Heuristiken im Branch-and-Cut-Framework SCIP

Timo Berthold

25. Oktober 2007

Diverse Problemstellungen aus dem Gebiet des Operations Research lassen sich effektiv als Gemischt-Ganzzahliges Programm (Mixed Integer Program, kurz MIP) formulieren. Obwohl das Lösen eines MIPs ein $\mathcal{NP}$-schweres Optimierungsproblem ist, können viele praktisch relevante Instanzen in kurzer Zeit optimal oder bis auf eine geringe Optimalitätslücke gelöst werden. Die Standardmethode zum Lösen von MIPs ist das Branch-and-Cut-Verfahren, eine Kombination aus Branch-and-Bound und Schnittebenenverfahren.

Heuristiken (Griechisch: εὑρίσκειν – finden) versuchen zulässige Lösungen zu konstruieren, es besteht jedoch keine Garantie, dass eine solche gefunden wird. Allerdings finden gute Heuristiken in der Regel deutlich schneller Lösungen als die oben genannten exakten Verfahren, und sie zeichnen sich durch eine vergleichsweise geringe Laufzeit aus. Dadurch kann die Zulässigkeit eines MIPs deutlich früher erkannt und suboptimale Teile des Branch-and-Bound-Baumes abgeschnitten werden. Des Weiteren ist in der Praxis oftmals eine "genügend" gute heuristische Lösung bereits ausreichend. Heuristiken sind daher ein wichtiger Bestandteil moderner MIP-Solver.

In das Branch-and-Cut-Framework SCIP [1, 3] sind insgesamt 23 Heuristiken integriert. Diese lassen sich grob in vier Kategorien unterteilen: Runde-, Diving-, Zielfunktionsdiving- und Large-Neighborhood-Search-Heuristiken (kurz LNS). Die einzelnen Konzepte werden wir im Folgenden darstellen und den Einfluss von Heuristiken auf die Gesamtperformance von SCIP erläutern.

Der entscheidende Punkt bei heuristischen Rundeverfahren ist, den LP-zulässigen Bereich während des sukzessiven Rundens der einzelnen fraktionalen Variablen nicht zu verlassen oder gegebenenfalls die LP-Zulässigkeit schnellstmöglich wieder herzustellen. In SCIP sind vier Rundeheuristiken [1, 4] verfügbar. Diese unterscheiden sich dadurch, ob die linearen Nebenbedingungen zwischenzeitlich verletzt werden dürfen und wie mit bereits ganzzahligen und mit kontinuierlichen Variablen verfahren wird.

Die Idee des Divings ist, den Branch-and-Bound-Baum durch eine Tiefensuche zu erkunden, um zu einem Knoten mit ganzzahligem LP-Optimum zu gelangen. Die sechs in SCIP implementierten Divingverfahren unterscheiden sich im Wesentlichen durch die verwendeten Branchingregeln, die allesamt stark auf Zulässigkeit zielen.

Der Ansatz beim Zielfunktionsdiving ist, die Zielfunktion so zu manipulieren, dass man eine ganzzahlige Lösung der LP-Relaxierung findet. Als Beispiel für ein solches Verfahren sei hier die Feasibility Pump genannt. Sie wurde 2005 von Fischetti, Glover und Lodi [6] präsentiert, 2007 haben Achterberg und Berthold [2] die Objective Feasibility Pump eingeführt. Diese Version findet durch

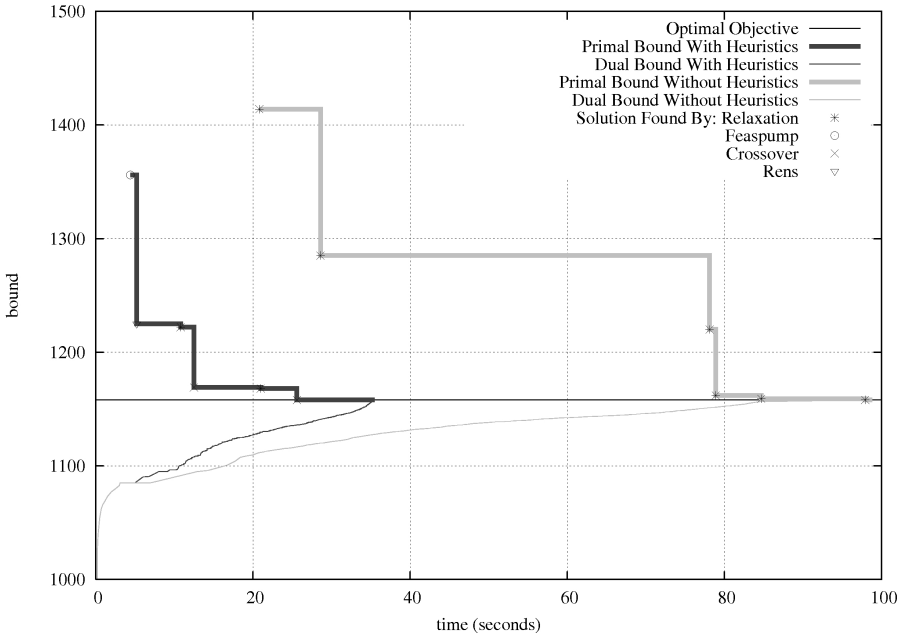

Abbildung 1: Instanz aflow30a: Performance von SCIP mit und ohne Heuristiken

das Einbeziehen der ursprünglichen Zielfunktion des MIPs erheblich bessere Lösungen in vergleichbarer Zeit.

LNS-Verfahren beruhen auf der Lösung eines deutlich kleineren Teil-MIPs, welches aus dem originalen MIP durch Fixieren von Variablen oder Hinzufügen von Constraints hervor geht. Dieses Teil-MIP repräsentiert dabei die Nachbarschaft eines ausgezeichneten Punktes, zum Beispiel der besten bisher gefundenen Lösung. In SCIP gibt es fünf LNS-Heuristiken: Vier davon sind Verbesserungsheuristiken, Rens [4, 5] hingegen nimmt das Optimum der LP-Relaxierung als Ausgangspunkt. Wird das Rens-Teil-MIP gelöst, erhält man eine nachweislich beste Rundung des LP-Optimums bzw. den Beweis, dass keine zulässige Rundung existiert. Damit bekommt man durch Rens ein Qualitätsmaß für Rundeheuristiken zur Verfügung gestellt.

Bezüglich der Gesamtperformance von SCIP zeigt sich, dass das Deaktivieren einzelner Heuristiken meist nur eine vergleichsweise geringe Auswirkung hat – bis auf eine Ausnahme weniger als 5% Veränderung der Laufzeit – wohingegen das Deaktivieren sämtlicher Heuristiken zu klaren Einbußen führt. Die Laufzeit und die Anzahl der benötigten Knoten steigt im Mittel auf das Doppelte, die bei schwierigen Instanzen verbleibende Lücke zur Optimalität steigt um ca. 50%. Für deutlich weniger Instanzen kann innerhalb einer Stunde die Optimalität bewiesen oder überhaupt eine Lösung gefunden werden. In Abbildung 1 ist exemplarisch der Verlauf von Primal- und Dualschranke mit und ohne Heuristiken bei einer der 129 Instanzen unserer Testmenge dargestellt. Die Ergebnisse unterstreichen die Relevanz von Heuristiken innerhalb eines Branch-and-Cut-Frameworks und belegen, dass insbesondere das Zusammenspiel verschiedener Heuristiken von Bedeutung ist.

Literatur

[1] T. Achterberg. *Constraint Integer Programming.* Doktorarbeit, Technische Universität Berlin, 2007.

[2] T. Achterberg und T. Berthold. *Improving the Feasibility Pump.* Discrete Optimization (Special Issue 4), Seiten 77-86, 2007.

[3] T. Achterberg, T. Berthold, M. Pfetsch und K. Wolter. *SCIP (Solving Constraint Integer Programs).* `http://scip.zib.de`

[4] T. Berthold. *Primal Heuristics for Mixed Integer Programs.* Diplomarbeit, Technische Universität Berlin, 2006.

[5] T. Berthold. *RENS – Relaxation Enforced Neighborhood Search.* ZIB-Report 07-28, 2007. `http://opus.kobv.de/zib/volltexte/2007/1053/`

[6] M. Fischetti, F. Glover und A. Lodi. *The Feasibility Pump*, Mathematical Programming 104(1), Seiten 91-104, 2005.