Abstract

Mixed-integer linear programming (MILP) plays a crucial role in the field of mathematical optimization and is especially relevant for practical applications due to the broad range of problems that can be modeled in that fashion. The vast majority of MILP solvers employ the LP-based branch-and-cut approach. As the name suggests, the linear programming (LP) subproblems that need to be solved therein influence their behavior and performance significantly.

This thesis explores the impact of various LP solvers as well as LP solving techniques on the constraint integer programming framework SCIP Optimization Suite. SCIP allows for comparisons between academic and open-source LP solvers like Clp and SoPlex, as well as commercially developed, high-end codes like CPLEX, Gurobi, and Xpress.

We investigate how the overall performance and stability of an MILP solver can be improved by new algorithmic enhancements like *LP solution polishing* and *persistent scaling* that we have implemented in the LP solver SoPlex. The former decreases the fractionality of LP solutions by selecting another vertex on the optimal hyperplane of the LP relaxation, exploiting degeneracy. The latter provides better numerical properties for the LP solver throughout the MILP solving process by preserving and extending the initial scaling factors, effectively also improving the overall performance of SCIP. Both enhancement techniques are activated by default in the SCIP Optimization Suite.

Additionally, we provide an analysis of numerical conditions in SCIP through the lens of the LP solver by comparing different measures and how these evolve during the different stages of the solving process.

A side effect of our work on this topic was the development of TreeD: a new and convenient way of presenting the search tree interactively and animated in the three-dimensional space. This visualization technique facilitates a better understanding of the MILP solving process of SCIP.

Furthermore, this thesis presents the various algorithmic techniques like the row representation and iterative refinement that are implemented in SoPlex and that distinguish the solver from other simplex-based codes. Although it is often not as performant as its competitors, SoPlex demonstrates the ongoing research efforts in the field of linear programming with the simplex method.

Aside from that, we demonstrate the rapid prototyping of algorithmic ideas and modeling approaches via PySCIPOpt, the Python interface to the SCIP Optimization Suite. This tool allows for convenient access to SCIP's internal data structures from the user-friendly Python programming language to implement custom algorithms and extensions without any prior knowledge of SCIP's programming language C. TreeD is one such example, demonstrating the use of several Python libraries on top of SCIP. PySCIPOpt also provides an intuitive modeling layer to formulate problems directly in the code without having to utilize another modeling language or framework.

All contributions presented in this thesis are readily accessible in source code in SCIP Optimization Suite or as separate projects on the public code-sharing platform GitHub.

Zusammenfassung

Die gemischt-ganzzahlige Programmierung (MILP) kommt bei der Modellierung und Lösung einer Vielzahl von verschiedenen mathematischen Optimierungsproblemen mit oft weitreichenden wirtschaftlichen Folgen zum Einsatz. Die am weitesten verbreiteten Löser für solche MILP-Probleme verwenden den LP-basierten *branch-and-cut*-Ansatz. Hierbei wird die Ganzzahligkeitsbedingung relaxiert und das sich daraus ergebende lineare Programm (LP) dazu verwendet, die Lösungsqualität abzuschätzen und den Lösungsraum schrittweise aufzuteilen und zu verkleinern.

In dieser Arbeit vergleichen und analysieren wir verschiedene algorithmische Techniken und Software-Implementierungen, die zur Optimierung dieser LP-Relaxierungen herangezogen werden können. Wir benutzen dazu die SCIP Optimization Suite, die es aufgrund ihrer modularen Struktur erlaubt, solche Vergleiche zwischen freien Lösern wie Clp oder SoPlex und kommerziellen Hochleistungscodes wie CPLEX, Gurobi oder Xpress durchzuführen.

Wir untersuchen speziell, wie die von uns entwickelten Methoden *LP solution polishing* und *persistent scaling* im LP-Löser SoPlex das Verhalten von SCIP bei der Lösung von MILPs beeinflussen. Das erstere Verfahren dient dabei der Verminderung der Fraktionalität der errechneten LP-Lösungen durch Ausnutzen von multiplen optimalen Lösungen, während das zweite zu einer Verbesserung der numerischen Eigenschaften beiträgt, indem die initialen Skalierungsfaktoren über den gesamten Lösungsprozess weiterverwendet werden.

Außerdem beinhaltet diese Arbeit eine Übersicht über sämtliche Eigenschaften und mathematischen Techniken, die im LP-Löser SoPlex implementiert sind und diesen von anderen vergleichbaren Implementierungen des Simplex-Algorithmus abheben.

Weiterhin präsentieren wir Ergebnisse von Studien zur numerischen Stabilität von SCIP während der unterschiedlichen Phasen des MILP-Lösens. Hierbei beleuchten wir die Stabilität aus der Perspektive des LP-Lösers und stellen mit dem von uns entwickelten Python Paket TreeD eine neue Möglichkeit vor, wie der Suchbaum interaktiv und animiert im dreidimensionalen Raum dargestellt werden kann. Diese Visualisierungstechnik eignet sich zur anschaulichen Darstellung des MILP-Löseprozesses von SCIP und kann somit zu einem besseren Verständnis dessen beitragen.

Darüber hinaus zeigen wir die schnelle und intuitive Erarbeitung algorithmischer Prototypen auf, die mit der von uns entwickelten SCIP Optimization Suite-Erweiterung PySCIPOpt möglich sind. Hiermit kann mittels der anwendungsfreundlichen Programmiersprache Python auf viele interne Datenstrukturen von SCIP zugegriffen werden, um in kurzer Zeit ohne C/C++-Kenntnisse neue Ideen zu implementieren, wie wir am Beispiel von TreeD zeigen. Auch die intuitive Modellierung ganzer Optimierungsprobleme ist möglich, ohne die zugrundeliegenden Daten in eine weitere Modellierungsumgebung zu überführen.

Sämtliche Entwicklungen und Ergebnisse sind entweder bereits in die SCIP Optimization Suite eingeflossen oder als separate Pakete über die Code-Plattform GitHub verfügbar und frei verwendbar.