University of Zurich[UZH]

Dynamic Distributed Constraint Optimization in Signal/Collect

Thesis January 12, 2015

Daniel Hegglin
of Oerlikon ZH, Switzerland

Student-ID: 08-721-102
dani.hegglin@gmail.com

Advisor: **Mihaela Verman**

Prof. Abraham Bernstein, PhD
Institut für Informatik
Universität Zürich
http://www.ifi.uzh.ch/ddis

Acknowledgements

- Mihaela - Prof. Bernstein - Phillip Stutz - UZH - Proofreaders

Zusammenfassung

- Stand Constraint Optimization - Stand Dynamic Constraint Optimization - Implementation drei unterschiedlicher Approaches fÃ¼r Constraint Optimization fÃ¼r MeetingScheduling Problem - Benchmark in statischer und dynamischer Umgebung mit unterschiedlichen Konfiguration von Problemen, Agenten, Meetings, Dynamics - Ev. Conclusion Vorgeschmack

Abstract

This thesis is about understanding the performance and behaviour of various existing distributed constraint optimization algorithm approaches (complete, local-iterative message-passing, local-iterative best-response) in context of a dynamic environment, e.g changing constraints or other parameters. The use case for this analysis will be the scheduling problem, which needs to be mapped to the algorithms accordingly. The goals are adding a general module to the existing framework for dcops, which can simulate dynamic environments / parameter changes in various ways, mapping existing dcop algorithms to signal collect and evaluate their performance. An additional goal would be to suggest or test a blended algorithm with local-iterative characteristics that handles change better by applying techniques from dynamic approaches which are not local-iterative. The benchmarking is done with respect to change (resilience to change / stability, amount of variable value changes necessary to bounce back), solution quality (how fast can the algorithms reach a defined quality), Time-to-Convergence (how long does it take to converge).

Table of Contents

1

Introduction

1.1 Motivation & Goal

Constraint optimization allows to solve problems in various areas. The distributed nature of many of those problems has been extensively adressed by research in distributed constraint optimization and the formulation of numerous algorithms with diverse design approaches. However, most of those algorithms were designed and most studies are conducted on the premise that problems are static in their predefined state and do not change over the course of the problem solving process. This might work as as a step by step procedure, but it does not work in a distributed manner with multiple agents [**?**]. But as a matter of fact, many problems have dynamic properties and in a world with ever increasing complexity and speed, those become ever more relevant. Constraints can change, but also the involved variables as well as the problem domain itself. One could for instance imagine a real-time business analytics software continously calculating the most optimal solution to a problem or drones exploring an area to find survivors of a earthquake. Research in dynamic distributed constraint optimization is sparse. Mailler et al. attribute this to a lack of standardized benchmarks [Mailler and Zheng, 2014]. There has also been some research done at DDIS on constraint optimization problems with a special focus on max sum.

This thesis tries to explore dynamic distributed constraint optimization by implementing three different algorithm approaches and compare their performance in a dynamic environment. The main goal here is to understand the behaviour of the max-sum algorithm that should be able to handle changing properties. It is further a goal to show ways of benchmarking these type of problems from various aspects. The example problem for the thesis will be meeting scheduling and the implementation will be carried out on signal/collect, the graph processing engine developed at DDIS at ifi UZH. The goal here would also be to show the capabilities of the engine to handle these kind of problems.

1.2 Structure

First, an overview will be given about various definitions and aspects of constraint optimization in general, as well as the aspects of distributed and dynamic environments.

Further, an overview will be provided about different approaches of algorithms to solve constraint optimization problems and their advantages and disadvantages in various contexts as well as the family they are coming from.

In the design part, the problem definition and the mapping of the problem on to the three algorithms will be explained. Also, the common denominator parts of the algorithms and an interface for dynamic changes will be explained. The design of the data collection will also be briefly introduced.

In the implementation part, details of the mapping to signal/collect will be explained and a few words on the testbed solution will be added.

After that I will conduct benchmarks to measure first the properties of the algorithms related to the problem mapping. Second, I will run various tests on changing constraints, variables and the domain with different rates and different problem densities to determine the dynamic performance of the algorithms on the particular problem of meeting scheduling. Both benchmarks will be discussed, further work and limitations will be pointed out and a conclusion will be given.

2

Background & Related Work

2.1 Constraint Optimization

- related work - formal definition of a constraint optimization problem

2.2 Distributed Constraint Optimization

- related work: soDistributed Constraint Satisfaction (DisCSP) was formalized (Yokoo et al. 1998). Here, - why distributed how

2.3 Dynamic Distributed Constraint Optimization

- related work: Petcu, mailler, find more - aspects of dynamic environments - what can change in a problem

2.4 Problems

2.4.1 Meeting Scheduling Problem

- related work: find definition again that is citable -¿ book chapter 12 - explanation with formal definition - example

2.4.2 Other Problems

- brief related work - brief explanation with formal definition - brief example
  - graph coloring, ...

2.5 Algorithm Design Approaches

2.5.1 Complete

- basic idea - advantages - disadvantages

2.5.2 Local-Iterative - Best Response

- basic idea - advantages - disadvantages

2.5.3 Local-Iterative - Message Passing

- basic idea - advantages -disadvantages

2.5.4 Other Approaches

- brief related work - brief explanation of approaches and why they don't fit: too much information sent, centralized, too complicated, not localized

3

Design

3.1 Meeting Scheduling Problem

3.1.1 Formal Definition

- General problem and definitions - hard constraints Different, Same (sources for that) - timeslot utility design: first slot preference, preferences, soft constraints (free slots, 'blocked' slots)

3.1.2 Problem Generation

3.2 Framework

3.2.1 Basic Functionality

3.2.2 Dynamics Controller

3.3 Mapping of DPOP

3.3.1 Analysis

- Why DPOP - Pseudocode

3.3.2 Graph Structure

3.3.3 Vertices

3.4 Mapping of MGM

3.4.1 Analysis

- Why MGM - Pseudocode

3.4.2 Graph Structure

3.4.3 Vertices

3.5 Mapping of MaxSum

3.5.1 Analysis

- Why MaxSum - Pseudocode

3.5.2 Graph Structure

3.5.3 Vertices

3.6 Monitoring Platform

- Why is it needed: lightweight, non-blocking, analysis on other spot, direct transmit to analyzing platform, live monitoring - How can http requests be better than filewrites

4

Implementation

4.1 Signal Collect

- Explanation of the system - Explanation of Run Modes - Explanation of the basics behind it (Akka -¿ relation to play framework) - !!!! Stretch goal: Implemenation for Slurm done, multi machine is not done yet

4.2 Modes & Parameters

"density" "algorithm" "execution" "mode" "param" "timeslots" "meetings" "agents" "runs" "factoragents" "factormeetings "maxagents" "maxmeetings"

4.3 Monitoring Platform

- Play framework explanation - Akka actors as pools for utilities, conflicts, mean, etc. - Live monitoring with Websockets based on simple example - supports multiple simultanous test runs

5

Benchmark

5.1 Testing Environment

- few words on minions cluster - cpu, ram, partition speed, etc

5.2 Results I: Algorithms Performance in Static Environments

- Number of Agents - Number of Meetings - Number of Timeslots - Problem Density - Run Mode on Signal Collect

5.2.1 Time to Convergence

- Heatmap fÃ¼r alle drei - Verlauf alle drei - Scalability

5.2.2 Solution Quality over Time

- Heatmap fÃ¼r alle drei - Verlauf alle drei - Scalability

5.2.3 Conflicts over Time

- Verlauf alle drei - Scalability

5.2.4 Messages

- Verlauf alle drei - Scalability

5.3 Results II: Algorithms Performance in Dynamic Environments

- Additional parameter - Different Scenarios: Constraints, Variables, Domain - Change one, multiple - Amount of Change: Percentage, Number

constraints: mode, interval, percentage variables: mode, interval, new Neighbourhood, new Agent, add / remove, how many domain: mode, interval, percentage, increase / decrease

5.3.1 Time to Convergence

- Single Change

5.3.2 Solution Quality over Time

- Single change

5.3.3 Conflicts over time

- Rate / Avg. Conflicts - Density!!

5.3.4 Messages

- Number of exchanged messages -¿ maybe not

5.3.5 Stability: Utility

- Multiple change: Peak Average

5.3.6 Stability: Quality

- Multiple change: Peak Average

6

Limitations & Future Work

- mehr algorithmen testen - mehr probleme testen - generalisieren der aussagen - benchmarks ausbauen - distribute to multiple machines was not possible with signal collect at the time

7

Conclusions

- Ausgangslage - Was wurde gebaut: implementationen, was gut was schlecht - Was wurde getestet: welche tests waren sinnvoll, welche weniger - Was wurde herausgefunden: wo performen die algorithem schlecht, gut - Was kÃP$_n$nte nÃ$\frac{1}{4}$tzlich sein (testbed)

References

[Mailler and Zheng, 2014] Mailler, R. and Zheng, H. (2014). A New Analysis Method for Dynamic , Distributed Constraint Satisfaction. In Lomuscio, A., Scerri, P., Bazzan, A., and Huhns, M., editors, *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*, pages 901–908.

A

Appendix 15

A.1 Results I: Selected Data

A.2 Results II: Selected Data

List of Figures

List of Tables