Robotics and Autonomous Systems 62 (2014) 1047–1059

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Graph-based optimal reconfiguration planning for self-reconfigurable robots

Feili Hou*, Wei-Min Shen

Information Sciences Institute, University of Southern California, United States

HIGHLIGHTS

- On finding the least (dis)connect actions to reconfigure between arbitrary shapes.
- A proof that the optimal reconfiguration planning problem is NP-complete.
- An algorithm which can generate the optimal reconfiguration sequence.
- An algorithm that finds the near-optimal reconfiguration sequence in polynomial time.

ARTICLE INFO

Article history:
Available online 12 September 2013

Keywords:
Modular robots
Optimal reconfiguration planning
Computational complexity

ABSTRACT

The goal of optimal reconfiguration planning (ORP) is to find a shortest reconfiguration sequence to transform a modular and reconfigurable robot from an arbitrary configuration into another. This paper investigates this challenging problem for chain-type robots based on graph representations and presents a series of theoretical results: (1) a formal proof that this is an NP-complete problem, (2) a reconfiguration planning algorithm called MDCOP which generates the optimal graph-based reconfiguration plan, and (3) another algorithm called GreedyCM which can find a near-optimal solution in polynomial time. Experimental and statistical results demonstrate that the solutions found by GreedyCM are indeed near-optimal and the approach is computationally feasible for large-scale robots.

1. Introduction

Different from conventional robots, self-reconfigurable robots can adapt their own configurations and offer more versatile capabilities for challenging tasks in different environments. In applications such as reconnaissance, rescue missions, and space applications where the task and the environment are not always fully known in advance, the ability to adapt shape may be critical for a robot to accomplish its tasks, maximize its potential, and recover from unexpected damage. Self-reconfigurable robots with modular architecture would be ideal for such situations. Thus, realizing the ability of self-reconfiguration with a large number of independent modules has been one of the most important and challenging topics in the field of modular and reconfigurable robots.

Based on the design of modules, self-reconfigurable robots can be classified into two main categories: lattice-type and chain-type. In lattice-type robots, modules are arranged in a 2D or 3D lattice space of cells, and reconfiguration is achieved by a module to detach from its current lattice location, move along the surface of other modules, and then dock to a module at an adjacent lattice cell. Examples of such robots include 3D Fracta [1], Molecule [2], Telecube [3], Crystalline [4,5], ICubes [6], ATRON [7], Catom [8], the Programmable Parts [9], Stochastic-3D [10], Miche [11], Vacuubes [12] etc. In chain-type robots, modules can form movable chains and loops of any graphical topology, and the reconfiguration is achieved through "connect" and "disconnect" operations between modules along with the joint motion of chains. Hardware implementations of this class of robots include CONRO [13,14], PolyBot G3 [15], M-TRAN III [16], Molecubes [17], SuperBot [18–20], CKBot [21], Odin [22], YamoR [23], Roombot [24], Motein [25] etc. The different geometric arrangement of modules between lattice-type and chain-type modular robots makes their reconfiguration planning mechanisms fundamentally different.

This paper is mainly focused on the reconfiguration planning of chain-typed robots. The objective of self-reconfiguration planning is to figure out the reconfiguration steps for changing connectivity among the modules so as to transform the robot from the current configuration into a goal configuration. In the distributed fashion, it is the question of how the modules coordinate with others to figure out the necessary connections and disconnections given that they can only communicate and sense locally. Currently, only a few works were published on chain-type reconfiguration.

Casal [26] first tackled the problem of chain reconfiguration and presented a divide-and-conquer approach to the problem. This

* Corresponding author.
E-mail addresses: fhou@usc.edu (F. Hou), shen@isi.usc.edu (W.-M. Shen).

0921-8890/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.robot.2013.06.014

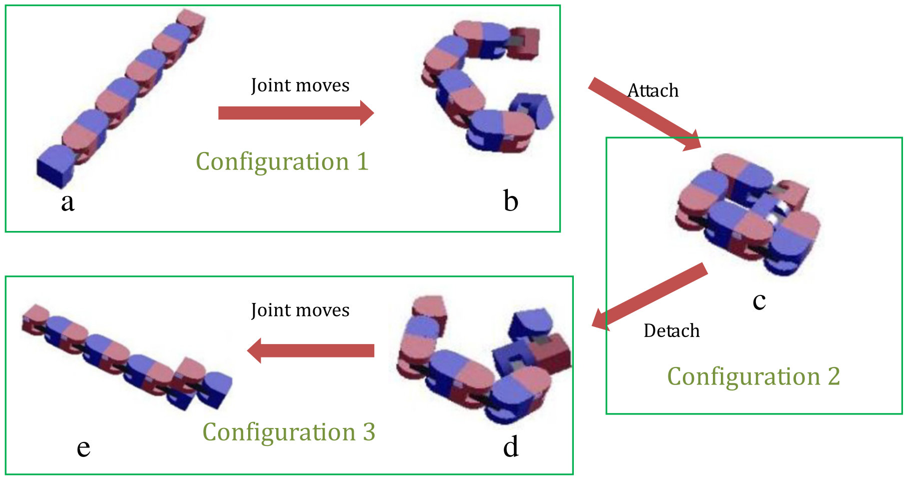

Fig. 1. An example of reconfiguration process.

algorithm is further developed by Yim et al. and theoretical results regarding its running time and the number of moves needed to reconfigure are given in [27]. Nelson [28] used the principal component analysis in conjunction with standard weighed bipartite graph matching theory to compare the initial and goal configurations and generate the reconfiguration steps. Payne [29] used a hormone inspired distributed controller and implemented the reconfiguration from "I" shape to "T" shape on CONRO. Gay [30] used the modules called DOF-Box II to units to build furniture that could change shape. Asadpour [31] has developed a self-reconfiguration planning method based on heuristic search, where the graph signatures method is used to test isomorphism between configurations, and the maximum common graph between configurations is used as a heuristic function to guide the search.

Aside from the research on reconfiguration planning, some other work on configurations is also indirectly related to our problem. I-Ming Chen [32] mentioned that the number of configurations is exponential, and even enumerating all the configurations is difficult. Castano [14] and Park [33] provide ways of finding the functionally identical configurations using graph isomorphism and configuration matching techniques. Chirikjian [34] established lower and upper bounds for the minimum number of moves needed for lattice-type reconfiguration, although it may not be applicable for chain-type reconfiguration.

The existing literature has used different techniques for finding an efficient reconfiguration sequence. However, the optimal solution with the least reconfiguration steps has never been reached. It is commonly believed that this problem is computationally intractable, but concrete evidence for this belief is still lacking. Some key research questions still remain open in this area. For example, how hard is it to find the least number of reconfiguration steps? Can an optimal solution be found efficiently? How to find the optimal solutions?

The goal of this paper is to investigate the optimal reconfiguration planning problem, i.e. finding the least number of reconfiguration steps to transform from one arbitrary configuration to another. During physical execution, some additional reconfiguration steps might be needed to compensate hardware limitation, which depend on many robot-dependent factors like the modules' internal degree of freedom, self-collision, gravity and others, and vary considerably from one module design to another. To gain the intrinsic theoretical insight of this problem, we exclude the hardware concerns for now, and work on the high-level connectivity planning in terms of graph representation in this paper.

We first analyze the complexity of the optimal reconfiguration planning problem by rephrasing the optimal reconfiguration problem into the configuration matching problem. Based on our previous work [35], we provide a theoretical proof that the optimal reconfiguration planning problem of finding the least number of reconfiguration steps to transform between two configurations is NP-complete, i.e. a polynomial algorithm is unlikely to exist. Next, two different reconfiguration planning methods are proposed for different needs. The first one is called MDCOP, which has a theoretical guarantee to find the optimal graph-based reconfiguration plan. MDCOP works by converting a reconfiguration problem into a distributed constraint optimization problem (DCOP), and then solve it through existing DCOP algorithms. The second algorithm is called GreedyCM and it can find extremely near-optimal solutions in time that is polynomial to the size of the robot. Since the control of modular robots is inherently distributed, both of our methods are developed in distributed fashion, where the robot can efficiently identify the reconfiguration steps in a multi-modular-coordination way.

In the rest of the paper, Section 2 defines the problem of self-reconfiguration planning and Section 3 maps the problem into a configuration-matching problem. Section 4 analyzes the computational complexity of the optimal reconfiguration planning. Section 5 presents the two algorithms for reconfiguration planning in the graph-based representations. Experimental results are demonstrated in Section 6, and conclusions and future work discussion are given in Section 7.

2. Problem statement

2.1. The optimal reconfiguration planning problem

A modular robot is composed of a set of modules, and its configuration is an arrangement of the connectivity of the modules. The two elementary reconfiguration actions for rearranging connectivity are: (1) making some new connections, also called *attach* or *connect* actions, which are usually executed along with the motion of chains, and (2) disconnecting some current connections, also called *detach* actions or *disconnect* actions. We say that two configurations are *adjacent* if one can be transformed into the other by one reconfiguration action. Fig. 1 shows an example of the reconfiguration process using SuperBot [18–20]. The robot transform from configuration 1 to configuration 2 by an attach action, and then to configuration 3 through a detach action. Please note that only attach or detach actions, rather than the joint movements, will change the configuration. So, in our example, (a) and (b) have the same configuration even though they look different. So do (d) and (e).