INF431

Homework on Branch-and-Bound for the TSP

Topic proposed by Leo Liberti

1 BB in Java

Design an implement a combinatorial Branch-and-Bound (BB) algorithm for the Travelling Salesman Problem (TSP) in Java. You can either design the code from scratch or use an existing template[1].

2 Mathematical programming

Mathematical Programming (MP) is a very general modelling technique that can be used on most practical optimization problems. Propose a mathematical program (i.e. a set of decision variables, an objective function, some constraints defininig the sought structure) for the following transportation problem.

Question 1 An Italian transportation firm should carry some empty containers from its 6 stores (in Verona, Perugia, Rome, Pescara, Taranto and Lamezia) to the main national ports (Genoa, Venice, Ancona, Naples, Bari). The container stocks at the stores are the following:

	Empty containers
Verona	10
Perugia	12
Rome	20
Pescara	24
Taranto	18
Lamezia	40

The demands at the ports are as follows:

	Container demand
Genoa	20
Venice	15
Ancona	25
Naples	33
Bari	21

Transportation is carried out by a fleet of lorries which can only carry one container each. The transportation cost for each container is proportional to the distance travelled by the lorry, and amounts to 30 euro / km. Distances are as follows:

[1] http://www.lix.polytechnique.fr/~liberti/teaching/dix/inf431-11/bbTSP-template.java

	Genoa	Venice	Ancona	Naples	Bari
Verona	290 km	115 km	355 km	715 km	810 km
Perugia	380 km	340 km	165 km	380 km	610 km
Rome	505 km	530 km	285 km	220 km	450 km
Pescara	655 km	450 km	155 km	240 km	315 km
Taranto	1010 km	840 km	550 km	305 km	95 km
Lamezia	1072 km	1097 km	747 km	372 km	333 km

Propose a MP formulation for devising a transportation plan that aims to minimize the transportation costs whilst satisfying the demand constraints at the ports and not exceeding availability at the stores. ◇

3 BB for TSP: tightening the MP bound

In general, MILP-based BB methods require two formulations during their execution: the *original* formulation, whose feasible solutions describe the sought mathematical structures, and the *relaxation*, which is usually like the original formulation without the integrality constraints. So, if the original formulation is a MILP, the relaxed formulation is an LP. The relaxation changes at each BB node α by constraining the variables of the disjunctions valid at α to the corresponding values. Once the LP relaxation is solved, a *relaxed solution* is obtained.

The CPU time taken by the BB algorithm to terminate is proportional to the number of nodes processed, so if the lower bound $\bar{c}$ increases its value, the chances for the event $\bar{c} \geq d^*$ to happen also increase. *Tightening*, or *improving*, the lower bound (or the relaxation) at a given BB node means applying techniques to increase $\bar{c}$.

MILP relaxations can be tightened by finding *violated valid inequalities*, i.e. inequalities (or equations) which are valid with respect to all feasible solutions of the original formulation, but which are violated by the relaxed solution. Adjoining such inequalities to the relaxation yields a tightened LP, whose solution is guaranteed to be different from the relaxed solution obtained previously. The tightened LP is still a valid relaxation for the original formulation (because the adjoined inequality is valid), but yields an improved lower bound (because it has more constraints than the previous relaxation and because it is invalid with respect to the relaxed solution).

As was shown in the PC material, the TSP can be formulated as follows:

$$\left.\begin{array}{rrcl} \min_{x,y} & \sum_{i\leq n}\sum_{\substack{j\leq n\\ j\neq i}} d_{ij}x_{ij} & & \\ \forall i \leq n & \sum_{\substack{j\leq n\\ j\neq i}} x_{ij} & = & 1 \\ \forall j \leq n & \sum_{\substack{i\leq n\\ i\neq j}} x_{ij} & = & 1 \\ \forall i,j \in \{2,\ldots,n\} \text{ with } i\neq j & y_i - y_j + (n-1)x_{ij} & \leq & n-2 \\ \forall i \leq n & y_i & \in & \mathbb{R} \\ \forall i \leq n, j \neq i & x_{ij} & \in & \{0,1\}. \end{array}\right\} \quad (1)$$

Consider solving Formulation (1) by BB, and let $\alpha = (Y, N)$ be some information at the current BB node, with Y, N such that $x_{ij} = 1$ for all $(i,j) \in Y$ and $x_{ij} = 0$ for all $(i,j) \in N$.

Question 2 Using the subtour elimination constraints:

$$\forall \varnothing \subsetneq U \subsetneq [n] \quad \sum_{\substack{i\in U\\ j\in[n]\setminus U}} x_{ij} \geq 1, \quad (2)$$

propose a technique for tightening the relaxation of Formulation (1) at a BB node α. ◇

Question 3 Find some other valid inequalities (apart from the subtour elimination inequalities (2)) for the TSP. ◇

4 Formulation-based BB for the Knapsack problem

Question 4 An investment bank has a total budget of 14 million euros, and can make 4 types of investments (numbered 1,2,3,4). The following tables specifies the amount to be invested and the net revenue for each investment. Each investment must be made in full if made at all.

Investment	1	2	3	4
Amount	5	7	4	3
Net revenue	16	22	12	8

Formulate a Binary Linear Program (BLP), i.e. a mathematical programming formulation involving binary decision variables and linear objective function and constraints) to maximize the total net revenue. ◇

Solving the problem by BB requires to branch on the (binary) variables specifying whether an investment is made or not.

Question 5 Suggest a way to compute an upper bound (we are maximizing!) to the problem using the BLP formulation and the greedy algorithm. ◇

Question 6 Carry out the BB computation by hand and exhibit the BB tree. ◇

References

[1] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover, New York, 1998.