Karlsruher Institut für Technologie

Bachelor thesis

Title of Thesis

Name of author

Date: 23. 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

A Hypergraph $H = (V, E, c, w)$ is defined as a set of verticies V a set of hyperedges E where each edge may contain an arbitrarily large subset of V. $c : V \rightarrow \mathbb{R}_{\geq 0}$ is a function applying a weight to each Vertex and $w : E \rightarrow \mathbb{R}_{\geq 0}$ applying a weight to each hyperedge. Two verticies u, v are adjacent if $\exists e \in E | u, v \in e$ and a vertex u is incident to a hyperedge e if $u \in e$. The size $|e|$ of an Hyperedge e is the number of verticies contained in e. A k-way partition of a Hypergraph H is a partition of V into k disjoint blocks $V_1, ..V_k$. $part : V \rightarrow [0, k-1]$ is a function referencing the corresponding block of a k-way partition to a vertex u. A k-way partition is balanced when the weight of each block $V_i | 1 \leq i \leq k, \sum_{v_i \in V_i} c(v_i) \leq (1+\epsilon) \lceil \frac{\sum_{v \in V} c(v)}{k} \rceil$ for a balance constraint ϵ. A valid solution is a balanced k-way partition. An invalid solution is a partition where either the balance criterion is not met, or H has been partitioned for a different value for k. A Hyperedge e is a cut edge $cut(e)$ if $\exists u, v \in e | part(u) \neq part(v)$. The connectivity of a Hyperedge e is $\lambda(e) = \sum_{i=0}^{k-1} \delta(e, i) | \delta(e, i) = \begin{cases} 1 & \exists v \in e \; part(v) = i \\ 0 & \text{else} \end{cases}$

The set cut edges in H is defined as $cut(E) := \{e \in E | cut(e)\}$. The multiset connectivity edges in H is defined as $conn(E) := \{a(e) \in E | cut(e)\} | a(e) := \lambda(e)$ The cut metric $cut(H) := \sum_{e \in E} \begin{cases} \text{w(e)} & \text{e is cut edge} \\ 0 & \text{else} \end{cases}$ and gives the value of cuts. The connectivity metric $(\lambda - 1)(H) := \begin{cases} \lambda(e) * w(e) & \text{e is cut edge} \\ 0 & \text{else} \end{cases}$ Both metrics can be used to measure the quality of a solution. Throughout this thesis the solution quality is referenced. The metrics are interchangeable in this regard. An Individual I is a valid solution for the k-way partition problem of H. An individual eligible for further operations is considered $alive$. The difference of two individuals I_1, I_2 is $diff(I_1, I_2) := cut(I_1) \ominus cut(I_2)$ The connectivity difference of two individuals I_1, I_2 is $strongdiff(I_1, I_2) := conn(I_1) \ominus conn(I_2)$ A population P is a collection of Individuals.

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 objective of the algorithm is to optimize solution quality of a k-way partition of H within bounded time. During this time the algorithm will operate on already existing solutions in order to generate improved solutions.

4.2 Population

The algorithm will produce multiple individuals, which are inserted and removed from the population. At any given time only a finite amount of individuals are $alive$ which is the maximum population size. Furhter individuals have to compete for a place in the population. The population size is an important parameter, as a small value limits the solution scope and a high value limits convergence. We use KaHyPar to fill the initial population. in order to select a proper population size for the runtime, we dynamically allocate 15% of the runtime to create and fill the initial population.

4.3 Diversity

By measuring the difference between two individuals we can gain some knowledge over their internal structure and similarities. using the edges rather than the verticies for difference ensures that only elements relevant for the solution quality are considered for difference and perturbations of the partition blocks are not influencing the difference. As such we can consider the diversity of the population as the difference of the current individuals. Therefore a low diversity is a population where the individuals are in close proximity considering the solution space or convergent. A high diversity means that the individuals are spread throughout a larger portion of the solution space.

4.4 Selection Strategies

For an evolutionary algorithm we attempt to generate new, improved solutions by using existing solutions. A logical conclusion is that good individuals probably will generate

solutions close to the original individual and as such good. We select our individuals using tournament selection

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 used by KaHyPar 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, since similar parents are essentially just a vcycle. 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