Parallel Programming and Applications
P. Fritzson and L. Finmo (Eds.)
IOS Press, 1995

Parallel Cutting Plane Generation for the TSP

Thomas Christof
Institut für Angewandte Mathematik, Universität Heidelberg
Im Neuenheimer Feld 294, D-69120 Heidelberg, Germany

Gerhard Reinelt
Institut für Angewandte Mathematik, Universität Heidelberg
Im Neuenheimer Feld 294, D-69120 Heidelberg, Germany

Abstract. Recently, a complete linear description of the traveling salesman polytope on 9 nodes has been computed. In this report we discuss heuristic separation procedures for making use of these "small" facets in a general branch-and-cut-code for solving the traveling salesman problem. It will be shown that this approach is particularly suited for implementation on parallel hardware. Aspects of parallelization and computational results are discussed.

1. Cutting plane algorithms for the TSP

The traveling salesman problem, i.e. the problem of finding a shortest roundtrip through a set of cities, is one of the most prominent problems in the area of combinatorial optimization (see [2] for a survey). A well-established method for solving TSPs to optimality or for providing provably good solutions is the branch & cut approach([3]), which is a special variant of the branch & bound principle. The crucial part of a branch & bound algorithm is the computation of lower bounds (for minimization problems). In a branch & cut approach, lower bounds are obtained from linear-programming (LP) relaxations.

Let $K_n = (V_n, E_n)$ denote the complete undirected graph on n nodes. For $F \subseteq E_n$ and any $x \in \mathbb{R}^{E_n}$, $x(F)$ denotes the sum $\sum_{e \in F} x_e$. For a node set $W \subseteq V_n$, $E(W) \subseteq E_n$ denotes the edge set $\{uv \in E_n \mid u, v \in W\}$ and $\delta(W) \subseteq E_n$ denotes the edge set $\{uv \in E_n \mid u \in W, v \in V_n \setminus W\}$. We call $\delta(W)$ a cut with shores W and $V_n \setminus W$. A Hamiltonian cycle is a subgraph $H = (V_n, T)$ of K_n satisfying the following requirements

a) all nodes of H have degree 2,

b) H is connected.

In the following, we will call the edge set T Hamiltonian cycle. The solution set of the TSP is the set of all Hamiltonian cycles of K_n.

Given an objective function $c \in \mathbb{R}^{E_n}$ which associates a "length" c_e with every edge of K_n the TSP consists of finding a shortest Hamiltonian cycle in K_n. If we associate

with every edge e of E_n the variable x_e, then the TSP can be solved by finding a solution of the following integer linear program:

$$\begin{array}{lll} \min & cx & \\ \text{s.t.} & \delta(\{v\}) = 2, \text{ for all } v \in V_n & (1.1) \\ & x(\delta_n(S)) \geq 2 \text{ for all } \emptyset \neq S \subseteq V_n & (1.2) \\ & 0 \leq x_e \leq 1, \text{ for all } e \in E_n & (1.3) \\ & x_e \text{ integer, for all } e \in E_n & (1.4) \end{array}$$

A first linear programming relaxation, the subtour relaxation, is obtained by eliminating the integrality requirement (1.4). This relaxation can be tightened by adding further inequalities. Note, that already the subtour relaxation is formulated using an exponential number (in n) of inequalities. Better relaxations even need a much larger number of additional inequalities. Therefore, to solve the resulting linear programming problem, one follows the so-called cutting plane principle to avoid listing all necessary inequalities explicitly. A cutting plane algorithm starts with some initial relaxation, say consisting of (1.1) and (1.3) and then adds further inequalities as needed. More precisely, as soon as the optimal solution $\overline{x}$ of some LP relaxation is determined which is not integral, a separation algorithm tries to identify inequalities which are violated by $\overline{x}$, but satisfied by all integer solutions of (1.1) - (1.3). As long as such inequalities are found, they are added to the current relaxation and the new LP is solved etc.

Separation algorithms are known for the subtour elimination constraints (1.2), for comb-, clique-tree-, and path- inequalities ([4], [1]). These algorithms are able to exploit special structure of the inequalities.

However, for most classes of inequalities no efficient separation procedures are known. In this report we describe how further inequalities can be identified. The corresponding separation heuristics are based on complete linear descriptions of small TSP polytopes.

2. Small traveling salesman polytopes

With every Hamiltonian cycle $T \subseteq E_n$ we associate its incidence vector $\chi^T \in \{0,1\}^{E_n}$ defined by

$$\chi^T_e = \begin{cases} 1 & \text{if } e \in T, \\ 0 & \text{if } e \notin T. \end{cases}$$

The traveling salesman polytope is the convex hull of all Hamiltonian cycles

$$\text{STSP}(n) = \text{conv}\{\chi^T \mid T \subseteq E_n \text{ is a Hamiltonian cycle}\}.$$

The linear description of STSP(n) is given by (1.1), (1.2), (1.3) and many further inequalities. However, while it is known that no equations have to be added to those in (1.1), no complete knowledge of all necessary inequalities is available and it is unlikely that such a complete description can be given in general.

Using our software PORTA, which is based on the Fourier-Motzkin elimination method for eliminating variables in a system of linear inequalities, we computed the complete linear description of STSP(n) for $n = 3, 4, \ldots, 9$.

Every facet-inducing inequality can be transformed to a unique normal form. We say, two inequalities in normal form $fx \leq f_0$ and $gx \leq g_0$ belong to the same class if a permutation $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ exists such that $\sigma(f) := f_{\sigma(u)\sigma(v)} = g$.

Table 1: Facet structure of STSP(n)

n	Hamiltonian cycles	facets of STSP(n)	classes of facets
3	1	0	0
4	3	3	1
5	12	20	2
6	60	100	4
7	360	3,437	7
8	2,520	194,187	24
9	20,160	42,104,442	192

Due to the symmetric structure of STSP(n) many inequalities can be obtained by this way. Statistics on the facet structure of STSP(n) for n up to 9 are given in Table 1.

An important issue of any nontrivial STSP-facet is, that it can be extended by lifting operations ([5]). Let $fx \leq f_0$ be a valid inequality for STSP(n). For any given $h \in V_n$, define the clique-weight

$$\xi = \max\{f_{ih} + f_{jh} - f_{ij} \mid i, j \in V_n \setminus \{h\}, i \neq j\}.$$

Consider now the complete graph $K_{n+k} = (V_{n+k}, E_{n+k}) = (V_n \cup S, E_{n+k})$, $|S| = k$. We say that the inequality $f^*x \leq f^*_0$ for $STSP(n+k)$ is obtained by clique-lifting of $fx \leq f_0$ w.r.t. node h if $f^*x \leq f_0 + \xi k$ and

$$f^*_{uv} = \begin{cases} f_{uv} & u \in V_n, v \in V_n, \\ f_{hv} & u \in S, v \in V_n \setminus \{h\}, \\ \xi & u \in S \cup \{h\}, v \in S \cup \{h\}. \end{cases}$$

It can be proved that any inequality resulting from such a lifting operation is valid for STSP($n+k$). Moreover, if $k \geq 2$, then the extended inequality is facet-inducing, if $fx \leq f_0$ is facet-inducing.

By adding the equations of (1.1) in a suitable way, every facet-inducing inequality of STSP(n) can be transformed to a so called h-normal form with

a) $f \geq 0$,
b) $f_{uh} = 0$ for some $h \in V_n$, for all $u \in V_n$, $u \neq h$.

Note that an inequality in h-normal form which is facet-inducing for STSP(n) is also facet-inducing for STSP($n+k$), $k \geq 2$, (trivial-lifting).

3. Separation heuristics

We now describe a general separation heuristic for small facets.

Let $\overline{x}$ be the optimal solution of the current LP relaxation, where we assume that all equations (1.1) and all inequalities (1,2), (1.3) are satisfied. We define the Graph $(G, \overline{x}) = (V_n, E_n, \overline{x})$ by associating with every edge $e \in E_n$ the value $\overline{x}_e$ of its corresponding variable in the linear program. If $\overline{x}$ is fractional, we want to identify an inequality $fx \leq f_0$ that is valid or even facet-inducing for STSP(n) but violated by $\overline{x}$, i.e. $f\overline{x} > f_0$.

The heuristic basically works as follows.

In a first preprocessing step we shrink paths in $(G, \bar{x})$ that consist of edges e with $\bar{x}_e = 1$ to single nodes. Depending on the chosen strategy this procedure is performed repeatedly. We denote the shrunken graph by $(G', x') = (V', E, x')$.

The second step can be seen as the core of the procedure. Let $fx \leq f_0$ be a small facet-inducing inequality in h-normal form defined on k nodes, having $k - r$, $r \geq$ supporting nodes. In the graph (G', x') we search for a subset of $k - r$ nodes, such that a small facet-inducing inequality in h-normal form is violated on this subset of nodes. We assume that $|V'| \geq k$. First we construct a subset of nodes $W \subseteq V'$, with $|W| \geq k - r$ for which

$$\frac{x'(E(W))}{|W|}$$

has a large value in the following way:

(1) Choose a center node $v \in V'$, set $W = \{v\}$
(2) While $|W| \leq k - r$
set $W = W \cup \{v \in V' : \exists u \in W : x'_{uv} > 0\}$
(3) While $|W| \geq p$
(where p is some parameter $k - r \leq p \leq k - r + 5$)
set $W = W \setminus \{u\}$ with $\sum_{v \in W \setminus \{u\}} x'_{uv} = \min_{w \in W} \sum_{v \in W \setminus \{u\}} x'_{wv}$

Having computed W in this way, we choose an arbitrary subset $P \subseteq W$ of $k -$ nodes. We assign the $k - r$ nodes of P to the $k - r$ nodes supporting the facet and evaluate fx'. If $fx' > f_0$, a violated inequality is found, otherwise we try to find a better assignment of $k - r$ nodes to the nodes supporting the facet. This can be done by varying the assignment of P to the nodes supporting the facet or by replacing nodes of P by nodes of $W \notin P$. We have used improvement heuristics based on the change of two nodes (2-opt) and on the change of three nodes (3-opt). It turned out to be better not to stop the improvement as soon as a violated inequality is found, but to try to increase the left hand side further to obtain an inequality with higher violation.

If we have found a violated inequality we finally perform a lifting operation with the nodes that we have shrunk above. By the properties of the shrinking procedure a violated inequality in the shrunk graph corresponds to a violated inequality in the original graph.

4. Implementation/Parallelization

Our implementation uses the branch & cut code for solving the TSP described in [3] enhanced by the heuristics for small facet separation. The available hardware consists of a 4 processor PARSYTEC POWER XPLORER based on Power PC 601 processors using a SUN SPARCSTATION 20 as front-end.

The basic setup is the following. The core of the branch & cut code (basically the code of [3]) runs on the SUN front-end. Since the emphasis of the project is the evaluation of the usefulness of small facet separation, we decided to parallelize only the corresponding heuristics. There are, of course, further options for parallelization in a more general approach, but we do not elaborate on this.

Whenever the standard branch & cut code is not able to find an inequality separating the current fractional LP-solution, this solution is sent to each processor of the parallel machine. Every processor is responsible for a certain subset of the small facet

to be separated; in the experiments here, the 192 classes of facets of STSP(9) were distributed to the four processors such that each processor had to treat 48 of these classes. (In fact, some of the 192 classes are trivial or subtour elimination constraints. These were ignored.) For every class of inequalities a processor is responsible for, it executes the separation heuristic described in section 3. Violated inequalities are sent to the host. The host collects these violated inequalities and it halts separation on the parallel processors if enough violated inequalities (depending on the chosen strategy) were found. These inequalities are added to the current LP and the standard branch & cut code continues.

For communication between host and parallel machine we use several logical channels which are physically realized as files. Although a communication using files is not as fast as using standard software, like PVM, we decided to do so for two reasons. First we want to ensure easy portability of our code. Second, since every processor executes the separation heuristic for many facets, we have a very coarse granularity and the communication overhead is negligible. We use the following semaphore concept. Every processor of the parallel machine and the host is linked by a message and a data "channel". At any time the pair of "channels" is either owned by the host or by the processor. This means either the host or the processors are allowed to write to the associated files. In detail, after writing the LP-solution, the host switches the ownership. The right to write now stays at the processors of the parallel machine until they have finished their work. Using a third "channel", the host can send an interruption signal to the processors.

5. Computational results

The incorporation of the small facet separation into the branch & cut code for solving the TSP can be accomplished according to several strategies. In this section we report about first experiments.

The first question to be answered is whether it is useful at all to put effort into small facet separation. To this end, we ran the code described in section 4 on a number of publicly available TSP instances from TSPLIB ([6],[7]) with the following strategy. Every time the standard branch & cut code did not find any violated inequality, heuristics for small facets were executed. All violated inequalities identified by these heuristics were added to the current linear program. These additional inequalities should provide tighter LP relaxations than the standard version and, therefore, lead to a reduction in the number of nodes in the branch & cut tree.

Table 2 compares the new code with the standard code. Column 1 gives the name of the problem instance, column 2 contains the number of nodes in the branch & cut tree without small facet separation, column 3 gives the total number of violated inequalities found by our separation heuristics, and column 4 gives the number of tree nodes resulting when these inequalities are added. Note that each problem name contains the size of the problem, e.g. gil262 is a TSP with 262 cities.

The table displays an enormous reduction in the number of nodes, thus verifying that indeed tighter LP relaxations were obtained and that small facet separation proved to be useful.

However, with the first approach outlined here, CPU times could only be reduced in rare cases. In general, CPU times were higher compared to the standard version. This can be credited to the following facts.

Table 2: Computational results

problem	standard nodes	small facets cuts	small facets nodes
gil262	6	98	2
gr120	2	26	0
gr229	60	742	32
lin318	18	197	12
pr152	8	160	6
pr299	102	654	32
rat195	26	317	32
si175	458	1249	152
si535	394	543	84
si1032	2	7	2

- Since more cutting planes are added, the LPs to be solved get larger and more difficult.
- Although derived from "small" facets, the inequalities added are dense and therefore cause further difficulties for the LP solver.
- We used very general data structures for representing the new inequalities which were not optimized for the pricing routine. For this reason, also pricing time increased.
- The cutting plane generation strategy may not be the optimum one. Since the LPs tend to become harder with the new inequalities, one has to find out which ones and how many should be added.

But, in any case, our conclusion is that it is worthwhile to investigate this cut generation approach further. Improvement possibilities are, e.g., the following ones.

- Speed up of cut generation heuristics by using appropriate data structures.
- Development of further cut generation procedures.
- Among the many classes of small facets, identification of a subset of most useful ones that should be considered with priority.
- Development of appropriate cut generation strategies (with respect to number and type of inequalities that are added).

It should be emphasized that the approach discussed here is not limited to the traveling salesman problem. Due to its generality, it can be applied to other combinatorial optimization problems as well. Once a collection of small facets for some problem is known and lifting of these facets is possible, implementation of first heuristics as described above is fairly easy. In particular, if already available separation algorithms are not as sophisticated as for the TSP, improvements in the solution process should be significant. Moreover, the approach is particularly suited for making use of massively parallel hardware. In fact, it strongly depends on the availability of parallel

hardware, because otherwise the almost 200 separation heuristics would have to be run sequentially.

6. Acknowledgement

We are grateful to Dieter Homeister who helped us with his experience in parallel programming techniques.

References

[1] J.M. Clochard, D. Naddef (1993), "Using path inequalities in a branch and cut code for the symmetric traveling salesman problem", in: G.Rinaldi and L.Wolsey (eds.), *Integer Programming and Combinatorial Optimization 3*, CORE, Louvain-la-Neuve, 291-311

[2] M. Jünger, G. Reinelt, G. Rinaldi (1995), "The traveling salesman problem", to appear in: M. Ball, T. Magnanti, C.L. Monma & G. Nemhauser (eds.) *Handbook on Operations Research and Management Sciences: Networks*, North-Holland

[3] M. Jünger, G. Reinelt, S. Thienel (1994), "Optimal and provably good solutions for the symmetric traveling salesman problem", *Zeitschrift für Operations Research*

[4] M.W. Padberg, G. Rinaldi (1990), "Facet identification for the symmetric traveling salesman problem", *Math. Programming 47*, 219-257

[5] M. Queyranne Y. Wang (1993), "Hamiltonian path and symmetric traveling salesman polytopes", *Math. Programming 58*, 89-110

[6] G. Reinelt (1991), "TSPLIB - A Traveling salesman Problem Library", *ORSA Journal on Computing 3*, 376-384

[7] G. Reinelt (1995), "TSPLIB 95", *Report Universität Heidelberg*