CHAPTER 1
INTRODUCTION

1.1. Time - The Fourth Dimension

1.1.1. The Role of ITS in Managing Traffic Congestion

Time is the essence of today's ever mobile world. While long distance and inter-continental travel time seem to be getting shorter each year, daily commuters spend more and more time just getting to their office. One main reason for such a situation is traffic congestion. Traffic congestion is perhaps the most conspicuous problem on roads and one that needs immediate attention. Analysts conservatively estimate that traffic congestion results in a loss in the order of billions of dollars every year. Intelligent Transportation Systems (ITS) help in this direction and a large part of today's ITS scheme is devoted to studying methods that can mitigate this problem. Advanced Traffic Management Systems (ATMS) and Advanced Traveler Information Systems (ATIS) are fields[1] in ITS that have congestion management as a major priority. ITS does have a vested interest in this task. If they can demonstrate their success in managing traffic congestion to public satisfaction, then their global ITS plan would receive greater approval. The success of ITS would usher in a new era in transportation, possibly the most significant one after the introduction of the inter-state highways. Hence, it is clear that congestion management has wide-reaching economic, social and political implications.

1.1.2. The Role of Operations Research in Transportation Engineering

Operations research has always played a major role in transportation engineering for all modes of transportation. Airline networks regularly use large scale optimization models for scheduling. Commercial trucking companies use fleet management techniques. Shipping companies use similar methods. System Optimal and User Optimal Traffic assignment methods have been successfully applied in managing traffic and

[1] ATMS and ATIS (currently) are major components of the ITS architecture that are devoted to traffic management.

reducing traffic congestion. While it has been agreed that there is no perfect solution for traffic congestion within the present framework, it is possible to reduce congestion to more tolerable levels. Although past methods have mainly addressed the problem of the feasibility of a plan, modern operations research methods can go one step further and obtain optimal solutions. Such a task needs sophisticated mathematical methods and software. Network Optimization is a specialized field in operations research that deals with optimal network flows. Typical applications include maximal flow, minimum cost flow, shortest path and assignment algorithms. Methods that involve congestion management are traffic assignment, traffic routing, congestion pricing, etc. Among these methods, traffic routing would be an immediate priority as it forms the core of the overall assignment plan. Today's city routes with ever-changing traffic patterns lead to dynamic networks. The state of such a network varies with time, and can be fully described only by using all four dimensions (x, y, z, t), where dimensions (x, y, z) represent the three space coordinates, and where t represents the time dimension. To analyze this situation, one would need to use dynamic algorithms that can take into account this time-dependent behavior. This leads to the concept of dynamic network optimization and real-time traffic routing.

1.2. Dynamic Traffic Routing

1.2.1. Past and Present Approaches to Routing

It would be interesting to compare traffic routing in the past, present and in the near future. In the past, drivers just had to queue up and wait until the congestion cleared. Analysts were content with just studying the queuing time and predicting waiting times, but made no attempt to actually solve the problem. Today's traffic diversion methods are oriented toward a "local" optimum, i.e., a point-to-point diversion that diverts traffic around the point of congestion. Although this method may work in simple cases, where traffic flow is relatively low and there are no other areas of congestion in the vicinity, this method may just result in the shifting of the congestion spot to another place in many cases. Static and dynamic shortest path algorithms can help users take anticipatory action

based on the detection of congestion upstream and thus represent a solution closer to a "global" optimum. In the near future, dynamic algorithms may help in achieving better routing schemes by prescribing optimal policies over both time and space. Indeed, incorporation of time-varying parameters in computations is an important feature of these algorithms. This feature also encourages the possibility of deploying these algorithms in some real-time traffic routing software.

1.2.2. The ATIS Scenario

The three main components which are present during ATIS operations are the infrastructure, controller(s) and users of the system [Koutsopoulos and Xu, 1993]. In the ATIS scenario, the design parameters would involve the **intelligence** of the system, the **frequency** of information updates and their **locations**. The intelligence of the system determines the **effectiveness** of the system. The frequency of information updates determines the **responsiveness** of the system. Finally, the locations of these information nodes determine the **adaptiveness** of the system.

1.2.3. Example

Consider a simple two-route network shown in Figure 1. We can use simple traffic flow equations to construct a mathematical model of the traffic in the system [Drew, 1968]. We can use this example to highlight all the routing issues addressed in this study.

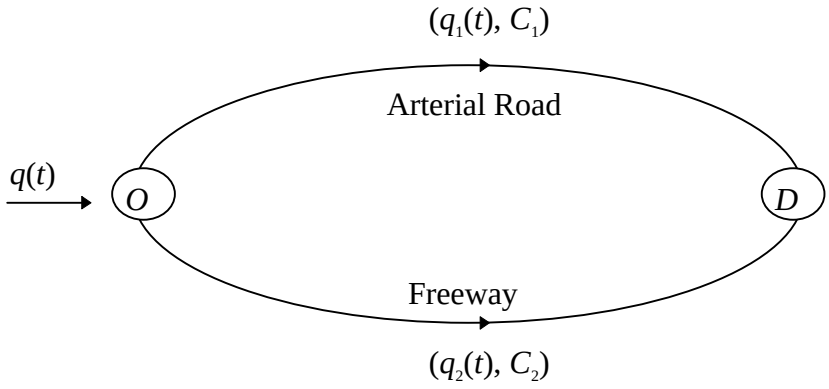

Figure 1. A simple Two-Route Case that Illustrates Routing Issues.

Traffic moves from a residential area to some downtown area. The capacity of the freeway is C_2, and the capacity of the arterial road is C_1. The capacities of alternate *O-D* paths is negligible. Initially, as the traffic input, $q(t)$, increases, the freeway is capable of accommodating traffic and no diversion is required. Gradually as traffic flow increases and the ratio $q_2(t)/C_2$ increases, a portion of the traffic, $q_1(t)$, must be diverted to the arterial road. The system stabilizes, provided we use some stable routing algorithm that partitions the traffic input accurately (for example, we can stipulate that the flow to capacity ratios be equal for the paths). The problem occurs when a bottleneck is created on the freeway either due to high traffic concentration or some incident that reduces the effective capacity to a smaller value C'_2. Traffic would continue to pour onto the freeway if this information is not carried downstream immediately. Also, the flow input $q(t)$ at the origin may exceed the new capacity of the system ($C_1 + C'_2$), and either flow control will be needed at *O* to reject the excess load or multiple *O-D* paths will need to be found. The second problem is the use of an unstable algorithm to divert traffic. This would result in oscillation of congestion between the two paths. The third problem is the actual result of informing drivers to change their routes. Some problems associated with this are concentration and over-reaction [Koutsopoulos and Xu, 1993].

For the simple case of constant traffic input $q(t) = q$, a small perturbation of q_1 would ideally result in a damped oscillation of flow/capacity values, eventually leading to equilibrium. For more complex cases, with multiple routes and dynamic flows, such an equilibrium may not be reached and oscillations may persist. Also, we need to find the paths having minimum costs over time. If a dynamic, adaptive and stable routing algorithm is used, this undesirable oscillation can be minimized and overall stability of the system is not affected much.

1.2.4. Time-Dependent Shortest Path Algorithms

Real-time routing algorithms and dynamic traffic assignment procedures increasingly use time-dependent shortest path (TDSP) algorithms. Sometimes multiple

time-dependent shortest paths may also be needed. This study is devoted to studying, classifying, implementing and testing TDSP algorithms. In addition, we will discuss some modifications introduced in these algorithms in order to implement these algorithms in incident management software packages. A schema for the overall software design is also presented. The motivation for this study and a discussion of analogous routing procedures in data networks is described in the next section. A detailed classification and study of the various time-dependent shortest path algorithms and *k*-shortest path algorithms available in the literature are described in Chapter 2. A description of the algorithm developed as a part of this study is presented in Chapter 3. Object-Oriented Methodology used to design a software layout where such algorithms can be efficiently implemented in practical incident management systems is described in Chapter 4. Empirical computational results and a statistical analysis of these results is presented in Chapter 5. A summary of results, conclusions and recommendations are discussed in the final chapter. Some statistical definitions for the results of Chapter 5 and a pseudocode form of the algorithm developed in this study is included in the appendix. References used in this study are listed in the end.

1.3. Motivation

1.3.1. Routing - Routing Algorithms For Static and Dynamic ITS Networks

Recent developments in ITS reflect a propensity for increased use of sophisticated algorithms for routing. Most of these algorithms have been applied successfully in the past for routing in data (computer) networks. Hence, it would be a good idea to study routing methods and issues in computer networks and understand the similarities and differences between computer networks and transportation networks. Such a study would also afford valuable insight into primary research concerns in developing new routing algorithms, especially in the context of the centralized architecture of ITS, where traffic flow would exhibit a behavior close to that of “packets” in computer networks.

1.3.2. A Comparison of Computer Networks and Future ITS Road Networks

Any generic network consists of nodes interconnected with links that act as conduits for data (traffic flow) transfer between nodes. A well-defined architecture has been developed for computer networks, that specifies the protocols under which routing tasks are performed, apart from performing other functions. The most popular architecture is the Open Systems Interface or the OSI layered system. The architecture consists of several layers, (for example, the network layer, physical layer, link layer, etc.) that roughly handle tasks at a level of complexity specified by the name given to the layer. Interestingly enough, the architecture proposed for the Advanced Vehicle Control Systems (AVCS) scenario is remarkably similar to the OSI system. It uses a hierarchical layered system and closely resembles the OSI system. Also recall that ATIS operations call for a system that consists of infrastructure, controllers and users. The emerging philosophy in ITS indicates that future transportation networks will be subjected to a greater degree of centralized control than what currently exists.

1.3.3. What is a Routing Algorithm ?

Bertsekas and Gallagher [1992] refer to the routing algorithm as *the network layer protocol that guides packets* (*information stored as small strings of bits*) *through the communication subset to their correct destination*. Some reasons for the complexity of routing algorithms are listed below.

1. Coordination between all nodes of the network.
2. Possibility of link and node failures.
3. Congestion.

Note that the possibility of link failures and congestion is very similar to that faced in transportation networks due to incidents or traffic jams. In the ITS scenario, since vehicles will be provided with information, coordination between local traffic management centers will be an important factor. Usually two types of algorithms are generally used for routing in networks.

1. Shortest path based routing algorithms.
2. Optimal routing algorithms based on other measures.

For each of these algorithms, one can add a “static” or a “dynamic” prefix to indicate if the algorithm takes into account the variation of the state variables in the network with time. The efficiency of a routing algorithm depends on how it performs during times of congestion in the network. The main tasks that have to be performed by these routing algorithms are listed below.

1. Route choice.
2. Error-free and reliable delivery of message.

1.3.4. Performance Measures for Routing Algorithms

There are two important system performance measures that will be crucial in deciding whether we accept or reject a routing algorithm.

1. Throughput in the network or quantity of service.
2. Average packet delay or quality of service.

These performance measures are analogous to what a maximal network flow algorithm or a min-cost network flow algorithm tries to achieve. One crucial difference between computer and traffic networks is the factor of user-behavior in transportation networks. It would be unrealistic to model cars exactly like packets, at least in the non-ITS scenario, but as time progresses and traffic congestion (unfortunately) increases, drivers may be forced to travel on pre-assigned paths for sake of the overall well-being of the system. Such an enforcement would lead to problems of equity and implementation. One solution for this problem would be to provide sufficient incentives that encourage drivers to take pre-assigned paths, using some method to price congestion. (An analogy for such an idea can be seen in airline ticketing, where passengers can sometimes make a

low-cost trip by choosing a multiple-leg flight.) This is also directly related to the problem of finding system-optimal entry times into the network with a waiting incentive to the user at the source node.

According to Bertsekas, routing interacts with flow control (similar to a traffic signal at source nodes and ramps) in determining these performance measures. Figure 2 illustrates the interaction of routing and traffic control. Herein lies the crux of the dynamic routing problem.

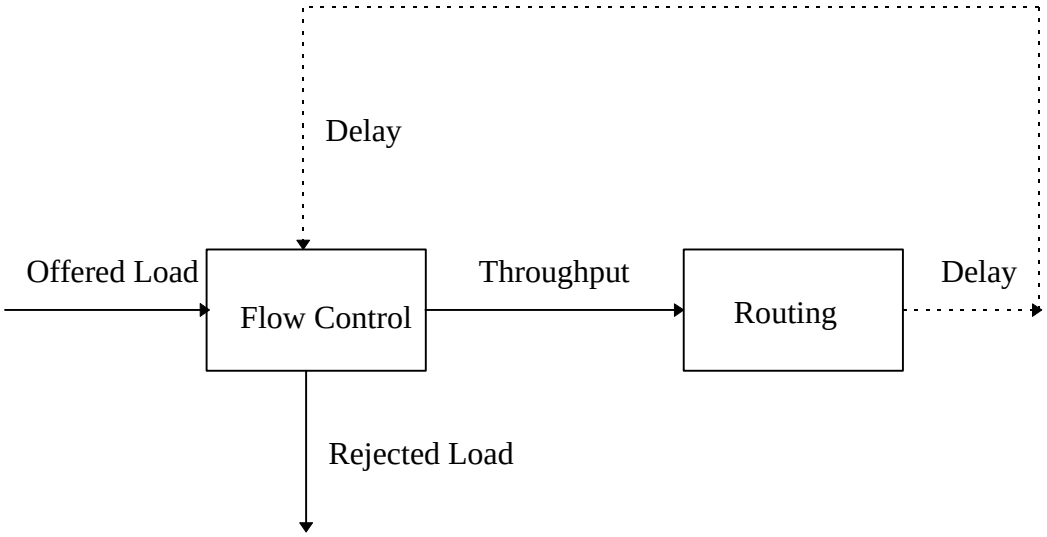

Figure 2. The Interaction of Routing and Flow Control.

The quantity and quality of service are usually opposite goals, due to adverse feedback effects. Increasing throughput results in increased delays. These effects are more conspicuous during periods of congestion. The same phenomenon can be observed in airports, where increasing arrival and departure rates result in greater unscheduled delays to flights. Interestingly, the aircraft delay costs here are very expensive and dynamic routing algorithms could be applied for routing this dynamic flow of aircraft through the links (taxiways) of the (airfield) network.

1.3.5. A Comparative Study of Popular Routing Algorithms

Among the two types of routing algorithms used, static shortest path methods do not incorporate feedback effects completely. Also, the emphasis is on minimizing delay. As a result, the network is prone to oscillations. Static optimal routing algorithms perform much better, but due to the non-inclusion of dynamic parameters, they would not be useful in real-time applications. Time-dependent shortest path (TDSP) algorithms do capture this dynamic effect, but no effort till now has been made to address the issue of quantity of service directly. Note that quantity of service is addressed indirectly via incorporation of feedback effects, but current TDSP algorithms would fail when multiple paths are required to meet traffic demands, during periods of extreme congestion. Consider the example shown in Figure 3.

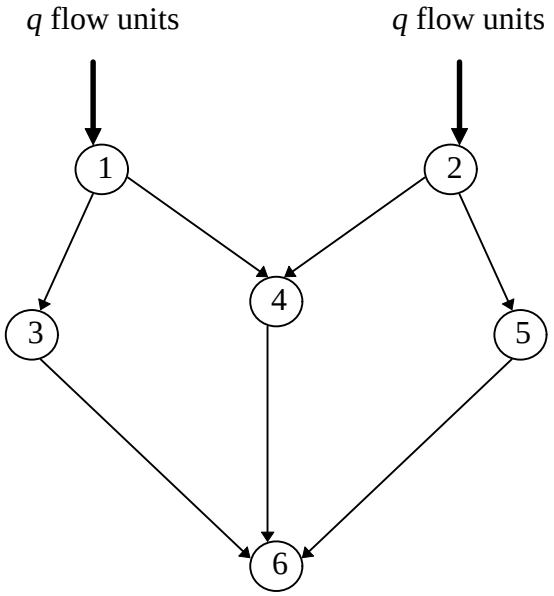

Figure 3. A case Where Multiple Shortest Paths are Necessary.

All links have a capacity of Q flow units. The shortest cost paths from the origin nodes 1 and 2 to destination node 6 are {1, 4, 6} and {2, 4, 6}. A flow of q units have to be routed from the origins to the destination node at the least cost. For values of q less than $Q/2$, this can achieved by routing the packets via the shortest path. For values of q greater $Q/2$,

the flow inputs have to be partitioned. As a result, multiple optimal paths are needed to route packets to their destination. For larger values of *q*, excess load will have to be rejected to avoid unreasonable delays.

1.3.6. System-Optimal Routing

Dynamic traffic assignment (DTA) that use system-optimal routing (rather than shortest path based routing) is an ongoing research field. Although such a method would probably be extremely accurate, it may be computationally expensive and slow, thus reducing its efficacy in any real-time application. The reason for this drawback is that whereas optimal routing algorithms primarily use gradient algorithms and nonlinear optimization, shortest path based algorithms use graph theoretic methods and are typically 10-100 times faster than optimal routing algorithms. A network optimization algorithm (for example, the RELAXT-IV code) is much faster compared to the simplex method when used to solve a pure network flow problem. DTA can be useful only if efficient implementation schemes can be found for practical applications. Note that all algorithms discussed in the study until now, have at least one drawback - they either are computationally slow or only address the issue of quantity or quality of service. This poses an intriguing question: *Is there any dynamic algorithm that can incorporate both quality and quantity of service and be fast enough for practical implementation* ?

This study partly addresses some issues related to this question. Note that a goal-programming or a bi-criterion objective function may be needed to solve this problem. The need for such performance measures and the consequences of any algorithm not measuring up to the required levels of service are can be illustrated by Figure 4. This figure illustrates the behavior of good and bad routing algorithms with increase in traffic.

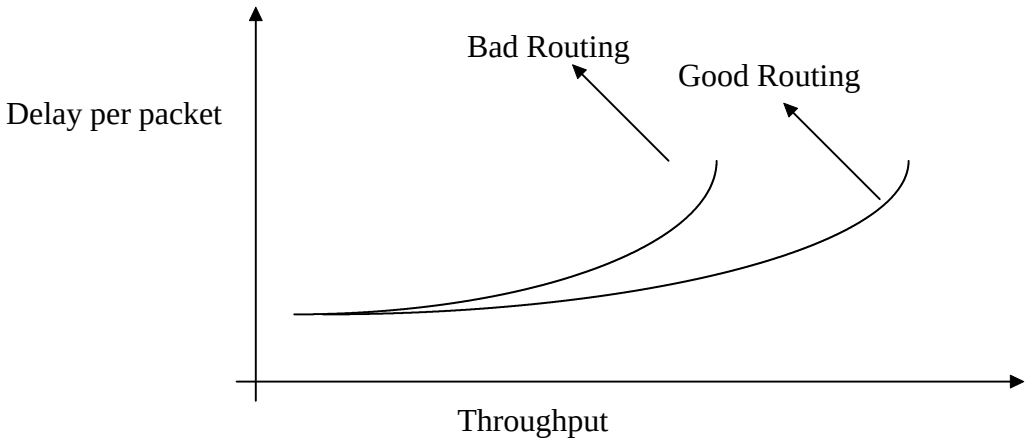

Figure 4. Good and Bad Routing Algorithms.

1.3.7. Good and Bad Routing Algorithms

In networks, a flow control mechanism rejects excess traffic and the throughput realized is the difference between the offered load (traffic demand) and the rejected load (cars that have to queue up on ramps before they are allowed to enter the network). A good routing algorithm would be more successful in keeping delay low as the flow control increases, or to be precise, good routing increases throughput for the same level of delay per packet during high traffic conditions and reduces the average delay per packet at moderate or low traffic conditions. We can formulate a simple mathematical programming model for this problem. Consider a time-space network $G_T(N_T, A_T)$[2]. We can associate the 2-tuple (t_{ij}, q_{ij}) with each link (i, j) in the network. Also, assume that we have the travel-delay versus flow curve for each link for each time-horizon under consideration, *i*.e., $t_{ij} = d_{ij}(q_{ij})$. In the simple case, we can construct a static, linear model by assuming linear functions for all state variables. Define routing decision variables x_{ij} as the number of flow units routed through the link (i, j). We can then formulate the following integer programming problem.

[2] A time-space network has nodes of the type (i, t), and arcs of the type $((i, t_1), (j, t_2))$, where t represents some point in time.

Maximize q

subject to:

$$\sum_{j \in FS(i)} x_{ij} - \sum_{j \in RS(i)} x_{ji} = \begin{cases} q & \text{if } i = s \\ -q & \text{if } i = f \\ 0 & \text{otherwise.} \end{cases}$$

$$\sum_{(i,j) \in A_T} \sum d_{ij}(x_{ij}) \leq T^*$$

$$0 \leq x_{ij} \leq c_{ij} \quad \forall (i, j) \in A_T$$

x integer.

where

c_{ij} is the capacity of link (i, j),

q is the flow to be routed from source s to destination f,

T^* is the maximal allowable delay, and

d_{ij} is the delay function for link (i, j).

The model shown above is a maximal-flow problem subject to the flow conservation constraints and the additional constraint that the total delay for the route is not to be more than some specified value T. We can construct similar models for many routing applications. Such network-flow models with side-constraints are typical in cases where we have opposing objectives to consider.

1.3.8. Distributed and Centralized Algorithms

Another important issue is the way these algorithms are implemented in practice. Based on implementation, they can be classified as either centralized or distributed. In centralized algorithms, all the route choices are made at a central node, while in distributed algorithms, the route choice computations are shared among the network nodes based on information exchange. Obviously, under the centralized ITS architecture, centralized algorithms would be in place, but as we shall see later, most computer

networks (ARPANET, for example) use distributed algorithms. A classical example for centralized and distributed algorithms for shortest paths is Dijkstra's algorithm and the Bellman-Ford algorithm, respectively.

The ARPANET initially used the Bellman-Ford algorithm. The nodes exchange information every 0.625 seconds. The link-delays are modeled based on the number of packets in the queue. This value fluctuates rapidly and thus renders the network vulnerable to oscillations. A large positive constant is therefore added to the link-delays to reduce this effect, but this results in the reduction of the sensitivity of the algorithm to congestion. Note that, today, variable message signs are used to provide congestion information to travelers, and hence drivers can make adaptive route choices. In the ITS scenario, one can visualize a traveler being provided with more detailed travel information that will be frequently updated, resulting in the transportation network behavior resembling that of a computer network. If travelers are provided with no prior information by traffic information/management centers and are guided using on-board displays, etc., then their behavior is closer to a distributed system.

The TYMNET on the other hand, uses a centralized algorithm. Although this algorithm has worked well, there is the fear of the central node failing, resulting in system-wide confusion. This is not the case in a distributed system. In the ATIS case, if the travelers are given pre-assigned routes by a central traffic management center, it becomes a centralized implementation.

IBM's SNA uses a policy where the route choice is partially left to the user. Here, a choice of several paths is offered. A common routing method in some networks is to use hierarchical routing, where routes are selected in some order of priority. Adaptive routing has been used in telephone networks. This seems to be a method that could be adopted for routing traffic in road networks.

1.3.9. Conclusions From the Study of Routing in Computer Networks

Some general conclusions that can be drawn from this study are summarized below.

1. Routing in transportation and data networks will be functionally similar in the future.
2. Providing multiple routes is beneficial in improving the quantity of service.
3. Oscillations must be avoided but sensitivity to congestion may be significant.
4. The failure of the traffic management center may be dangerous to a centralized system.
5. Adaptive routing with constantly updated information is helpful in avoiding congested routes.
6. Quality of service, quantity of service and speed are the three most important performance measures for any routing algorithm.

Based on all these factors, time-dependent shortest path algorithms seem to be the best bet for routing algorithms currently, unless DTA (based on optimal routing methods that consider other measures) can be implemented efficiently. In any case, DTA would require high-speed computing facilities that would be very expensive.

1.4. The Linear Programming Formulation of the Shortest Path Problem.

Let us first study the linear programming formulation for the static shortest path problem and see how the problem can be solved efficiently. The shortest path (SP) problem is a classical and important combinatorial problem that arises in many contexts. Here, a brief description of the Linear Programming (LP) formulation is given and the equivalence of the SP problem to the min-cost network flow problem is shown. We can derive the necessary and sufficient conditions for optimality for an LP problem from the Karush-Kuhn-Tucker (KKT) conditions for optimality. An optimal solution to an LP problem must satisfy the conditions of *primal feasibility, dual feasibility and complementary slackness*. (Interested readers can read [Bazaraa, Jarvis, and Sherali, 1990] for a more complete discussion.)

Consider a network $G(N, A)$ having $|N|$ nodes and $|A|$ arcs. Each arc has a delay cost of a_{ij}. These costs can be negative, but the network contains no negative cycles. Let FS(i) and RS(i) denote the forward star and reverse star functions for node i. Then the (primal) LP formulation for the SP problem can be written as shown below.

1.4.1. The Primal Formulation

(SPP) Minimize $\sum_{(i,j)\in A} a_{ij} x_{ij}$ (1)

subject to:

$$\sum_{j\in FS(i)} x_{ij} - \sum_{j\in RS(i)} x_{ji} = s_i \forall i \in N \quad (1a)$$

$$x_{ij} = 0 \text{ or } 1 \ \forall \ (i, j) \in A \quad (1b)$$

$s_s = 1, \quad s_t = -1, s_i = 0 \ \forall \ i \neq (s, t)$.

There are exactly two non-zero values for each column (arc) in the node-arc incidence matrix (the node-conservation constraints represented in matrix form). For arc (i, j), a +1 is found in row i and a -1 is found in row j. Due to the total unimodularity property of this network structure, the (extreme point) optimal solutions take only integral values. Hence the 0-1 requirement for x_{ij} can be equivalently replaced by the constraint $x_{ij} \geq 0$. Problem SPP represent the primal problem. Let us write the dual problem for (1). Denoting the dual variables corresponding to constraint i by w_i, and letting $p_i = -w_i$, the dual formulation (SPD) can be written as shown below.

1.4.2. The Dual Formulation

(SPD) Maximize $p_t - p_s$ (2a)

subject to:

$p_j - p_i \leq a_{ij} \ \forall \ (i, j) \in$ (2b)

p_i unrestricted.

An optimal solution is guaranteed in the absence of negative cost circuits in the network. At optimality, we have

$$p_j \leq p_i + a_{ij}, \forall (i, j) \in \quad (3)$$

$$p_j = p_i + a_{ij}, \forall (i, j) \text{ with } x_{ij}>0. \quad (4)$$

Note that p_i (the negative of the dual variable, w_i) represents the shortest path length to node *i*, and the resulting shortest path tree rooted at the node *s*, corresponds to the optimal basis for (1).

1.4.3. Labeling Algorithms for the SP Problem

Labeling algorithms are the most popular and efficient algorithms for solving the SP problem. These algorithms utilize a label for each node that corresponds to the tentative shortest path length p_i to that node. The algorithm proceeds in a way such that these labels are improved until the shortest path is found. There are two types of labeling algorithms - label setting (LS) and label correcting (LC). The LS algorithm sets the label of one node permanently at each iteration, thus increasing the shortest path vector by one component at each step. The LC algorithm does not set any label permanently. All the components of the shortest path vector are obtained simultaneously, after the algorithm terminates. A predecessor label is stored for each node, which represents the previous node in the shortest path to the current node. This is used to construct the shortest paths to each node, by backtracking. A complete discussion is presented in Chapter 2.

1.4.4. A Shortest Path Algorithm for a Network having Mixed Costs

While the SP problem can be solved as a min-cot network flow problem, specialized algorithms have been developed to solve the SP problem. These algorithms are much faster than a more general LP algorithm. The Partitioned Shortest Path (PSP) algorithm of Glover and Klingman [1986] is one such example. The PSP algorithm (an LC algorithm) is an efficient method that calculates the shortest path from the origin node to all other nodes in the network. Unlike Dijkstra's algorithm (an LS algorithm) [1959], this algorithm can be used for networks where link costs can be of mixed sign, but have no *negative cycles*. Each node *i* stores two parameters, p_i and pre(*i*). The parameter p_i

represents the tentative shortest path distance from the origin to node i and pre(i) represents the predecessor function of node. The procedure also requires a storage scheme for nodes whose labels are potential candidates for being updated. This is done using two lists NOW and NEXT. Initially, only the origin node is inserted in NOW, and NEXT remains empty. At each step of the procedure, we sequentially scan the forward star of the nodes in NOW. The node is then deleted from NOW. Node labels that get updated are put in NEXT if they are not already present there. When NOW is empty, NEXT is set to NOW and the process continues. When NOW and NEXT are both empty, the algorithm terminates with the node labels p_i representing the shortest path lengths from the origin to node i, and pre(i) yields the SP spanning tree rooted at the origin node s. The procedure is listed below.

1. $p_s = 0$. $p_i = \infty \ \forall i \neq s$.

   NOW = $\{s\}$, NEXT = $\{\varnothing\}$

   Let C_o = sum of the negative costs in the network.

2. If NOW = $\{\varnothing\}$ go to step 3.

   Else

   ii) select $i \in$ NOW

   iii) NOW = NOW - $\{i\}$

   iv) p_j = minimum $\{p_j, p_i + a_{ij}\}$, $\forall j \in$ FS(i)

   v) If $p_j < C_o$, stop. A negative cycle exists. Else continue with step (vi).

   vi) If p_j is updated,

      a) pre(j) = i.

      a) If $j \notin$ NEXT, NEXT = NEXT $\cup$ $\{j\}$

3. If NEXT = $\{\varnothing\}$, stop.

Else

   i) NOW = NEXT.

   ii) Go to step 2.

We can now see that static shortest path problem can be solved using labeling algorithms in an efficient manner. Next, we consider the case where the travel time on a link is dependent on the actual time of travel on that link. This introduces some complications in applying a labeling algorithm. Several approaches have been used to solve such a time-varying shortest path problem. Also, keeping in mind the requirement for alternate routes, multiple optimal paths may also be need to be found. These approaches are described in the next chapter.