Constraint-Satisfaction-Probleme

B. Nebel, S. Wölfl
R. Mattmüller, M. Westphal
Wintersemester 2009/2010

Universität Freiburg
Institut für Informatik

Projekt P3

Abgabe: Mittwoch, 10. Februar 2010

In diesem Projekt soll es um das Lösen von Constraint-Optimierungs-Aufgaben gehen. Dabei sollen Instanzen des Problems des Handlungsreisenden mittels Branch-and-Bound und zum Vergleich mittels stochastischer lokaler Suche gelöst werden.

> *Hinweis:* Bei den Projekten geht es um die Implementierung, nicht so sehr um theoretische Aspekte. Wenn Sie an bestimmten Stellen nicht genau wissen, wie die Algorithmen und Definitionen aus der Vorlesung in die Praxis umgesetzt werden, können Sie daher gerne Fragen stellen — entweder per E-Mail oder in der Übungsgruppe.
>
> Projekte können in C, C++, Java oder Python bearbeitet werden. Andere Programmiersprachen sind nach Absprache möglich; in diesem Fall bitte vor Bearbeitung des Projekts bei uns melden.
>
> Die eingereichten Programme müssen die geforderten Ein- und Ausgabeformate verwenden, einige Tests bestehen und **ausreichend kommentiert** sein. Programme, die diesen Anforderungen nicht genügen, werden nicht akzeptiert, aber es besteht die Möglichkeit, innerhalb der Abgabefrist nachzubessern. Daher bitten wir darum, frühzeitig abzugeben, um ausreichend Zeit für Nachbesserungen zu haben.

Wir beschäftigen und in diesem Projekt mit dem Problem des Handlungsreisenden (TSP), also dem Problem, in dem ein Handlungsreisender n Städte besuchen muss und eine möglichst kurze Tour sucht, die alle Städte enthält. Formal sind n Knoten mit paarweisen Distanzen $d(i, j) \geq 0$ gegeben. Gesucht ist eine möglichst kurze Tour, also eine Permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, so dass die Länge

$$len(\pi) = \sum_{i=1}^{n-1} d(\pi(i), \pi(i+1)) + d(\pi(n), \pi(1))$$

von π minimal wird.
Dieses Problem kann als Constraint-Optimierungsproblem mit Variablen v_1 (erste besuchte Stadt) bis v_n (n-te besuchte Stadt), jeweils mit Domänen $\{1, \ldots, n\}$ (Stadt 1 bis n), mit einem weichen Constraint (Minimierung der Tourlänge $F_1(a) = \sum_{i=1}^{n-1} d(a(v_i), a(v_{i+1})) + d(a(v_n), a(v_1))$ einer Belegung/Tour a) aufgefasst werden. Der einzige harte Constraint ist $\mathit{alldifferent}(v_1, \ldots, v_n)$, d. h., dass jede Stadt genau einmal besucht wird.
Wir wollen annehmen, dass alle Distanzen zwischen Städten Euklidische Distanzen sind, dass es also ausreicht, in der Problemstellung die Städte als Punkte

in der Ebene anzugeben. Die Distanz zwischen zwei Städten mit Koordinaten (x_1, y_1) und (x_2, y_2) beträgt dann $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Aufgabe P3.1 (Vorarbeit, 0,5 Punkte)

Schreiben Sie einen Parser, der TSP-Instanzen in der folgenden Form einliest und die entsprechende Distanzmatrix für die spätere Verwendung aufstellt:

```
x1 y1
x2 y2
...
xn yn
```

Dabei gibt die erste Spalte die x- und die zweite Spalte die y-Koordinate der Stadt an. Alle Werte sind ganzzahlig.
Wenn Sie hier bereits die Datenstrukturen definieren, auf denen Ihr TSP-Solver arbeitet, können Sie bei der Gelegenheit auch die gewünschte Ausgabe implementieren: diese sollte in der ersten Zeile die gefundene Tour, beginnend bei Knoten 1, enthalten, etwa `1 5 7 2 8 4 3 6`, und in der zweiten Zeile die Kosten dieser Tour.

Aufgabe P3.2 (Branch-and-Bound-Suche, 1 Punkt)

Implementieren Sie Branch-and-Bound-Suche für das Problem des Handlungsreisenden. Da hier die Berechnung der first-choice evaluation function so teuer ist wie das Lösen des Problems selbst, sollten Sie sich eine geeignete TSP-spezifische Bewertungsfunktion ausdenken und implementieren.

Aufgabe P3.3 (Stochastische lokale Suche, 1 Punkt)

Bei schwierigeren Probleminstanzen versagt Branch-and-Bound-Suche, jedenfalls werden in vertretbarer Zeit keine erwiesenermaßen optimalen Lösungen gefunden. Deshalb wollen wir uns nun auch mit suboptimalen Lösungen zufrieden geben. Eine Möglichkeit, an suboptimale Lösungen zu gelangen, wäre es, die Branch-and-Bound-Suche vorzeitig abzubrechen und die beste bislang gefundene Lösung zurückzugeben. Für potentiell nur suboptimale Lösungen gibt es aber noch eine weitere Klasse von Algorithmen: lokale Suche.
Ein Weg, stochastische lokale Suchalgorithmen für TSP zu spezialisieren, ist die sogenannte 2-opt-Heuristik. In jedem Schritt werden aus der aktuellen Tour zwei Kanten entfernt und durch zwei neue Kanten ersetzt, welche die durch die Kantenentfernung entstandenen Fragmente wieder zu einer neuen, kürzeren Tour verbinden.
Implementieren Sie 2-opt mit random restarts und weiteren fortgeschrittenen Techniken Ihrer Wahl (siehe z. B. `http://en.wikipedia.org/wiki/Travelling_salesman_problem`).

Aufgabe P3.4 (Systematische Auswertung, 1 Punkt)

Wir haben einige TSP-Instanzen unterschiedlicher Schwierigkeit zusammengestellt, die Sie unter `http://www.informatik.uni-freiburg.de/~ki/teaching/ws0910/csp/tsp-problems.tar.gz` herunterladen können.
Sie können die folgenden Experimente mit Zeit- und Speicherbeschränkungen durchführen, die dem System, auf dem sie die Experimente durchführen, angemessen sind (z. B. 5 Minuten pro Instanz, 1 GB Speicher).

	Branch-and-Bound				Lokale Suche
TSP-Instanz	Zeit	Knoten	Kosten	opt?	Kosten
tsp_6					
tsp_8					
tsp_10					
tsp_12					
tsp_15					
tsp_20					
tsp_25					
tsp_30					

Tabelle 1: Vergleich von Branch-and-Bound mit lokaler Suche.

Evaluieren Sie die beiden Implementierungen (Branch-and-Bound, lokale Suche) anhand der gegebenen Instanzen und messen Sie alle benötigten Werte, um Tabelle 1 ausfüllen zu können (also benötigte Zeit, Anzahl erzeugter Knoten im Suchbaum und Kosten der Lösungstour bei Branch-and-Bound und Kosten der Lösungstour nach Timeout bei lokaler Suche). Sollte auch die Branch-and-Bound-Suche die Zeitbeschränkung nicht einhalten, geben Sie die beste bis dahin gefundene Lösung an. Kennzeichnen Sie in der entsprechenden Spalte, ob es sich um normale Terminierung mit bewiesenermaßen optimaler Lösung oder Timeout mit möglicherweise suboptimaler Lösung handelt.
Erstellen Sie ferner für ausgewählte Probleminstanzen Graphen, bei denen auf der x-Achse die Zeit und auf der y-Achse die Kosten der innerhalb dieser Zeit besten gefundenen Lösung aufgetragen sind, und die eine entsprechende (monoton fallende) Kurve für das Verhalten von lokaler Suche und von Branch-and-Bound enthalten.

Aufgabe P3.5 (Wettbewerb, 0,5 Punkte)

Wir werden die Abgaben auf weiteren Beispielinstanzen evaluieren und für die Gruppe einen halben Bonuspunkt vergeben, deren TSP-Solver im folgenden Sinne besten ist: Für jede innerhalb von 5 Minuten (optimal oder suboptimal) gelöste Instanz erhält jede Gruppe $len(\pi^*)/len(\pi)$ Punkte, wenn π die kürzeste von ihr gefundene Tour und π^* die kürzeste (bekannte) Tour überhaupt für das Problem ist. Die Gruppe mit den meisten Punkten gewinnt.

Bitte geben Sie an, in welcher Konfiguration (welcher Algorithmus, mit welchen Parametern, ...) wir Ihren Solver für den Wettbewerb aufrufen sollen. Alternativ können Sie auch eine Variante ihres TSP-Solvers mit hart-kodierten Parametern speziell für diese Aufgabe abgeben.