Supervision of Robot Tasks Planning Through Constraint Networks Acquisition

Mathias Paulin

Supervision of robot tasks planning through constraint networks acquisition.

Student: Mathias PAULIN
Supervisor: Jean SALLANTIN

LIRMM (CNRS/University of Montpellier), France,
{paulin, js}@lirmm.fr

Abstract. In recent years, robotic, and more precisely humanoïd robots, have made great improvements, especially in some close links with Artificial Intelligence. However, each time you have to design a new robot, it still exists a specific difficulty. You have, with some specific elementary actions, to combine them in a high-level vision to describe some complex tasks. In this article, we propose a theoretical framework using constraint networks to supervise planning of robot tasks. Each elementary action is modelled with a CSP automatically acquired by Machine Learning. To describe high-level task we only have to sequence some of the multiple acquired CSPs using planning tools. Some experimental results on a one-leg jumping robot have validated the automatically constraint networks modelling process.

1 Introduction

In recent years, robotic have made great improvements as the Sony's AIBO dog [1] or humanoïd robots such ASIMO [2] testify some. Current robots can perform a great number of elementary actions which are automatically executed. However, each time they have to design a new robot, the main problem that roboticians have, consists in linking and combining these elementary actions in order to perform a complex task. An elementary action is modelled by sets of mathematical equations which involve really complex physical laws, such as mechanical energy conservation law, kinetic momentum, partial derivative equations... Combine and link these elementary actions in order to perform a complex task is consequently difficult and requires a lot of computation hours, where the result is often an unique pre-computed sequence of elementary actions.

In this article, we propose to interact with roboticians in the planning process. Our approach is based on elementary actions automatically modelled with constraint networks by Machine Learning (using the CONACQ platform [4, 6]). The aim of the CSP acquisition is to **automatically** obtain a vocabulary $\mathcal{V}$ such that each word $w \in \mathcal{V}$ expresses an elementary action of the robot. The CSP modeling has the double advantage of abstracting a declaratory model from the elementary actions and to have good computational properties which is a capital aspect for planning. Thus, determining a valid sequence of elementary actions consists in building a valid sentence of words from $\mathcal{V}$. To perform high-level task, we sequence these multiple networks using planning tools. With this approach, we wish to relieve the modeling task of roboticians and contribute to the automatic constitution of laws of order for robots.

This paper is organized as follows. Section 2 presents some formal definitions of the basic concepts used in the article. In Section 3 we describe the theoretical framework we propose to tackle tasks planning of robots. Before concluding (Section 5), we present a first experiment which is a first empirical validation of our approach.

2 Preliminaries

2.1 Constraint Programming

A constraint network $\mathcal{P}$ is defined as a triplet $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a finite set of n variables, $\mathcal{D} = \{D(x_1), \ldots, D(x_n)\}$ is the set of their domains, and where $\mathcal{C} = \{c_1, \ldots, c_m\}$ is a sequence of constraints on $\mathcal{X}$. A constraint c_i is defined by the sequence $var(c_i)$ of variables it involves, and by the relation $rel(c_i)$ specifying the allowed tuples on $var(c_i)$.

The instanciation of values on the variables of $var(c_i)$ *satisfies* c_i if it belongs to $sol(c_i)$. An complete instance e on $\mathcal{X}$ is a tuple $(v_1, \ldots, v_n) \in D(x_1) \times \ldots D(x_n)$. An instance e is a solution of a constraint network if it satisfies all the constraints of the network. Otherwise, it is a non solution.

2.2 Constraint network acquisition

The goal of the constraint network acquisition consists in automatically acquiring a model from solutions and non solutions of the problem we want to model with a constraint network [4, 6, 7]. For the constraint acquisition process, the set $\mathcal{X}$ of the variables and their values domains $\mathcal{D}$ are known, and we dispose of E^+, a subset of solutions of the studied problem, and of E^- a set of non solutions. The goal of the acquisition process is to model the problem in a constraint solver. Our learning bias $\mathcal{B}$ is then a constraint library from this solver.

A constraints sequence belongs to the learning bias if and only if this sequence involves only constraints of the bias library. Given a set of variables $\mathcal{X}$, their domains $\mathcal{D}$, two sets E^+ and E^- of instances on $\mathcal{X}$, and a learning bias $\mathcal{B}$, the constraint acquisition problem consists in finding a set of constraints $\mathcal{C}$ such that $\mathcal{C} \in \mathcal{B}$, $\forall e^- \in E^-$, e^- is a non solution of $(\mathcal{X}, \mathcal{D}, \mathcal{C})$, and, $\forall e^+ \in E^+$, e^+ is a solution of $(\mathcal{X}, \mathcal{D}, \mathcal{C})$.

2.3 Planning problem

Planning[1] consists in finding, among a set of possible actions, which actions have to be applied, and how apply its, in order to attain a goal defined in a environment called World. The STRIPS formalism is commonly used to express planning problems. STRIPS uses the concepts of State, Goal and Action which are defined as follows. A ***State*** is a conjunction of positive literals describing the World at a specific time. A ***Goal*** is a particular state which describes partially the World at a final time. A state s ***satisfies*** a goal g if s contains all the literals in g. An ***Action*** A is defined by its name, its parameters, its *precondition* and its *effect*. The ***precondition*** of A is a conjunction of

[1] In this section, we present the planning problem in its ordinary definition [9].

literals which expresses the set of conditions which must be verified in order to execute A. The ***effect*** of A is a conjunction describing the state changes when A is executed.

Given an initial state S_i, a final state S_f corresponding to a goal, and $\mathcal{A}$ a set of actions, the planning problem consists in finding a sequence of actions from $\mathcal{A}$ such that if all these actions are performed from S_i, they allow to reach a state satisfying S_f.

3 Theoretical framework

The aim of approach we propose in this article consists in supervising quickly and easily the planning of tasks for robots. In this section, we describe the theoretical framework we propose and explain how and why using CSPs is a good alternative to the common roboticians approach.

3.1 General principle

Given a robot able to perform a set $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_m\}$ of elementary actions. An action $\mathcal{A}_i$ is considered as an ***elementary*** action when it is defined in a non decomposable way (in time, in moves, in robotic motor activation, etc.). Given a task which can't be performed with a single elementary action, the roboticians problem consists in combining different elementary actions in order to perform the given task.

For several years, Constraint Programming has been a success growing and is largely used in many industrial applications. The main reason of this success lies in its declaratory aspect and inherent dissociation between a model describing the problem ("What") and the techniques of resolution used to solve it ("How"). As presented previously, elementary actions are modelled by complex mathematical equations which are very difficult to solve. We propose consequently to model elementary actions by constraint networks using simple constraints which have good computational properties. To guarantee the requirements of speed and simplicity, we will use a set of very simplified constraints which will answer the planning problems and not complex mathematical equations. To determine a plan for performing a specific task, we will then use planning tools and constraints solver to determine a sequence on elementary actions which must be executed in order to perform the task.

3.2 Proposed approach

In the first part of this sub-section, we explain how we model elementary actions with CSPs. The second part presents how we use constraint acquisition techniques to automatically acquire CSPs. Finally, we describe how planning tools allow us to combine and sequence some of the multiple acquired CSPs in order to carry out a high level task.

Modeling elementary actions with CSPs

The approach we propose in this article is a generic one and could be deployed for any poly-articulated robot. In this section, we consider then a robot R which can perform

the set $\mathcal{A} = \{\mathcal{A}_1, ..., \mathcal{A}_m\}$ of elementary actions in a specific environment.

The robot R is consisted by a set $\delta = \{d_1, ..., d_n\}$ of descriptors which describe various dimensions of R and its environment, and by the set $\alpha = \{a_1, ..., a_k\}$ of its motors. Let be $\mathcal{A}_i \in \mathcal{A}$ an elementary action of R and $\mathcal{L}$ a constraints library. We propose to model $\mathcal{A}_i$ with CSPs in the STRIPS formalism by modelling three constraint networks $\mathcal{P}_p$, $\mathcal{P}_a$ and $\mathcal{P}_e$ defined as follows:

$$\begin{cases} \mathcal{P}_p = (X_p, \mathcal{D}_p, \mathcal{C}_p) \text{ models the conditions in which } \mathcal{A}_i \text{ can be performed,} \\ \mathcal{P}_a = (X_a, \mathcal{D}_a, \mathcal{C}_a) \text{ models how the motors must react to perform } \mathcal{A}_i, \\ \mathcal{P}_e = (X_e, \mathcal{D}_e, \mathcal{C}_e) \text{ models the state of } R \text{ after the execution of } \mathcal{A}_i. \end{cases}$$

In the previous definition, X_p, X_a and X_e are subsets of α, $\mathcal{C}_p$, $\mathcal{C}_a$ and $\mathcal{C}_e$ are subsets of $\mathcal{L}$, and $\mathcal{D}_p$ (*resp.* $\mathcal{D}_a$, $\mathcal{D}_e$) is the set of domains of the variables from X_p (*resp.* X_a, X_e). It should be noted that this CSP modeling allows to acquire, for each elementary action $\mathcal{A}_i$, a model which abstracts a set of possible instances of $\mathcal{A}_i$. Our approach models a *space* of possible solutions (and not a single one). Combine several elementary actions between them is consequently more easily.

Automatic CSPs building by constraint acquisition

In order to *automatically* model an elementary action $\mathcal{A}_i$, we have choose to use the constraint network acquisition platform CONACQ [4, 6].

For the acquisition process, we have to dispose of a constraints library $\mathcal{L}$, called learning bias, and of two disjoint sets E^+ and E^- which respectively contain positive instances of $\mathcal{A}_i$ and negative ones. This training data, simply given by roboticians, will be analysed by CONACQ which will automatically build (*c.f.* Section 4) a constraint networks $\mathcal{P}$ matching with the constraint acquisition problem. The constraint networks $\mathcal{P}_p$, $\mathcal{P}_a$ and $\mathcal{P}_e$ are then immediatly deduced from $\mathcal{P}$ as included sub-networks of $\mathcal{P}$.

The constraints library $\mathcal{L}$ must answer two partially opposite requirements: *simplicity*, in order to garantee good computationnal properties, and *modeling capability*, in order to efficiently describe each elementary action. The choice of $\mathcal{L}$ will be consequently a capital aspect of the modeling process and will be realized in interaction with roboticians in order to obtain the *best* ratio between simplicity and modeling capability.

Tasks planning

For each elementary action $\mathcal{A}_i \in \mathcal{A}$, we restrict then the acquired model to $(\mathcal{P}_p^i, \mathcal{P}_e^i)$, and express a task T to execute in the STRIPS formalism with the initial state E_i and a goal E_f. Finding a sequence of elementary actions allowing to perform T consists in these conditions in solving the planning problem $(E_i, E_f, \mathcal{A})$.

To solve this planning tasks, we use then planning tools used as black boxes. At the present time, we have not definitively choose the planning algorithm we will use.

According to criteria the roboticians will fix, we will prefer algorithms such as GRAPHPLAN [5], CPLAN [3] or other planning tools. However, we can note that the modelling of preconditions and effects by CSPs garantees strong propagation properties and allows to expect limited computation times for planning tools.

If there exists a plan for performing T, the planner will output a sequence $\mathcal{S} = |\mathcal{A}_i|\mathcal{A}_i \in \mathcal{A}|$ of the elementary actions the robot must apply to perform T.

4 TWIG: A preliminary empirical validation

In collaboration with the LIRMM robotic department, we experimented the automatic elementary actions modeling by CSP acquisition on a one-leg jumping robot named TWIG. Represented on Figure 1, TWIG has been designed by Sebastien Krut from [8]. Figure 1 is the result of the modeling of TWIG with the 3D Computer Aided Design software SOLIDWORKS [10]. We ran experiments on COSMOSMOTION, the physical simulator of SOLIDWORKS.

TWIG is constituted by a flywheel which is used to stay in vertical position, by a jack which TWIG uses to perform a longitudinal thrust, and by helicoid spring used to absorb a landing.
To describe TWIG, we have used the following set of descriptors: U_R and U_θ are the electrics powers respectively applied to the jack motor and the flywheel. R is the distance between the movable part of the jack and the fix one. $\ddot{x}_G$ and $\ddot{y}_G$ are the accelerations of the gravity point G, and x_G, y_G its coordinates. s is the foot/ground contact.

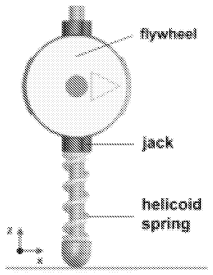

Fig.1 The TWIG robot.

For this preliminary experiment, we choose to model four actions of TWIG: Vertical jump, vertical landing, horizontal jump, and stable staying process. These actions are not elementary actions if we refer to definitions given above. However, this aspect does not have importance insofar as the purpose of this experiment were to validate that CONACQ was able to automatically model the actions of a robot with CSPs by constraint networks acquisition.

Each training instance given to CONACQ was a labelled (positive or negative) tuple $(t^i, s^i, x_G^i, z_G^i, U_R, U_\theta, t^f, s^f, x_G^f, z_G^f)$, where the t variable refers the time, and where the exponents i and f indicate respectively the variables state at the beginning and at the end of each action. CONACQ has automatically modelled the four actions by the following constraint networks:

Vertical Jump

$$\begin{cases} \mathcal{P}_p = \{(s^i = 1)\} \\ \mathcal{P}_a = \{(U_R = 500,\ U_\theta = 0)\} \\ \mathcal{P}_e = \{(s^f = 0,\ x_G^f = x_G^i,\ z_G^f = z_G^i + 3,\ t^f = t^i + 2)\} \end{cases}$$

Vertical landing

$$\begin{cases} \mathcal{P}_p = \{(s^i = 0)\} \\ \mathcal{P}_a = \{(U_R = 0,\ U_\theta = 0)\} \\ \mathcal{P}_e = \{(s^f = 1,\ x_G^f = x_G^i,\ z_G^f = z_G^i - 3,\ t^f = t^i + 20)\} \end{cases}$$

Horizontal Jump

$$\begin{cases} \mathcal{P}_p = \{(s^i = 1)\} \\ \mathcal{P}_a = \{(U_R = 300,\ U_\theta = 450)\} \\ \mathcal{P}_e = \{(s^f = 0,\ x^f_G = x^i_G + 3,\ z^f_G = z^i_G + 2,\ t^f = t^i + 6)\} \end{cases}$$

Stable staying process

$$\begin{cases} \mathcal{P}_p = \{(s^i = 1)\} \\ \mathcal{P}_a = \{(U_R = 0,\ U_\theta = 0)\} \\ \mathcal{P}_e = \{(s^f = 1,\ x^f_G = x^i_G,\ z^f_G = z^i_G,\ t^f = t^i + 1)\} \end{cases}$$

5 Conclusion

In this article, we have proposed a theoretical framework using constraint networks acquisition to acquire CSPs which model the elementary actions of a robot. We have shown in a second time how the acquired CSPs could be manipulated by planning tools in order to define quickly and automaticaly actions plans for poly-articulated robots.

Some aspects, such as the choice of the planning tools, are not yet defined totaly at present. However, the first experiment on the one-leg jumping robot TWIG has validated the automatic modelling process using CONACQ, which is the first step of our approach. So we are reasonnably optimistic for future works we have to carry out in order to make our theoretical framework as an effective platform for supervising tasks planning for robots.

Acknowledgments

We summarize in this paper the work we carried out in close co-operation with Jean Sallantin, Eric Bourreau and Rémi Coletta, for the theoretical and formal aspects, and Sébastien Krut and Christopher Dartnell for the TWIG experiments. That they all are here to thank for their contributions...

References

1. AIBO *http://www.eu.aibo.com/* Sony Aibo Europe - Official Website.
2. ASIMO *http://asimo.honda.com/* ASIMO Humanod Robot - Honda Robotic Technology.
3. P. van Beek, X. Cheng *CPlan: A constraint Programming Approach to Planning.* Proceedings of AAAI'99, 1999.
4. C. Bessiere, R. Coletta, F. Koriche, B. O'Sullivan *A SAT-Based Version Space Algorithm for Acquiring Constraint Satisfaction Problems.* Proceedings of ECML'05, Porto, Portugal, pages 747-751, October 2005.
5. A. L. Blum, M. L. Furst *Fast Planning Through Planning Graph Analysis.* Proceedings of Artificial Inteligence, pages 281-300, 1997.
6. R. Coletta, C. Bessire, B. O'Sullivan, E.C. Freuder, S. O'Connell, J. Quinqueton *Semi-automatic modeling by constraint acquisition.* Proceedings of CP-2003, Short paper, LNCS 2833, Springer Kinsale, Cork, Ireland., pages 812-816, September 2003.
7. A. Lallouet, A. Legtchenko *Two Contributions of Constraint Programming to Machine Learning.* Proceedings of ECML'05, Porto, Portugal, pages 617-624, October 2005.
8. Marc H. Raibert *Legged robots that balance.* MIT Press Series In Artificial Intelligence, Cambrige, Massachussetts, ISBN:0-262-18117-7, 1986.
9. S. Russell, P. Norvig *Artificial Intelligence - A Modern Approach.* Second Edition, Prentice Hall, ISBN:0-13-080302-2, 2003.
10. SolidWorks *http://www.solidworks.com* 3D CAD softwares.