University of Zurich[UZH]

Dynamic Distributed Constraint Optimization in Signal/Collect

Thesis January 13, 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

First and foremost, I'd like to thank my advisor Mihaela Verman, Research Assistant and PhD Candidate at the DDIS group. She was always supporting me and helped me with technical and formal questions. She provided me with helpful tipps concerning code quality and continous integration, she always had helpful comments on algorithm performance and papers. She never hesitated to give a comment and help me. She always had time. Without her continous efforts and inputs, this work would not have been possible.
Special thanks go to Dr. Thomas Scharrenbach for arranging the thesis proposal and his guidance at the beginning of this work. I'd also like to thank Philip Stutz for his work on the Signal/Collect framework that made my research possible and his support in technical questions regarding the usage of the framework. I hope his work will be useful to many other people and solve many other problems.
Finally, I'd like to thank the Dynamic and Distributed Systems Group (DDIS) at the University of Zurich and Prof. Abraham Bernstein for the opportunity to write my master thesis in their department. Without it's excellent infrastructure it would not have been possible to run the benchmarks in the given time.

Zusammenfassung

Abstract

Distributed Constraint Optimization is a well-researched subject and the various algorithm allow to solve many problems in various fields like for example graph coloring, traffic flow management or power grid management. However, the often dynamic nature of said problems is often not taken into consideration in the design of those protocols and benchmarking this kind of problems is not well-developed.

This thesis aimed to explore the dynamic properties and performance of different types of DCOP algorithms with a focus on local-iterative algorithms and especially on the MaxSum algorithm. A complete, a local-iterative message-passing and a local-iterative best response algorithm for distributed constraint optimization were implemented. The real-world use case of the meeting scheduling problem was mapped as a Distributed Constraint Optimization Problem for benchmarking. A framework has been designed that allows benchmarking of dynamic changes to constraints, variables and the problem domain. The algorithms have been benchmarked in a static as well as in a dynamic environment with various parameters. The main goal was to evaluate the time vs. solution quality and to try out proposed benchmarking techniques.

RESULTS

The thesis further proposes a solution to store the results of DCOP problem solving and monitoring the live performance of the algorithms.

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 [Petcu and Faltings, 2007]. 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 [**?**]. There has also been some research done at DDIS on constraint optimization problems with a special focus on max sum.

Describe problems here - brief related work - brief explanation with formal definition - brief example - graph coloring

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.

The implemented software is made available online under the Apache license version 2.0.[1][2]

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.

[1] https://github.com/danihegglin/DynDCO
[2] http://www.apache.org/licenses/LICENSE-2.0

2

Background & Related Work

In this section, constraint optimization and the distributed, as well as dynamic variants are briefly explained and brought into context of the related work. Also, the meeting scheduling problem are described and different algorithm design and their advantages and disadvantages are described and related work is mentioned.

2.1 Dynamic Distributed Constraint Optimization

[Chapman et al., 2011] nguyen trung 2012 Chapter 12 Distributed Constraint Handling and Optimization Farinelli

- related work - formal definition of a constraint optimization problem

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

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

2.2 Meeting Scheduling Problem

[Angulo and Godo, 2007] [Maheswaran et al.,] berger2008 Hassine 2007

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

2.3 Algorithm Design Approaches

Distributed Constraint Optimization can be done with numerous approaches.

Other Approaches: Bee Hive optimization, Genetic Algorithms, .. DynDCOAA, SBDO, Bee Colony algorithm, Ant colony algorithm, adopt, dsa-a, dsa-b stochastic ... - brief related work - brief explanation of approaches and why they don't fit: too much information sent, centralized, too complicated, not localized

The following subsections are going to explain the three chosen approaches for this thesis and which advantages, as well as disadvantages this approaches hold.

[**?**]

2.3.1 Complete

- basic idea - advantages - disadvantages

2.3.2 Local-Iterative - Best Response

[Chapman et al., 2011] [Maheswaran et al.,]

- Local iterative approximate best-response algorithms, such as the distributed stochastic algorithm (Tel, 2000; Fitzpatrick Meertens, 2003), the maximum-gain messaging algorithm (Yokoo Hirayama, 1996; Maheswaran et al., 2005), fictitious play (Brown, 1951; Robinson, 1951), adaptive play (Young, 1993, 1998), and regret matching (Hart Mas-Colell, 2000). In this class, agents exchange messages containing only their state, or can observe the strategies of their neighbours. In game-theoretic parlance, this is known as standard monitoring2 , and, as the name suggests, is a typical informational assumption implicit in the literature on learning in games.

- advantages - disadvantages

2.3.3 Local-Iterative - Message Passing

[Chapman et al., 2011] -

- advantages -disadvantages

3

Design

In this section, the benchmark problem will be defined and mapped as DCOP, the framework design will be explained and the mapping of the algorithms on to the Signal/Collect framework will be described. Additionally, the design and considerations regarding the implemented monitoring platform are detailled.

3.1 Meeting Scheduling Problem

3.1.1 Formal Definition as DCOP

- General problem and definitions - hard constraints Different, Same (sources for that) [Chapman et al., 2011] chun, andy 2003 grubshtein mes, martijn 2007

- timeslot utility design: first slot preference, preferences, soft constraints (free slots, 'blocked' slots)

Rajiv T. Maheswaran, Milind Tambe, Emma Bowring, Jonathan P. Pearce, and Pradeep Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed multi-event scheduling.

DCOP Formulations for DiMES Now that we have captured our problem in the DiMES framework, we need an approach to find an (optimal) solution. A DCOP is defined as a tuple ¡ A, X, D, R ¿ [2] where A is a finite set of agents A = A1, ..., Am, X is a set of variables X = x1, ..., xN distributed among the agents in A (each variable belongs to a unique agent), D is a set of domains D = D1, ..., DN where a domain Di is the set of values that variable Xi can take. Finally R is a set of relations R = r1...rp such that every relation defines a cost for the combinations of values from D, ri : Di1x..xDik R The goal is to choose values for variables to optimize an objective function, the utility. The convertion from a DiMES to a DCOP can be made in 3 different ways [6]: TSAV (Time Slots As Variables): a variable Xn(t) is the n-th resources t-th time slot. The complexity of TSAV grows if the time range increases or the time quantization interval decreases. EAV (Event As Variables): X K represents the starting time of E K, with domain t, ..., t PEAV (Private Events As Variables): A modification of EAV to ensure privacy protection for agents information. As in EAV each participant controls a private variable corresponding to the starting time of a meeting. But the difference is that in this formulation agents dont share their internal valuation for the resources, thus maintaining

privacy. The experimental results in shows that EAV outperforms PEAV by one order of magnitude when using passup in meeting scheduling. As in this project it is fundamental to preserve agents preferences as well as to have the best variable control scheme we will consider only PEAV formulation.

Multi-agent meeting scheduling with preferences: efficiency, privacy loss, and solution quality Franzin

problem density, etc

3.1.2 Problem Generation

- Decision: Available datasets are not usable because it was not possible to scale to high numbers. It was chosen to create problems randomly. Frodo2

- brauche ein paar papers die das untermauern

- participations fixed -

Distributed Meeting Scheduling: CHAPTER 10

3.2 Framework

3.2.1 Foundation

It was decided to build a specific framework for benchmarking dynamic problems. The foundation of the framework is the signal/collect framework [Stutz et al., 2010][1], which is built on top of Akka[2] and written in Scala[3]. It is a graph processing engine with a programming model that features vertices and edges. The vertices have a state and send signals along the edges to other vertices, which can contain any datatype. The signal usually is their state or in context of their state. The vertices collect the signals and perform a compution on the collected data, before adjusting their state according to the calculations and sending out the newly generated signal. This model allows to reduce complex algorithms to a few lines of code and is applicable for many problems.

The reason for choosing this framework in this thesis is the structural fit to distributed constraint optimization problems and the possiblity to add and remove vertices during runtime. This allows for dynamically changing problem computations. Further, the capability of running problems asynchronously or with synchronous signal steps and the possibility to distribute the system on multiple machines add to the advantages.

3.2.2 Parameters & Modes

This framework was designed with the benchmarks in mind. It should be possible to pass the general run parameters like which algorithm to use, which Signal/Collect run mode

[1] http://uzh.github.io/signal-collect
[2] http://akka.io
[3] http://www.scala-lang.org

(synchronous/asynchronous) to use, normal run mode or dynamic variations (changeConstraints, changeVariable, changeDomain) with specific parameters. For testing, two test modes were implemented: SingleTest and MultiTest. Single Test is for single runs obviously and MultiTest allows to specifiy how many runs each specification should be doing to create a median of those runs, how much the agents respectively the meetings should increase after a specification has been processed and where to stop. By this, it is possible to explore an area of specifications. Finally, it should be possible to define different problems. For this thesis, the parameters for problem density (blocked timeslots percentage), timeslots, meetings and agents were defined.

3.2.3 Functionality & Structure

The considerations for the structure were that I needs to be hierarchical to allow different constraint optimization problems to be run on the framework. So, the hierachy goes from a Base to a Dynamic to a Problem level. It should include ProblemFactories and Problems, GraphFactories and Graphs, as well as Vertices.

- Problem Factory: Problem -¿ MeetingSchedulingProblem class diagram

- Graph Factory: Graph -¿ DPOP Graph, MGM Graph, MaxSum Graph explanation - Graph

- BasicVertex -¿ DynamicVertex -¿ MeetingSchedulingVertex -¿ DPOP, MGM, MaxSum Vertex class diagram!!!

3.2.4 Dynamics Controller

how does it work, what can it do

- Registration of vertices - change constraints - change variables - change domain

- Initial design consideration was to make it part of the graph as signal collect allows mixup of different types. Created weird problems because I need to pause the vertex for interval changes. not part of the graph but connected to vertices -¿ because it was a problem with convergence and there was a problem with thread sleep in akka

3.3 Mapping of DPOP

based on: A Scalable Method for Multiagent Constraint Optimization

- Why DPOP DPOP is a primary example for a complete algorithm - Pseudocode

3.3.1 Graph Structure

3.3.2 Vertices

3.4 Mapping of MGM

- Why MGM Good example for Best-Response algorithm and DSA-B has already been implemented - Pseudocode

3.4.1 Graph Structure

3.4.2 Vertices

3.5 Mapping of MaxSum

[Yedidsion and Zivan,] Farinelli2008 zivan2012
- Why MaxSum MaxSum is the new kid on the block. - Pseudocode

3.5.1 Graph Structure

3.5.2 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 - 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

4

Benchmark

In this chapter the three algorithms are benchmarked with different parameters and scenarios. The first series of tests is conducted in a static environment for a basic understanding of the algorithm performance and the second series is held in various dynamic settings to explore the dynamic capabilities of the algorithms. The testing environment was the minion cluster of the departement of informatics at the University of Zurich. The cluster consists of 16 machines and each machine has 128 GB RAM and two E5-2680 v2 at 2.80GHz processors. Every processor has 10 cores and the interlink between the machines is a 40Gbps Infiniband setup. The cluster has different partition speeds (slow, fast, superfast). All tests were conducted on superfast partitions.

4.1 Results I: Algorithms Performance in Static Environments

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

theoretical foundation: chapman benchmarking hybrid algorithms, Mailler2014,

4.1.1 Time to Convergence

- Heatmap fÃ¼r alle drei - Verlauf alle drei - Scalability: one, multiple machines - Convergence Distribution

4.1.2 Solution Quality over Time

- Heatmap fÃ¼r alle drei - Verlauf alle drei - Scalability: one, multiple machines - Quality Distribution

4.1.3 Conflicts over Time

- Verlauf alle drei - Scalability: one, multiple machines

4.1.4 Messages

- Verlauf alle drei - Scalability: one, multiple machines - Message Distribution

4.2 Results II: Algorithms Performance in Dynamic Environments

petcu 2007

- 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

4.2.1 Time to Convergence

- Single Change - Distribution - Verlauf alle drei

4.2.2 Solution Quality over Time

- Single change - Distribution - Verlauf alle drei

4.2.3 Conflicts over time

- Rate / Avg. Conflicts - Density!!

4.2.4 Messages

- Number of exchanged messages -¿ maybe not

4.2.5 Stability: Utility

- Multiple change: Peak Average - Median Utility?

4.2.6 Stability: Quality

- Multiple change: Peak Average - Median Utility

5

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 - not fixed participations -¿ test effect of participations on performance

6

Conclusions

- Ausgangslage - Was wurde gebaut: implementationen, was gut was schlecht - Was wurde getestet: welche tests waren sinnvoll, welche weniger

- Welche Benchmarks sind sinnvoll - Was wurde herausgefunden: wo performen die algorithem schlecht, gut - Was kÃ$\mathbf{P}_n$nte nÃ$\frac{1}{4}$tzlich sein (testbed)

- FOCUS on contributions!

References

[Angulo and Godo, 2007] Angulo, C. and Godo, L. (2007). Distributed meeting scheduling. *Artificial Intelligence Research and ...*, pages 1–8.

[Chapman et al., 2011] Chapman, A. C., Rogers, A., Jennings, N. R., and Leslie, D. S. (2011). *A unifying framework for iterative approximate best-response algorithms for distributed constraint optimization problems*, volume 26.

[Maheswaran et al.,] Maheswaran, R. T., Pearce, J. P., and Tambe, M. A Family of Graphical-Game-Based Algorithms for Distributed Constraint Optimization Problems. pages 1–19.

[Petcu and Faltings, 2007] Petcu, A. and Faltings, B. (2007). Optimal Solution Stability in Dynamic, Distributed Constraint Optimization. *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'07)*, pages 321–327.

[Stutz et al., 2010] Stutz, P., Bernstein, A., and Cohen, W. (2010). Signal / Collect : Graph Algorithms for the (Semantic) Web.

[Yedidsion and Zivan,] Yedidsion, H. and Zivan, R. Applying MaxSum to DCOP MST.

A

Appendix 15

A.1 Results I: Selected Data

A.2 Results II: Selected Data

List of Figures

List of Tables