Karlsruher Institut für Technologie

Bachelor thesis

Title of Thesis

Name of author

Date: 19. September 2017

Supervisors: Prof. Dr. Peter Sanders
Dipl. Inform. Zweiter Betreuer

Institute of Theoretical Informatics, Algorithmics
Department of Informatics
Karlsruhe Institute of Technology

Abstract

In this thesis we augment the existing Hypergraph Partitioner KaHyPar with an evolutionary framework with the goal to improve the solution quality.

Acknowledgments

I’d like to thank Timo for the supply of Club-Mate

Hiermit versichere ich, dass ich diese Arbeit selbständig verfasst und keine anderen, als die angegebenen Quellen und Hilfsmittel benutzt, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Satzung des Karlsruher Instituts für Technologie zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe.

Ort, den Datum

Contents

1 Introduction

1.1 Motivation

Hypergraph Partitioning is a highly complex field of study. The Motivation behind this work is to add an evolutionary framework to the existing Hypergraph partitioner KaHyPar in order to improve the cuts of the Hypergraph.

1.2 Contribution

1.3 Structure of Thesis

2 Fundamentals

2.1 General Definitions

Hypergraph := (H;V;E;C) Node contraction := Node uncontraction := Net := cut := connectivity := quality := population := individual := solution := imbalance:=

3 Related Work

The Hypergraph Partitioner KaHyPar uses a multilevel approach for partition. The original Hypergraph H is coarsened by contracting two nodes u, v using a selection strategy until a certain limit is passed. On the coarsened Hypergraph a simpler partitioning algorithm is chosen to generate an initial partitioning. Afterwards to contraction operations will be reverted and during each step of the uncoarsening phase local search algorithms are used to improve the current connectivity of H. The local search is based on the principle of Fiduccia-Mattheysis algorithm where operations of decreasing quality are considered, but only executed if the sum of operations results in a net gain.
The memetic Graph Partitioner KaHiP uses evolutionary actions to increase solution quality and is the main inspiration for the operations implemented in KaHyParE.

4 KaHyParE

4.1 Overview

The main objective of the algorithm is to optimize solution quality within bounded time. The algorithm will produce multiple solutions

4.2 Population

We manage a Population P which is the resource for the partitions. An individual I is a member of the population, if it is a valid solution for the given Hypergraph H.

4.3 Diversity

We define diversity as

4.4 Selection Strategies

4.5 Combine operations

4.5.1 Basic Combine

The basic combine uses two parent partitions P_1 P_2 in order to create a child individual C. This is achieved by only allowing contractions of nodes u, v when these nodes are not placed in different partitions in either parent. Afterwards we do not perform an initial partitioning, instead we consider the coarsened Hypergraph and see which of the parents gives the better objective on said graph. Since the contraction condition is rather strong and we do not need to do an initial partitioning, we can remove the coarsening limit s to allow for more contractions. The uncoarsening and application of local search algorithms which guarantee to atleast maintain solution quality in combination with using the better partition of the two parents ensures that the child solution is at least as good as the best parent solution. It is imporant to note that the combine operation is more powerful the

more diverse the parent individuals are. This will be important in the replacement strategy which will insert the child into the population.

4.5.2 Cross Combine

4.5.3 Edge Frequency Multicombine

4.5.4 Basic Combine + Edge Frequency Information

4.6 Mutation operations

4.6.1 VCycle

4.6.2 VCycle + New Initial Partitioning

4.6.3 Stable Nets

4.7 Replacement Strategies

5 Experimental Evaluation

5.1 Implementation

5.2 Experimental Setup

5.2.1 Environment

5.2.2 Tuning Parameters

5.2.3 Instances

5.3 Your Experiment Headline

6 Discussion

6.1 Conclusion

6.2 Future Work

A Implementation Details

A.1 Software

A.2 Hardware