable services from a massive number of available services, but also dealing with massive number of results returned from different services”.

The IP for Smart Objects (IPSO) alliance guidelines for IP-based Smart Objects Systems is an attempt to fix the first issue, while the IoT-Framework aims at providing a solution to the second issue.
4.2 IPSO Guidelines

The IPSO specification defines Representational State Transfer (REST) interfaces for common types of sensors and actuators in use in smart environments, the goal being to have a generic way to interact with them. Examples of such interfaces include the “IPSO Luminosity Sensor” interface, the “IPSO Presence Sensor” interface and the “IPSO Temperature Sensor” interface, among others [22]. When the smart object in question does not belong to a specific category, the “IPSO Generic Sensor” interface can be used instead.

Every smart object type is uniquely identified by an object ID (e.g. the object ID for the “IPSO Presence Sensor” is 3302). One of the advantages of using an object ID is that you can determine the type of sensor you are dealing with just by fetching its object ID. For every smart object, multiple resources are also defined. Most of the time, a resource represents a measure, a characteristic of the object, or a parameter you can set. For instance, the resources defined for the “IPSO Temperature Sensor” are the following:

- Sensor Value (which returns the temperature in °C)
- Min Measured Value
- Max Measured Value
- Min Range Value (i.e. the minimum value that can be measured by the sensor)
- Max Range Value (i.e. the minimum value that can be measured by the sensor)

To retrieve data from a specific resource, all you need to do is a GET request at the Uniform Resource Locator (URL) following the pattern: [smart object type]/[smart object index]/[resource type] [23]. To set a smart object parameter, a PUT request has to be done instead. For instance, here is how you turn on a light having the index 1:

Req: PUT /lt/1/on (Accept: text/plain)
Res: 2.05 Content (text/plain)
    Body: 1
4.3 IoT-Framework

The IoT-Framework is a web system that works as a search engine which offers streams of data provided by multiple sensors of different types (temperature, humidity, pollution...). It was developed in the Project CS course given at Uppala University, in collaboration with Ericsson Research and SICS (Swedish Institute of Computer Science).

The purpose of the IoT-Framework is “to easily view, handle and interact with data streams. Within the system, users can register their sensors, create streams of data (e.g. the temperature in Uppsala), and view them on a graph. In addition, the system supports searching capabilities, helping the user with a full-text query language and phrase suggestions, allowing a user to find such streams using filters (based on the location of the devices or the meta-information provided by the user through tags), sort them using different criteria and rank the found results.

Moreover, users may be interested in the combination of several streams in order to know certain values such as the average, sum, minimum and maximum, instead of the measurements taken in a specific location. Thus, IoT-Framework also allows for this kind of aggregations, creating a virtual stream.

Finally, the IoT-Framework also supports the creation of triggers attached to streams and virtual streams. A trigger is a mechanism that notifies the user when a specific criteria is met such as reaching a certain temperature. Currently, the available functions are lesser than, span and greater than". [24]
Chapter 5

First Step: Rule Engines

5.1 Introduction

If we have a look at the IoT use cases listed in 3.2, we can notice that the word “smart” is used many times. However, truly smart systems are not quite there yet. Today, the Internet of Things is at a stage where we begin to be able to collect data of all kinds using sensors networks to help humans taking better decisions. This is a step in the right direction but it would be even better if this data could be directly used by other machines without humans having to act. In order to do so, machines need to be able to take decisions themselves given the appropriate input, hence the word “smart”. A possible way to build such smart machines is to use AI (Artificial Intelligence) planning techniques.

Moreover, we would like smart systems to be able to solve not only a specific problem, but a large class of problems effectively [25]. Indeed, it would be cumbersome to have to write a separate program for every single problem we might want to solve, and rather limited. It would be ideal if the core algorithm implemented in a smart street lighting system could be reused in a fall detection application for instance. This can be achieved by taking advantage of a subfield of AI planning, called domain-independent AI planning, which simply means that the AI planner does not rely on the problem characteristics to solve it.

The following sections (from 5.2 to 5.4) and well as the next two chapters investigate to what extent various AI planning techniques can be applied to some common IoT
use cases. The remainder of this chapter and chapter 6 explore domain-specific techniques while chapter 7 one focuses on domain-independent AI planning.

## 5.2 Motivating Examples

Needless to say, not every IoT application can benefit from AI planning techniques. For example, the chemical leakage detection system or the potable water monitoring system which we described previously is mainly about supervision. At most, those systems could push a notification to a website to warn citizens when the water quality gets worse. This is so simple that it would not leverage the power of AI planning. The same cannot be said for some other applications. Without being very complex, intrusion detection systems or fall detection systems require (a little) more intelligence. To take the example of the intrusion detection application, if a non-authorized person gets inside your home, you might want the system to send you a notification on your smartphone, alert the police and your neighbors, and automatically turn on the surveillance cameras. This kind of simple Machine to Machine (M2M) communication based on a trigger can be solved using IFTTT-like applications.

## 5.3 IFTTT-like Applications

IFTTT, which is the acronym for “If This Then That”, is an application that automatically performs an action when a particular event is triggered [26]. It is based on the idea of using “recipes” following the simple “if this then that” format to automate tasks. It can interact with web applications as well as with the physical world. Some example triggers (the “this” part of a recipe) are “it is going to rain tomorrow” or “I leave home” [27]. Some example actions (the “that” part of a recipe) are “Send me an email” or “Add a calendar event”. IFTTT is available as a web application as well as on mobile (it is limited to iOS and Android devices).

IFTTT supports a growing number of smart devices such as Philips hue connected bulbs [28], the Nest thermostat [29] or the Fitbit wireless activity tracker [30]. Recipes for the Internet of Things include: “Set your [Nest Thermostat] to Away and turn your Philips hue bulbs off”, “Add your Fitbit daily activity summaries
to a Google spreadsheet” and “Text a neighbor when your Nest Protect detects a smoke alarm emergency” [31].

5.4 Limitations

Even though this basic AI planner can already become handy in a lot of situations and solve some of the IoT use cases we encountered so far, it is obviously rather limited. Firstly, the task to achieve has to fit the restrictive pattern “if this then that”. Logical operators such as “or”, “and”, or “not”, which would let us describe more complex situations, are not supported. Secondly, this kind of AI planner does not allow more complex service composition, where a task that cannot be directly performed, could be decomposed into smaller (solvable) subtasks. The following chapter describes an AI planner which provides this modularity.
Chapter 6

Second Step: Domain Specific Planning

6.1 SHOP2

SHOP2, which stands for Simple Hierarchical Ordered Planner 2, belongs to the family of HTN (Hierarchical Task Network) planners [32]. In HTN planning, tasks to perform are decomposed into always smaller substasks, until it reaches a point where each subtask can be executed directly by the program. To know how to decompose a task into substasks, the planner needs to be provided with methods on how to do so, which have to be written beforehand for each type of task that you are interested in solving.

In the International Planning Competition (IPC) 2002, SHOP2 was distinguished by winning an award for its good performance. [33]

6.2 Implementation

6.2.1 Architecture Overview

The system is composed of two main elements (see the diagram below):
The IoT-Framework is the search engine for sensors that has been described above, whereas the AI planner is a SHOP2-like planner.

Let us go through an example to understand how it works. Suppose we want to optimize the ventilation usage in a meeting room. Ideally, we would want to turn off fans when the room is empty, and turn them on if there is a meeting, provided that it is too hot inside. In that scenario, we would provide the system with a keyword such as “optimize-ventilation”. The AI planner would then query the different sensors in the room (possibly temperature sensors and sensors measuring the amount of CO₂) thanks to the IoT-Framework. After fetching the latest values produced by those sensors, the IoT-Framework would send them back to the AI planner, which would then process them to produce a plan, to be executed right away, or later. In the latter case, it could be stored in the repository workflow.

More precisely, for each kind of task you are interested in executing, you have to define the methods and requirements you need in order the generate a plan, hence the two files you can see on the diagram above, called “Planning domain” and “Planning requirements”. In this scenario, the methods could be: “Turn on the
fans” and “Turn off the fans”, and the requirements: “The temperature is above 25 °C” and “The amount of CO₂ is higher than <define a threshold here>”. In the ventilation example, when using the keyword “optimize-ventilation”, the system will know that it will have to use the methods defined in the file related to ventilation optimization. Those methods are taking as a parameter the temperature in the room, and the amount of CO₂, so the AI planner will query the IoT-Framework, which will look for all the streams having either the tag temp or CO₂. Once the planner gets the latest values, it produces a plan.

The source code of this project is available on GitHub at the following address: https://github.com/qbahers/rbhop.

### 6.2.2 About the Planner

The good thing with HTN planners, is that they are quite easy to implement, and efficient in a lot of scenarios (the ventilation scenario we just described is a good example of scenario where HTN planners work well).

The AI planner implemented is inspired from Pyhop, a SHOP2-like AI planner [34]. There are three reasons for using this one instead of SHOP2. The first reason is that Pyhop is much easier to understand, since its implementation contains around 200 lines of codes only. The second reason is that it is easier to use (you don’t need to learn a specific planning language like PDDL). The third reason is that it was easier to integrate with the IoT-Framework. As for the performance difference between SHOP2 and the AI planner implemented, it will stay unnoticed for typical applications.

### 6.3 Limitations

Overall, for most of the IoT use cases, whether it is a smart home, a smart street lighting or a smart office application for instance, HTN planning will do just fine. The reason for this is that computing a plan basically comes down to finding a way through propositional formulae (using propositional variables as well as boolean operators) and “using” recipes to decompose tasks. Moreover, the number of states is generally quite small. In particular, it has been shown that relatively recent
algorithms were good enough to solve smart environment scenarios found in the literature [35]. However, we do not always have ideas for how to look for a solution to a particular problem. Therefore, for some applications, we cannot use “recipes” to guide the search. An example for this is the smart waste management scenario. That is when domain-independent planning techniques come into play.
Chapter 7

Third Step: Domain Independent Planning

7.1 Motivating Examples

7.1.1 Smart Waste Management

Today, garbage trucks pass by every household every week or so to empty bins. This way of collecting waste is inefficient because it does not take into account whether bins are actually full or not. To improve waste collection, we could put sensors inside bins to check their fill level. As soon as they would be nearly full, sensors would send a signal to a centralized system consisting of an AI planner, which would produce an optimized path for collecting full bins. This would allow to save fuel and pollute less. Moreover, by keeping track of how much waste each resident is generating, local authorities could reward those who are trying to produce less waste and recycle whenever it is possible, by slightly reducing their taxes for instance [7].

7.1.2 Agricultural Drones Route Optimization

Another scenario which can possibly leverage the power of domain-independent planners is the agricultural drones route optimization problem. At the moment, it is difficult for farmers to regularly check the growth of their crops, because farms are
Chapter 7. Third Step: Domain Independent Planning

becoming bigger and bigger. To assist them in monitoring their fields, one could imagine using agricultural drones with imaging capabilities [36] to take instant “pictures” of the health of their crops. Those drones would help farmers spot various diseases such as pest and fungal infestations that are difficult to notice at eye level. This would result in increased yields, as well as a decrease of pesticides use.

Regarding pesticides, farmers sometimes use helicopters to spray crops. We can imagine that in the future, helicopters will be replaced by drones. When this will become a reality, this will be yet another occasion to use AI planners to find an optimized path for the crop spraying drone and spare resources, whether it is fuel or electricity.

7.2 Heuristic Planning

7.2.1 International Planning Competition

The International Planning Competition, which is very often shorten to IPC, is an event held every two or three years during the International Conference on Automated Planning and Scheduling (ICAPS). Its main goal is to advance research in the automated planning area [37]. Since the first IPC in 1998, it has been the main incentive for pushing back the state of scientific research in the field [38]. Thus, it felt natural to look at the results of the last IPC competition to this day, held in 2011. In particular, the International Planning Competition is responsible of the advent of heuristic search in AI planning, which revolutionized the field, and has been dominating every competition for the last ten years or so, including the IPC 2011 [25]. The following sections focuses on the winner and runner-up of the sequential satisficing track (LAMA and Fast Downward Stone Soup respectively) and the winner of the sequential multi-core track (ArvandHerd) [39].

7.2.2 Fast Downward Stone Soup & LAMA

Fast Downward Stone Soup is a portfolio planner which uses different variants of the Fast Downward planner [40]. More specifically, the Fast Downward planning system
defines the basis upon which the different versions are built, variants differing from one another by using distinct search algorithms and heuristics. Fast Forward gained notoriety in 2004, when it won one of the tracks of the International Planning Competition [41]. Its success has led some researchers to try to make it even more efficient by tweaking it using new search algorithms or heuristics. By observing those experiments, three researchers, Malte Helmert, Gabriele Roger and Erez Karpas, realised two things:

1. “There is no single common search algorithm and heuristic that dominates all others for classical planning.”

2. “If a planner does not solve a planning task quickly, it is likely not to solve it at all.”

From there, they came up with the idea of using a portfolio of planning algorithms instead of a single one to build their planner (following the first assumption), and to run them sequentially, allocating to each of them a short time slot to try to find a solution (following the second assumption). Once one of the planners manages to find a solution, it is returned and the program terminates.

LAMA, like Fast Downward Stone Soup, is another planner based on the Fast Downward system, best known for finishing in the first place in the sequential track competition during the IPC 2011.

### 7.2.3 Arvand & ArvandHerd

Arvand is a planner which uses random walks in addition to heuristic search [42]. The reasoning for this is that heuristic search alone works well when it comes to quickly generate a solution, but is often unable to compute a nearly optimal plan, which is an issue if the problem to solve is resource-constrained. The smart waste management scenario is an example of such a problem. The reason behind heuristic search inefficiency is due to the fact that in regions of the search space called “plateaus”, where all the neighbouring states have more or less the same estimated cost, the search algorithm chooses the state with the lowest estimated cost, which is in reality no better than the others. This choice may eventually lead to a solution which is far from being close to optimal. To address this issue, Arvand combines
long jumps with random walks to quickly leave the plateau and explore further away states in the hope to find one with a low estimated cost.

ArvandHerd is simply the multi-core version of Arvand.

### 7.3 Experiments

#### 7.3.1 Problem Description

The problem considered for the experiment is the “Smart Waste Management” one discussed hereinabove: given information about which garbage trucks of a city are full (thanks to smart bins having sensory capabilities) as well as the distance between them, you want to generate a workflow using AI planning so that a garbage truck can come and empty them (the distance between two bins can be computed using the shortest path algorithm, provided that there is some metadata associated with the dustbin giving you its position).

The goal of the experiment is to determine if state-of-the-art planners are able to solve this problem for a number of bins which is large enough to be used in practice.

#### 7.3.2 Problem Modeling

The problem has been modelled in the PDDL language, which is the language used by a large majority of planners today. Two different files are necessary to characterize a problem in PDDL: the domain file and the problem file. The domain file describes the type of objects that we want to represent, which actions those objects can perform, under which conditions they are possible, and what is the effect of achieving an action. The problem file describes how many objects we have, set its characteristics and defines the initial state as well as the goal state.

In our case, the domain file states that the different objects that can be manipulated are one or more garbage trucks and one or more dustbins. Both garbage trucks and dustbins have a position and a capacity. Since the capacity of a garbage truck is not a limiting factor in practice, we consider in this experiment that the capacity of the
garbage truck is infinite. On top of that, the garbage truck can perform different actions, such as:

- Pick up a bin
- Drop a bin
- Move between different locations

The problem file specifies the position of the different bins (only full bins are considered), the distance between two bins if there is a path between them, and the position of the dustbin (which is empty) in the initial state. In the goal state, all the different bins must be dropped at the location where the dustbin initially started.

Performing an action, whether it is picking up a bin, dropping a bin, or moving between different locations, has a cost. The goal of the AI planner is to generate a plan which solves the problem while minimizing its cost.

The domain file used comes from a set of test problems provided for the International Planning Competition 2014 [43]. The corresponding problem file has been modified to fit the “Smart Waste Management” problem. The domain file as well as an example of the problem file can be found in the appendix at the end of this report.

### 7.3.3 Planner Used

The planner used in this experiment is the “Fast Downward” planner [44]. I decided to go with that one for two main reasons. The first reason is that it is one of the best performing planner to this day (as mentioned above, it was one of the best planner competing in the IPC 2011). The second reason is that its documentation is quite good. In particular, clear instructions are provided on how to run it, which is far from being always the case. The “Fast Downward” planner is now a merge of two projects, the initial Fast Downward project and the LAMA project (which won the sequential satisficing track in the IPC 2011).

The planner can be run using different configurations. For this experiment, I have chosen to use the “LAMA 2011” configuration of the planner, which is to some degree the same configuration as the one used by the LAMA 2011 planner during the IPC 2011 [45].
7.3.4 Results

The experiment has been conducted using a Mac Book Air 2013 edition, which has the following characteristics:

- Processor: 1.3 GHz, Intel Core i5
- Memory: 8GB 1600 MHz DDR3

The operating system used is Ubuntu 12.04 LTS, run under the VirtualBoxVM virtual machine.

The table below shows the results that have been obtained after running the Fast Downward planner for instances of the problem with an increasing number of bins.

<table>
<thead>
<tr>
<th>Number of Bins</th>
<th>Peak Memory</th>
<th>Search Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>7420 KB</td>
<td>1.5s</td>
</tr>
<tr>
<td>15</td>
<td>9132 KB</td>
<td>1.8s</td>
</tr>
<tr>
<td>20</td>
<td>17248 KB</td>
<td>4.8s</td>
</tr>
<tr>
<td>25</td>
<td>32436 KB</td>
<td>12.4s</td>
</tr>
<tr>
<td>30</td>
<td>49440 KB</td>
<td>22.2s</td>
</tr>
<tr>
<td>35</td>
<td>87628 KB</td>
<td>36.0s</td>
</tr>
<tr>
<td>40</td>
<td>110472 KB</td>
<td>65.1s</td>
</tr>
</tbody>
</table>

Table 7.1: Evolution of the peak memory and total time in relation to the number of bins

Before further analysis, it should be pointed out that the algorithm does not terminate (the timeout has been set to 10 minutes). After finding a possible plan, it resumes its search to try to find a better one. The search time in the table above corresponds to the time it took to find the first valid plan.

Below is the plot showing the impact of the number of bins on the search time:
7.3. Experiments

The following graphs illustrate the impact of the number of bins on the search time and peak memory:

**Figure 7.1: Impact of the number of bins on the search time**

**Figure 7.2: Impact of the number of bins on the peak memory**

Those two graphs seem to indicate that both the time complexity and memory complexity of this problem are exponential. This means that only instances of the problem for which the number of bins is small (say less than 60-80 bins) can be solved in a reasonable amount of time.
7.4 Limitations

The issue with the smart waste management scenario (or the one about optimizing routes for agricultural drones), is that they are similar to the traveling salesman problem, which belongs to the family of NP hard problems. Concretely, it means that state of the art AI planners, like the ones we just presented, are having trouble to find a short path if more than 60-80 cities are involved, or 60-80 bins, if we consider the bin collection problem instead. To put it differently, they do not scale very well. This leads to the need of finding optimizations, since you would want to collect every full bin in a city or district, say one thousand bins or so. The next chapter develops two possible ways to address this issue.
Chapter 8

Scalability

8.1 Scaling with Preprocessing

It goes without saying that preprocessing should be considered on a case-by-case basis. For the sake of coherence, let us continue with the smart waste management scenario, in the hope that the optimizations found for this use case may be applicable to other unrelated applications. As we have seen in the previous section, if we keep the problem description as it is, even the best AI planner out there won’t be able to find a solution to the dustbins collection problem, and that even if the number of full bins involved is relatively small. Thus, the problem description needs to be tweaked.

A simple idea we can have is the following: instead of considering full bins as checkpoints, we could only consider one full bin per street. The reason for this is simple: it basically does not matter if you have one bin or more located on the same street to collect, since you will have to travel across the whole road anyway. Alternatively, we could try to consider road intersections we have to pass by instead. Depending on the length of streets, using this technique could reduce the number of vertices by a factor of 5, just to make an educated guess. This means that now, state of the art AI planners would be able to find an optimized route for collecting bins if the number of bins involved is around not 30, but 150. Unfortunately, this number is still too small to consider using AI planning in the real world for this precise scenario.
Another idea we can then have is to try to separate the city concerned into different areas, solve the problem for every area using AI planning, and run the algorithm another time to figure out in which order you have to pass by every zone. This leads us to the next section, which explore different approaches to cluster bins in different groups.

### 8.2 Scaling with Clustering

#### 8.2.1 Kd Tree Space Partitioning Approach

The simplest approach is to put a grid on top of the city map, dividing it into squares, forming the different groups of bins. This approach is the first that is considered in the Google Maps API documentation, in the section dealing with restricting the number of markers on a map [46]. The issue there is that depending on the size of the squares, the number of bins per group might still be too high. Instead of using a finer grid to fix this problem, we can come up with a more clever idea borrowed from the nearest neighbor search in a 2d tree algorithm (see [47] for more information on the topic): you start by dividing the space into two groups containing the same amount of full bins (whether you divide it horizontally or vertically does not matter). Then you recursively do the same for the two sub-spaces, until you end up with groups of bins small enough for an AI planner to find a solution. It appears that this space partitioning technique is much better than the mere grid technique. The reason is that the grid technique works well is the bins are evenly distributed on the plan, which is not the case in practice: this leads groups of bins to be a lot smaller than they should be. The recursive space partitioning technique, on the other hand, is good at adapting itself to the typically irregular repartition of bins in a city.

#### 8.2.2 Minimum Spanning Tree Approach

Another approach is to use the single-link k-clustering algorithm. Here is how it works [48]:

1. Form V clusters of one object each.
2. Find the closest pair of objects such that each object is in a different cluster, and merge the two clusters.

3. Repeat until there are exactly k clusters.

One can note that is this exactly the Kruskal’s algorithm to compute the minimum spanning tree, except that we stop when we have k connected components.

The key there is to stop for the minimum number of clusters k for which the size of biggest cluster does not exceed the size after which the problem becomes unsolvable by the AI planner.

### 8.2.3 Computer Networks Analogy

If we think about it, connected bins are not that different from computer networks. Consequently, why not looking in that direction to see if they did not have to face the same scalability issues, and how they managed to solve them. Luckily, they did face those problems. The way they succeeded in solving routing scaling issues is by adjusting the size of IP prefixes at the router level, using techniques such as subnets (split one less specific prefix into multiple more specific prefixes) and aggregation (join multiple more specific prefixes into one large prefix) [49].

If we go back to our dustbin network, instead of considering real IP addresses, we can consider addresses following the same principle. Instead of being of the form: 192.168.47.42, those addresses could follow the pattern /[name of the city]/[name of the district]/[name of the street]/[street number]. This is possible to achieve if we use a system like the IoT-Framework (see Chapter 4 section 3), which can among other things store a lot of metadata related to a particular sensor (associated with a bin).

This approach seems to be best if enough metadata is provided because we will end up with cluster of bins that will make sense (composed of groups of streets or groups of districts), which won’t be necessarily the case if we follow the previous approaches described. This means that the path found to collect bins should be logical.
Chapter 9

Conclusion

9.1 Summary of Master’s Thesis Achievements

An overview of different scenarios where AI planning techniques can be successfully applied to the Internet of Things has been achieved. Those scenarios have been classified into three different categories: easy, medium and hard problems. For the first one, some solutions already exist. For the use cases which belong to the second category, a system based on an HTN planner has been developed, and connected to the IoT-Framework. Lastly, we have shown that state-of-the-art AI planners are able to solve more difficult problems. The only issue is that they can sometimes only solve small instances of the problem. Eventually, we have provided hints on how to circumvent this issue by doing some preprocessing as well as by using clustering techniques.

It should be pointed out that the best AI planner out there is not necessarily the best solution for every problem. Depending on the problem to solve, a simpler AI planner, like the one used in the second attempt, can be preferable, as it is a lot simpler.

9.2 Future Work

This report has shown that AI planning is an effective technique to add some automation and intelligence to typical IoT scenarios. Today, all the building blocks are
nearly available to build useful IoT applications taking advantage of AI planning.

The next step will be to make the IoT-Framework IPSO-compatible (the IPSO Alliance released its first draft on semantic Interoperability for the Internet of Things at the beginning of October 2014 [50]) and then conduct experiments with smart devices supporting this specification (or any other specification aiming at unifying their interfaces) when they will be available on the market.

Another possible improvement could be to connect a state-of-the-art planner to the IoT-Framework, like Fast Downward, in addition to the HTN based AI-planner that has been implemented, so that we can use the IoT-Framework for a wider range of use cases.

Lastly, in this work it is assumed that the environment is static during the computation (and the execution) of the plan. What this means is that once the planning process has started, we make the hypothesis that all of the incoming data used to generate a plan will stay accurate. However, this is not the case in practice, especially in an IoT context, where new information gathered by sensors can come at any time and all the time. Thus, it would be interesting to investigate how AI planners could deal with data volatility, so that it would be possible to re-plan while the plan is being generated or executed, without having to do the whole computation again.
Appendix A

Waste Management: Domain File

;; Transport sequential
;;

(define (domain transport)
  (:requirements :typing :action-costs)
  (:types
    location target locatable - object
    vehicle package - locatable
    capacity-number - object)
  (:predicates
    (road ?l1 ?l2 - location)
    (at ?x - locatable ?v - location)
    (in ?x - package ?v - vehicle)
    (capacity ?v - vehicle ?s1 - capacity-number)
    (capacity-predecessor ?s1 ?s2 - capacity-number)
  )
  (:functions
    (road-length ?l1 ?l2 - location) - number
    (total-cost) - number
  )
  (:action drive
    :parameters (?v - vehicle ?l1 ?l2 - location)
    :precondition (and
      (at ?v ?l1)
      (road ?l1 ?l2)
    )
    :effect (and
      (not (at ?v ?l1))
      (at ?v ?l2)
    )
  )
)
(increase (total-cost) (road-length ?l1 ?l2))

(:action pick-up
 :parameters (?v - vehicle ?l - location ?p - package ?s1 ?s2 - capacity-number)
 :precondition (and
 (at ?v ?l)
 (at ?p ?l)
 (capacity-predecessor ?s1 ?s2))
Appendix B

Waste Management: Problem File

; Transport city

(define (problem transport-city)
 (:domain transport)
 (:objects
  city-loc-1 - location
  city-loc-2 - location
  city-loc-3 - location
  city-loc-4 - location
  city-loc-5 - location
  city-loc-6 - location
  city-loc-7 - location
  city-loc-8 - location
  city-loc-9 - location
  city-loc-10 - location
  truck-1 - vehicle
  package-1 - package
  package-2 - package
  package-3 - package
  package-4 - package
  package-5 - package
  package-6 - package
  package-7 - package
  package-8 - package
  package-9 - package
  package-10 - package
  capacity-0 - capacity-number
  capacity-1 - capacity-number
  capacity-2 - capacity-number
  capacity-3 - capacity-number
  capacity-4 - capacity-number
  capacity-5 - capacity-number
  capacity-6 - capacity-number)
(:init
 (= (total-cost) 0)
(capacity-predecessor capacity-0 capacity-1)
(capacity-predecessor capacity-1 capacity-2)
(capacity-predecessor capacity-2 capacity-3)
(capacity-predecessor capacity-3 capacity-4)
(capacity-predecessor capacity-4 capacity-5)
(capacity-predecessor capacity-5 capacity-6)
(capacity-predecessor capacity-6 capacity-7)
(capacity-predecessor capacity-7 capacity-8)
(capacity-predecessor capacity-8 capacity-9)
(capacity-predecessor capacity-9 capacity-10)

; 480,435 -> 193,424
(road city-loc-4 city-loc-2)
 (= (road-length city-loc-4 city-loc-2) 29)
 ; 193,424 -> 480,435
(road city-loc-2 city-loc-4)
 (= (road-length city-loc-2 city-loc-4) 29)
 ; 918,341 -> 936,210
(road city-loc-5 city-loc-1)
 (= (road-length city-loc-5 city-loc-1) 14)
 ; 936,210 -> 918,341
(road city-loc-1 city-loc-5)
 (= (road-length city-loc-1 city-loc-5) 14)
 ; 651,235 -> 936,210
(road city-loc-6 city-loc-1)
 (= (road-length city-loc-6 city-loc-1) 29)
 ; 936,210 -> 651,235
(road city-loc-1 city-loc-6)
 (= (road-length city-loc-1 city-loc-6) 29)
 ; 651,235 -> 480,435
(road city-loc-6 city-loc-4)
 (= (road-length city-loc-6 city-loc-4) 27)
 ; 480,435 -> 651,235
(road city-loc-4 city-loc-6)
 (= (road-length city-loc-4 city-loc-6) 27)
 ; 651,235 -> 918,341
(road city-loc-6 city-loc-5)
 (= (road-length city-loc-6 city-loc-5) 29)
 ; 918,341 -> 651,235
(road city-loc-5 city-loc-6)
 (= (road-length city-loc-5 city-loc-6) 29)
 ; 560,901 -> 611,710
(road city-loc-7 city-loc-3)
 (= (road-length city-loc-7 city-loc-3) 20)
 ; 611,710 -> 560,901
(road city-loc-3 city-loc-7)
Appendix B. Waste Management: Problem File

(= (road-length city-loc-3 city-loc-7) 20) ; 447,732 -> 611,710
(road city-loc-8 city-loc-3)
(= (road-length city-loc-8 city-loc-3) 17) ; 611,710 -> 447,732
(road city-loc-3 city-loc-8)
(= (road-length city-loc-3 city-loc-8) 17) ; 447,732 -> 480,435
(road city-loc-8 city-loc-4)
(= (road-length city-loc-8 city-loc-4) 30) ; 480,435 -> 447,732
(road city-loc-4 city-loc-8)
(= (road-length city-loc-4 city-loc-8) 30) ; 447,732 -> 560,901
(road city-loc-8 city-loc-7)
(= (road-length city-loc-8 city-loc-7) 21) ; 560,901 -> 447,732
(road city-loc-7 city-loc-8)
(= (road-length city-loc-7 city-loc-8) 21) ; 663,402 -> 480,435
(road city-loc-9 city-loc-4)
(= (road-length city-loc-9 city-loc-4) 19) ; 480,435 -> 663,402
(road city-loc-4 city-loc-9)
(= (road-length city-loc-4 city-loc-9) 19) ; 663,402 -> 918,341
(road city-loc-9 city-loc-5)
(= (road-length city-loc-9 city-loc-5) 27) ; 918,341 -> 663,402
(road city-loc-5 city-loc-9)
(= (road-length city-loc-5 city-loc-9) 27) ; 663,402 -> 651,235
(road city-loc-9 city-loc-6)
(= (road-length city-loc-9 city-loc-6) 17) ; 651,235 -> 663,402
(road city-loc-6 city-loc-9)
(= (road-length city-loc-6 city-loc-9) 17) ; 362,940 -> 560,901
(road city-loc-10 city-loc-7)
(= (road-length city-loc-10 city-loc-7) 21) ; 560,901 -> 362,940
(road city-loc-7 city-loc-10)
(= (road-length city-loc-7 city-loc-10) 21) ; 362,940 -> 447,732
(road city-loc-10 city-loc-8)
(= (road-length city-loc-10 city-loc-8) 23) ; 447,732 -> 362,940
(road city-loc-8 city-loc-10)
(= (road-length city-loc-8 city-loc-10) 23)
(at package-1 city-loc-1)
(at package-2 city-loc-2)
(at package-3 city-loc-3)
(at package-4 city-loc-4)
(at package-5 city-loc-5)
(at package-6 city-loc-6)
(at package-7 city-loc-7)
(at package-8 city-loc-8)
(at package-9 city-loc-9)
(at package-10 city-loc-10)
(at truck-1 city-loc-10)
(capacity truck-1 capacity-10)
)
( :goal (and
(at package-1 city-loc-10)
(at package-2 city-loc-10)
(at package-3 city-loc-10)
(at package-4 city-loc-10)
(at package-5 city-loc-10)
(at package-6 city-loc-10)
(at package-7 city-loc-10)
(at package-8 city-loc-10)
(at package-9 city-loc-10)
(at package-10 city-loc-10)
))
( :metric minimize (total-cost))
)
Appendix C

Waste Management: Plan
Produced

(drive truck-1 city-loc-10 city-loc-7)
(pick-up truck-1 city-loc-7 package-7 capacity-9 capacity-10)
(drive truck-1 city-loc-7 city-loc-3)
(pick-up truck-1 city-loc-3 package-3 capacity-8 capacity-9)
(drive truck-1 city-loc-3 city-loc-8)
(pick-up truck-1 city-loc-8 package-8 capacity-7 capacity-8)
(drive truck-1 city-loc-8 city-loc-4)
(pick-up truck-1 city-loc-4 package-4 capacity-6 capacity-7)
(drive truck-1 city-loc-4 city-loc-2)
(pick-up truck-1 city-loc-2 package-2 capacity-5 capacity-6)
(drive truck-1 city-loc-2 city-loc-4)
(drive truck-1 city-loc-4 city-loc-9)
(pick-up truck-1 city-loc-9 package-9 capacity-4 capacity-5)
(drive truck-1 city-loc-9 city-loc-5)
(pick-up truck-1 city-loc-5 package-5 capacity-3 capacity-4)
(drive truck-1 city-loc-5 city-loc-1)
(pick-up truck-1 city-loc-1 package-1 capacity-2 capacity-3)
(drive truck-1 city-loc-1 city-loc-6)
(pick-up truck-1 city-loc-6 package-6 capacity-1 capacity-2)
(drive truck-1 city-loc-6 city-loc-4)
(drive truck-1 city-loc-4 city-loc-8)
(drive truck-1 city-loc-8 city-loc-10)
(drop truck-1 city-loc-10 package-1 capacity-1 capacity-2)
(drop truck-1 city-loc-10 package-2 capacity-2 capacity-3)
(drop truck-1 city-loc-10 package-3 capacity-3 capacity-4)
(drop truck-1 city-loc-10 package-4 capacity-4 capacity-5)
(drop truck-1 city-loc-10 package-5 capacity-5 capacity-6)
(drop truck-1 city-loc-10 package-6 capacity-6 capacity-7)
(drop truck-1 city-loc-10 package-7 capacity-7 capacity-8)
(drop truck-1 city-loc-10 package-8 capacity-8 capacity-9)
(drop truck-1 city-loc-10 package-9 capacity-9 capacity-10)
; cost = 333 (general cost)
Bibliography


