Zusammenfassung

In der vorliegenden Arbeit entwickeln wir Reduktionstechniken für kombinatorische Optimierungs- und Constraint Satisfaction Probleme, die in eine Baumsuche eingebettet werden können. In der kombinatorischen Optimierung werden hierfür üblicherweise Schranken und Variablenfixierung eingesetzt, wohingegen im Constraint Programming Filtrierungsalgorithmen den Suchraum beschneiden, indem sie Werte aus den Wertebereichen von Variablen entfernen. Ein wichtiges Ziel unserer Arbeit besteht darin, daß die entwickelten Algorithmen als "symbolische Constraints" in Constraint-Programming-Lösern und standard Optimierungssoftware wie etwa der SCIL-Bibliothek verwendet werden können. In dieser Hinsicht versteht sich die vorliegende Arbeit als ein Beitrag zur Entwicklung effizienter und einfach zu benutzender Optimierungssoftware.

Bei der exakten Lösung diskreter Optimierungsprobleme müssen in Wirklichkeit zwei Aufgaben gelöst werden. Erstens muß eine optimale Lösung konstruiert werden, von der dann zweitens die Optimalität nachgewiesen werden muß. Optimale oder zumindest fast optimale Lösungen können oft mit Hilfe von Heuristiken oder Approximationsalgorithmen gefunden werden, die jeweils auf das zu lösende Problem zugeschnitten sind. Im Gegensatz zum Finden einer Lösung mit sehr guter Qualität erfordert der algorithmische Optimalitätsbeweis die Betrachtung des kompletten Suchraumes, was im allgemeinen sehr viel aufwendiger ist als das bloße Durchsuchen einiger vielversprechender Regionen. Indem sie ganze Teile des Suchraumes abschneiden, können Problemreduktionstechniken einen wichtigen Beitrag zur Lösung beider Aufgaben leisten.

Die Arbeit gliedert sich in zwei Hauptteile. Teil 1 beinhaltet die Kapitel 2–4 und ist methodenorientiert. Das bedeutet, daß das Problem, eine bestimmte Konsistenz für einige ausgewählte Filtrierungsprobleme zu erreichen, theoretisch analysiert wird. Die Effizienz der entwickelten Algorithmen wird anhand ihrer Worst-Case-Komplexität und dem Grad an Konsistenz, den sie erreichen, gemessen.

Der erste Typ von Filtrierungsalgorithmus, den wir entwickeln, stellt eine spezielle Form einer symbolischen Constraint dar: In Kapitel 2 verfolgen wir das Ziel, einen Satz so genannter *Optimierungs-Constraints* zu entwickeln. Indem sie die Zielfunktion mit der Constraint-Substruktur eines Problems verknüpfen, können solche Constraints sowohl zum Abschneiden ganzer Teilbäume, als auch zur Wertebereichsreduktion von Variablen herangezogen werden, wobei letzteres auch als *kosten-basiertes Filtrieren* bezeichnet wird. Auf diese Weise verbinden Optimierungs-Constraints auf natürliche Weise die ausgezeichneten Optimierungsfähigkeiten des Operations Research mit den effizienten Modellierungs- und Filtrierungskonzepten des Constraint Programming.

Insbesondere studieren wir die Komplexität, wenn verschieden starke Grade von Konsis-

tenz für Optimierungs-Constraints erreicht werden sollen. Da sich das Problem, einen Zustand von Hyperkantenkonsistenz zu erreichen, sich für einige Optimierungs-Constraints als NP-hart herausstellt, führen wir einen neuen Konsistenzbegriff für Optimierungs-Constraints ein, die so genannte *relaxierte Konsistenz*. Basierend auf diesen beiden Konzepten von Konsistenz entwickeln wir effiziente kosten-basierte Filtrierungsalgorithmen für Kürzeste-Wege-Constraints (auf gewichteten, azyklischen Graphen, ungewichteten Graphen mit nicht-negativen Kantengewichten und gerichteten Graphen ohne negative Zyklen), gewichtete Unabhängige-Menge-Constraints auf Intervallgraphen, gewichtete All-Different-Constraints und Rucksack-Constraints. Diese Constraints sind dazu gedacht, um als grundlegende Bausteine bei der Modellierung realistischer diskreter Optimierungsprobleme verwendet zu werden. Indem sie das Wissen der zugrundeliegenden Constraint-Substruktur ausnutzen, können die zugehörigen Filtrierungsalgorithmen auf den für diese Strukturen bekannten Schranken und den effizienten Algorithmen zu ihrer Berechnung aufbauen.

Wie wir sehen werden, führt die lose Kopplung von Optimierungs-Constraints durch einfache Kommunikation via Wertebereichsreduktion zu wenig effektiver und daher auch wenig effizienter Problemreduktion. Wir stellen daher in Kapitel 3 eine Theorie vor, die die Verbindung von Optimierungs-Constraints durch die standard Dekompositionsmethoden Spaltengenerierung und Lagrange-Relaxierung motiviert.

Anschließend entwickeln wir einen zweiten Typ von Reduktionsalgorithmus, der mit seinen Entscheidungen nicht auf Kostenaspekten sondern allein auf der Constraint-Struktur basiert. Offensichtlich muß ein Suchknoten nicht weiter untersucht werden, wenn er faktisch mit einer bereits früher untersuchten Konfiguration übereinstimmt. Diese Situation begenet uns aber häufig, wenn Probleme behandelt werden, die Symmetrien aufweisen. In Kapitel 4 stellen wir daher eine Symmetriebrechnungsmethode namens SBDD vor, die auf der Erkennung von Dominanzbeziehungen zwischen Suchknoten beruht. Eine experimentelle Untersuchung zeigt, daß die Methode insbesondere hervorragend dazu geeignet ist, um hochgradig symmetrische Probleme anzugehen, die anderen Symmetriebrechungstechniken große Schwierigkeiten bereiten.

Der zweite Teil der Arbeit umfaßt die Kapitel 5–9 und ist anwednungsorientiert. Verschiedene Probleme der kombinatorischen Optimierung und des Constraint Satisfaction werden behandelt. The entwickelten Ansätze basieren auf den Algorithmen und Methoden des ersten Teils. Dies erlaubt zusätzlich zu den theoretischen Ergebnissen aus Teil 1 eine empirische Evaluation der zuvor entwickelten Reduktionsalgorithmen.

In Kapitel 5 betrachten wir zunächst das Airline Crew Assignment Problem. Der präsentierte Ansatz basiert auf der Idee des CP-basierten Spaltengenerierung in Verbindung mit Kürzeste-Wege-Constraints. Durch die Ausnutzung der besonderen Stärken von CP und OR kann die Berechnung realistischer Arbeitspläne bei Luftfahrtgesellschaften deutlich beschleunigt werden. Die hier vorgestellten Ideen wurden teilweise in ein industrielles Produktionssystem integriert und konnten dort drastische Einsparungen der Berechnungsdauer erzielen.

In Kapitel 6 betrachten wir das Automatische-Aufzeichnungs-Problem, das sich im Zusammenhang mit modernen Multimedia-Anwendungen stellt. Nachdem wir ein vollständig polynomielles Approximationsschema für das NP-harte Problem entwickelt haben, wird ein exakter Algorithmus vorgestellt, der Rucksack-Constraints und gewichtete Unabhängige-Menge-Constraints auf Intervallgraphen mit Hilfe der CP-basierten Lagrange-Relaxierung verbindet. Numerische Ergebnisse zeigen, daß unsere Implementierung effizient genug ist, um realistisch große Probleminstanzen in akzeptabler Zeit zu lösen.

Das kapazitierte Netzwerk-Design-Problem wird in Kapitel 7 behandelt. Untere Schranken können durch eine Dekomposition des Problems berechnet werden. Wir besprechen früher entwickelte Reduktionstechniken und zeigen, wie diese mit mit Hilfer CP-basierter Lagrange-Relaxierung verbunden werden können. Weiterhin präsentieren wir eine neue Technik, die lokal gültige Cuts generiert, die ebenfalls auf einer Lagrange-Relaxierung beruhen. Wir belegen experimentell, daß eine heuristische Variante unseres potentiell exakten Algorithmus in der Lage ist, in kürzerer Zeit bessere Ergebnisse zu erzielen als die besten bisher bekannten Heuristiken für das Problem

Ein neuer Ansatz für das Soziale-Golfer-Problem wird in Kapitel 8 vorgestellt. Indem wir SBDD und die neue Idee der unvollständigen Propagation redundanter Constraints integrieren, sind wir in der Lage, Lösungen für Instanzen zu berechnen, die zuvor außerhalb der Reichweite von Constarint-Programming-Lösern lagen.

In Kapitel 9 schließlich entwickeln wir einen exakten Löser für das Graph-Bisektions-Problem. Der Kern des Algorithmus besteht in einer Routine zur Berechnung unterer Schranken mit Hilfe der Approximation von maximalen Mehrsenken-Mehrgüterflüssen. Empirische Vergleiche mit einer früher entwickelten Schranke, die auf semidefiniter Programmierung beruht, belegen deutliche Verbesserungen sowohl in Bezug auf die Güte der Schranke als auch auf die Berechnungsdauer auf dünn besetzten, strukturierten Graphen. Insbesondere ist unsere Implementierung die erste, die die Bisektionsweiten von DeBruijn 9, Shuffle-Exchange 9 und Shuffle-Exchange 10 bestimmen konnte.