Semester Report SS03 of Heiko Schilling

Name:	Heiko Schilling
Supervisor:	Prof. Dr. Rolf H. Möhring
Field of Research:	Graph Algorithms
Topic	Algorithms on large and complex Networks
PhD Student	associated member since July 2001

Field of Research - Results and Preview

Software techniques. In this semester we proceeded in developing our framework for the implementation of complex nested algorithms. In particular we considered here the maximal flow algorithm by Garg, Könemann, [GK98] and our LP–based approach for solving length–constrained multi–commodity flow problems with load–dependend transit times, where constrained shortest path algorithms are used as subroutines, [JMS99].

The key idea of this framework are *template classes*, i.e. classes that may be parameterized by one or more of the following: types, constants of integral builtin types, or even other templates. At compile–time, each instantiation of the template with different arguments causes a separate compilation *(implicit specialization)* of the classes source code with the parameters replaced by the arguments. These software techniques are usually called *generic programming*, [MS94].

In the context of generic programming, the term *concept* is used to describe the collection of requirements that a template argument must meet for the function or class template to compile and operate properly. Hence the specification of a concept for a template argument type takes the role that a type has in non–generic code. For non–concept–conform template arguments so called *Traits classes*, [Mye95], are introduced. They are employed as mappings from types to auxiliary class types that provide the associated types/constants as members in order to meet the specification requirements.

By using these *traits class* techniques we are designing graph algorithms which are independent of the underlying graph data structure. By providing a traits class called `GraphView` for every applied graph data structure we ensure that specific algorithm requirements are met by every particular graph data structure. Furthermore we are able to provide modified graph views of the original one in case additional information is needed. As for our time–expanded graph model, [KLS02], by plugging in an additional time parameter as template argument we are able to provide a specific `TimeExpandedGraphView` for our algorithms build on the aforementioned original `GraphView`.

It is possible to extend this approach to routines and algorithms as well. If a main algorithm uses some subroutine which is described in an abstract way as it is the case for a maximal flow algorithm by Garg, Könemann. Here the main algorithm `MaxFlow` computes a maximum flow by iterative calling some `UpdatingFlow`–algorithm. This sub–algorithm computes a flow fulfilling side constraints, e.g. the cost of this flow has to be below some bound. Thereby the cost function and the lower bound are given by the main algorithm. There is a variety of possible algorithms for this updating flow computation, e.g. a series of shortest path computations w.r.t. to the given cost function. The important point is, that the main algorithm works with any sub–algorithm which fulfills this abstract specification and this is the point where the generic programming approach comes into play. By formulating the used sub–algorithm as a template argument we are able to implement this maximal flow algorithm by Garg, Könemann as a template class, i.e. in a very abstract way. The template argument is a algorithm generator which is able to provide a concrete implementation of some updating flow algorithm.

Of course, this abstract algorithm formulation still leaves room for improvements and creativity. As an example consider the following setting in the above mentioned maximal flow example. In order to create an instance of a `MaxFlow` object it is necessary to create first all members of this class and in particular to create an instance of the `UpdatingFlow` member object. This member object on the other hand needs access to data within the main `MaxFlow` object (cost function, cost bound), which is yet not accessible since the main object is not created until all members are created.

To cope with this problem we introduced a so called *reverse cast* which moves a pointer from a data member of a class to the containing class object and therefore obtaining access to the main object at running time. There are even more general methods which could be applied here, like the so called Barton-Nackman-Trick [BN97].

Constrained shortest path algorithms. In our aforementioned LP–based approach for solving multi–commidity flow problems the repeated constrained shortest path computations require up to 70% of the overall computation. Therefore we investigated acceleration methods for these algorithms. For this we used standard techniques for usual shortest path problems like destination–oriented search and bi–directional search.

With the destination–oriented approach it is tried to accelerate the path computation by the use of lower bounds to the distance to the respective target node. By a modification of the edge weights one forces the Dijkstra algorithm to prefer nodes, which are closer to the target node. This approach gives the best results so far.

In the case of the bi-directional approach, where at the same time shortest paths are computed from start node and target node, the problem of finding a satisfiable stop criteria could not be solved. It gets even worse if one combines this method with the destination–oriented approach.

The run time behavior of the combined acceleration procedures is by far more favorable, if one looks for doubly constrained paths instead of shortest length–constrained paths. This kind of paths is in particular relevant for column generation methodes for multi–commodity flow problems, where improving columns are given by such doubly constrained paths.

Implizit representation of time–expanded networks. This year on our yearly colloquium of the DFG key program a potential approach was discussed for the implicit representation of time–expanded networks, which should be further pursued. It concerns the representation by means of so called „Ordered Binary Decision Diagrams“ (OBDDs). By the working group of Ingo Wegener (Dortmund University) was an implicit maximal flow algorithm introduced for 0-1 networks, which requires on grid network works polylogarithmic running time $O(log^4 n)$. The interesting question here is, in what respect these techniques are applicable to further practically relevant graph classes. Here it is to clarify whether the specific structure of our time–expanded networks permits a reduced representation by means of OBDDs. During my first stay at the Dortmund group we found a very simple OBDD–representation of time–expanded 0-1 networks of size $O(|V|^2 * T^2)$, where T is the time horizon. In cases where the transit times τ_e for each edge e can be represented in a compact way this can be reduced to $O(|V|^2 * log(max(|V|, T)))$.

Activities

- *Monday Colloquia of the CGC Graduate Program.*
- Jahreskolloquium of the DFG project „Algorithms on Large and Complex Networks“, Tübingen, March 26–28, 2003.

- Research stay at the Ingo Wegener group Dortmund, April 21–22, 2003.
- Summer School on „Algorithms for Hard Problems“, Lugano , Switzerland, May 19-23, 2003.
- 2nd International Workshop on Experimental and Efficient Algorithms (WEA), Monte Veritá, Ascona, Switzerland, May 26 - 28, 2003.
- Biweekly meetings of the traffic research group in the Möhring group.
- Support of implementation work of several students working on their diploma thesis in the Möhring group.

Literatur

[BN97] J. J. Barton and L. R. Nackman. *Scientific and Engineering C++*. Addison-Wesley, Reading, MA, 1997.

[GK98] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 300–309, Palo Alto, CA, 1998.

[KLS02] Ekkehard Köhler, Katharina Langkau & Martin Skutella, *Time-Expanded Graphs with Flow-Dependent Transit Times*, in Proceedings of the 10th European Symposium on Algorithms (ESA'02), to appear.

[JMS99] Olaf Jahn, Rolf H. Möhring, and Andreas S. Schulz, *Optimal routing of traffic flows with length restrictions in networks with congestion*, Tech. Report Report 658-1999, TU Berlin, 1999.

[MS94] D. R. Musser and A. A. Stepanov. Algorithm–oriented generic libraries. *Software: Practice and Experience*, 24(7):632–642, July 1994.

[Mye95] N. Myers. A new and useful template technique: „Traits“. *C++ Report*, 7(5):32–35, June 1995.