CHAPTER VII

CONCLUSION AND FUTURE RESEARCH

This chapter concludes the thesis by highlighting our contributions in Section 7.1, and then suggesting some potential directions for future research in shortest path, multicommodity flow and their related problems in Section 7.2.

7.1 Contributions

This thesis contains extensive surveys of both the applications and solution methods of multicommodity network flow problems. The previous survey papers in multicommodity network flow were written more than two decades ago. During the past twenty years, many new methods and applications in the field have been proposed and researched. We survey over 200 references and summarize the applications in Chapter 1, and solution methods in Chapter 2.

The shortest path problem is a classic combinatorial optimization problem. It is considered to be easy but is very important because it appeared as a subproblem in many difficult problems. For example, solving origin-destination multicommodity network flow (ODMCNF) problems by the arc-path formulation will usually require extensive computations of shortest paths between multiple pairs of nodes.

Although the shortest path problem has been researched for more than 50 years, to the best of our knowledge, there exist no combinatorial algorithms designed specifically for solving multiple pairs shortest path problems (MPSP), until now. In Chapter 3 we survey over 100 references and summarize most of the shortest path algorithms in the literature, discuss their pros and cons, and then demonstrate that a new method called the Least Squares Primal-Dual method (LSPD), when used to solve the 1-1 and ALL-1 shortest path problems with nonnegative arc lengths, performs identical steps to the "classic" Dijkstra's algorithm. We also discuss the relationship between LSPD, Dijkstra, and the original primal-dual algorithm in solving the 1-1 and ALL-1 shortest path problems.

Inspired by an APSP algorithm proposed by Carré [64, 65], in Chapter 4 we propose two new MPSP algorithms, called $DLU1$ and $DLU2$, which are the first shortest path algorithms designed specifically for solving multiple pairs shortest path problems. They do not need to compute a single source shortest path (SSSP) tree multiple times, nor do they need to compute all pairs shortest paths (APSP) as many algorithms in the literature do. In particular, $DLU2$ can specifically compute the shortest paths or distances for the requested OD pairs without wasting time on computing other unrequested OD entries. Our algorithms are applicable to cases with negative arc lengths but not negative cycles. Like the Floyd-Warshall algorithm, our algorithms can also detect negative cycles.

For real world problems where the network topology is fixed but arc lengths or requested OD pairs are variable, our MPSP algorithms may take more advantage of similarities than other shortest path algorithms. In particular, our algorithms rely on a good node pivot ordering, which can be computed in advance by many fill-in reducing techniques used for the sparse Gaussian method in numerical linear algebra. For problems with fixed topology, this special node ordering needs to be determined only once, to produce an "ad hoc" efficient shortest path algorithm that is designed specifically for the problem topology. This property is advantageous, for problems such as ODMCNF in which shortest paths between multiple pairs of OD nodes have to be repeatedly computed for the same network topology but different arc lengths.

We show correctness and give the theoretical complexity of our algorithms. Although theoretically both $DLU1$ and $DLU2$ have $O(n^3)$ time bounds, we expect practical efficiency because of the fill-in minimizing techniques from numerical linear algebra.

Chapter 5 gives a detailed introduction to a sparse implementation, SLU, of Carré's algorithm. We have performed many computational experiments to compare several implementations of our algorithms $DLU1$, $DLU2$, and SLU, with other 5 label-correcting and 4 label-setting SSSP shortest path codes [74] which are considered to be the state-of-the-art. We have done thorough computational tests on 19 problem families of MPSP problems, each with 4 different percentages (25%, 50%, 75%, and 100%) of OD pairs requested. This is the first computational study in the literature for solving the MPSP problems. Unfortunately, our proposed algorithms do not perform more efficiently than most of the state-of-the-art shortest path codes for the cases we tested. However, it does give evidence that our MPSP algorithms perform absolutely better than Floyd-Warshall algorithms for solving MPSP problems. Also, on a real flight network in the Asia-Pacific region which contains 112 nodes and 1038 arcs, our new MPSP algorithms do appear to be competitive, and are even faster than many of the state-of-the-art SSSP codes. Our MPSP algorithms especially take advantage of problems where the distribution of OD pairs correspond to a matching in the OD matrix. That is, our MPSP algorithms perform relatively better for MPSPs with n requested OD pairs (s_i, t_i) where each node appears exactly once in both of the origin and destination sets but not at the same time. In this case, most of the known shortest path algorithms have to compute shortest paths for all pairs but our algorithms only compute the necessary shortest paths rather than solving the APSP problem.

Finally, in Chapter 6 we propose a new primal-dual method named the "primal-dual key path method" to solve ODMCNF problems. We describe the details and give two methods to compute the step length θ^*. We also explain how primal and dual degeneracy may affect our algorithms and propose perturbation methods to resolve the degenerate pivots and cycling. In limited computational tests on 17 test problems, we compare our primal-dual key path method (KEY) with the generic primal-dual method (PD), Dantzig-Wolf decomposition method (DW), and the CPLEX LP solver that solves the node-arc (NA) formulation. We conclude DW to be the most efficient implementation and give reasons why primal-dual based algorithms (PD and KEY) may perform worse than DW. Our computational results also confirm the fact that KEY is more suitable for cases with many OD commodities. Also, we discover that generating a single shortest path, rather than all shortest paths, for each commodity is good enough for the DW method.

Further breakdown of the total running time for PD, KEY and DW reconfirms the importances of a good MPSP algorithm in speeding up those path-based algorithms. In particular, up to 85% total running time of PD, 95% total running time of KEY and 55% total running time of DW may be spent on shortest path computations. Thus designing an efficient MPSP algorithm, as we have researched in Chapter 3, 4, and 5, is very important.

7.2 Future research

In this section, we propose some interesting problems for future research.

7.2.1 Shortest path

- Although we have proposed new MPSP algorithms, in our algorithms, sequence of operations is based on the node ordering, and does not consider the affect of individual arc lengths as most SSSP algorithms do. A good research topic is integration of arc length into our algorithm. In particular, it may be possible that some triple-comparisons can be avoided, and such techniques may speed up our algorithms.

- In Chapter 5, we chose test problems based on the computational experiments on SSSP algorithms by Cherkassky et al. [74]. Although the computational results are not so encouraging, it is possible that our MPSP algorithms may perform better on some specific classes of graphs. Complete graphs, for example, are a class of networks in which our MPSP algorithms should run faster than other algorithms not only theoretically, but also computationally. Note that our algorithms may have worse performance for larger networks since they require more memory, and accessing so much memory will slow down their performance. Hub-and-spoke networks may also be a class where our algorithms are advantageous since a careful node ordering which permutes the hub node and its nearby nodes to larger indices will avoid many fill-ins.

  What classes of graphs may be suitable for our MPSP algorithms? Why do our algorithms run so quickly for the Aisa-Pacific flight network? What are the factors that slow down or speed up our algorithms? Answers to these questions may lead us to design new and better MPSP algorithms, or help us to identify appropriate applications for our MPSP algorithms.

- Theoretically, the running time of a shortest path algorithm should be proportional to the total number of triple comparisons. However, due to the computer architecture, the running time may be "nonlinearly" affected by the memory caching and accessing which makes the algebraic algorithms less efficient for larger network. Another good research is to study the relationship between the total running time and total number of triple comparisons, or, to benchmark algorithms using total number of triple comparisons, instead of using the practical running time.

- Our MPSP algorithms are inspired by the Gaussian method applied in a path algebra system. In other words, computing shortest path distances for q OD pairs (s_i, t_i) is identical to computing the q specific entries $x_{s_i t_i}$ of the matrix $X = (I_n - C)^{-1}$ where C is the measure matrix and I_n is the identity matrix (see Section 3.4).

  Using similar arguments to the numerical linear algebra, we should be able to give a "twin" algorithm similar to our MPSP algorithms which compute q specific entries for some matrix B^{-1} without inverting the whole matrix B (as does the APSP algorithm), and without computing whole columns of the matrix that cover the requested OD pairs (as does by SSSP algorithm). We even do not need to compute the entries $B^{-1}_{n,j}, \ldots, B^{-1}_{i+1,j}$ to obtain the entry $B^{-1}_{i,j}$ (as does by Carré's algorithm). It remains an open question whether efficiently computing specific entries for the inverse of a matrix is difficult in most cases.

- We have discovered the new Least Squares Primal-Dual method (LSPD), when used to solve shortest path problems with nonnegative arc lengths, actually corresponds to the Dijkstra's algorithm (see Section 3.6). What happens in the case of negative arc lengths? Will LSPD perform better than other SSSP algorithms, and in which cases?

  In our survey and computational tests, the LP-based methods such as primal simplex methods, dual simplex methods, or primal-dual simplex methods are usually considered to be practically less efficient than the combinatorial label-setting and label-correcting methods. LSPD can also be classified as a LP-based method. Although it has special properties such as being imperious to degenerate pivots, whether it will solve MPSP faster or solve SSSP faster for cases with negative arc lengths require more investigation.

7.2.2 Multicommodity network flow

- Our limited computational tests show that our primal-dual key path method (KEY) is not very competitive. However, more tests should be done, especially using the problems with many OD commodities of which KEY can take more advantage.

- Both KEY and PD are primal-dual based algorithms, and perform worse than DW. From our computational results, we think the key to speeding up KEY and PD lies in the techniques used to obtain optimal dual solutions of the RPP. In particular, there usually exist multiple optimal dual solutions of the RPP. Among these multiple choices of dual improving directions, which one might help to reduce the total number of primal-dual iterations? Which improving direction might reduce the number of degenerate pivots? If we can reduce the number of primal-dual iterations, we should save much time in shortest path computations.

  The LSPD may be a good direction to study, since it is also primal-dual based and can avoid degenerate pivots. More research can be done in applying LSPD to solve the RPP, either in the node-arc form or in the arc-path form. The reason we cite the node-arc form here is because LSPD already has had success in solving the single commodity min-cost network flow problem [149, 30]. In that case, the node-arc form of the single commodity problem helps to shorten the time spent in least squares computations. We suspect similar methods may also be available for the multicommodity cases. We will continue to investigate the possibility of applying LSPD in solving the ODMCNF problems.

- Some new methods such as the volume algorithm of Barahona and Anbil [24], as introduced in Section 2.3.2, have not yet been applied to solve ODMCNF problems. We think it is a good research direction to try this new algorithm and compare its performance with other similar algorithms such as bundle methods and subgradient methods.

- Although DW has been shown to be very efficient in our tests, other research in the airline crew partitioning and cutting-stock problems [169] shows that a primal-dual subproblem simplex method might also be competitive. Since we already develop techniques for generating all of the shortest paths for our primal-dual algorithms, it should not be difficult to generate all of the paths with length within a small threshold value of the length of shortest paths, as required for the primal-dual subproblem simplex method to proceed. However, one disadvantage of this method is the necessity of path index bookkeeping, as encountered in our PD implementation.