DYNAMIC PROGRAMMING HEURISTIC FOR SYSTEM OPTIMAL ROUTING IN DYNAMIC TRAFFIC NETWORKS

Alfredo Garcia
Robert L. Smith
Department of Industrial & Operations Engineering
The University of Michigan
Ann Arbor, Michigan 48109-2117

and

Raja Sengupta
Institute for Transportation Studies
University of California
Berkeley, CA

Technical Report 95-22

November 1995

Dynamic Programming Heuristic for System Optimal Routing in Dynamic Traffic Networks.

Alfredo Garcia, Robert L. Smith.
Industrial and Operations Engineering Department.
University of Michigan, Ann Arbor, MI. 48109
and
Raja Sengupta
Institute for Transportation Studies
University of California.
Berkeley, CA.

June 16, 1995

Abstract

We propose a heuristic procedure to compute suboptimal routings (system optimal) in Dynamic Traffic Networks. The procedure is recursive which greatly simplifies and enhances. implementation and performance respectively. Finally, under mild assumptions on the modeling of congestion, the procedure is shown to compute system optimal routings. A first implementation is provided with computational results.

1. Introduction

We propose a new procedure to solve for suboptimal routings in Dynamic Traffic Networks. By optimal. we mean "System Optimal", i.e routings that minimize the total travel time experienced by all platoons. Since link travel times depend upon congestion level, they are time dynamic, which greatly increases the complexity of the problem. Nonetheless, there is a growing body of work on this problem in the literature (see [1],[2] and [3]). Various modeling and solution techniques have been examined. The critical issue however is tractability, since the goal is to provide with a fast and reliable way of computing the desired routings.
In view of this, and with eventual loss of optimality, we concentrate in a fast and if possible suboptimal procedure. The motivation comes from Kaufman and Smith [4], in which the author introduce the "time consistency" assumption for congestion models on links. This assumption, which turns out to be easily satisfied by many models, ensures for a forward solution of the single platoon routing problem. We extend their ideas to the multi-platoon case and show that the procedure here proposed also solves for system optimal routings, when all links in the network satisfy the "time consistency" assumption.

2. Dynamic Programming "Efficient" Heuristic

In this section, we provide theoretical basis for the proposed new procedure. To facilitate the analysis, a *random traveler* approach is taken. This allows to study the *system* optimality issues,

in terms of a single random platoon to be chosen. The rational of the procedure is as follows.: "Efficient" means that one wants to move platoons through the network so they reach intermediate destinations as soon as possible. However, to maintain tractability the procedure only keeps track of efficient routings through intermediate destinations, which raises the possibility of suboptimality since for some intermediate destinations or macronode it is not necessarily true that the efficient routing provides the best (system-optimal) route to attain the given macronode.
One can also view the procedure as an "aggregative" one(see, for instance, Bean,Birge and Smith[6]) in which we solve the original problem, which in turn can be seen as a shortest path problem, by aggregating to a single macronode, all different possibilities to reach in time a given set of intermediate destinations, we fix the cost to reach that macronode by efficient routing.

2.1 Decision Network Definition

We denote $m \in Z^+$, the total number of platoons to be considered and (N, A) the traffic network, consisting of a set of nodes N, and links $(i, j) \in A$ with $i, j \in N$. We also denote $(O, D)_p$ be the origin destination pair associated with platoon p , $(1 \leq p \leq m)$ with $O, D \in N$.

Based on the actual network we define a decision network (macro-network) which represents the space of all admissible routing decisions for the m platoons to be routed. A decison network is defined to be a pair $(\mathcal{X}, \mathcal{A})$ where $\mathcal{X} \subseteq N^m$ is the macro-node set and $\mathcal{A} \subseteq \mathcal{X} \times \mathcal{X}$ is the macro-arc set. Let $g_i : \mathcal{X} \rightarrow N$ be the projection for the i-th node in an m-tuple of nodes. Then the set $\mathcal{A}$ is required to satisfy the following condition

$$(X, Y) \in \mathcal{A} \Longrightarrow (g_i(X), g_i(Y)) \in A, \quad 1 \leq i \leq m$$

i.e., a macro-arc must be constituted of arcs in A. A route π in this decison network is any finite sequence of macro-nodes $\pi = X_0 X_1 \ldots$ with the property that $(X_j, X_{j+1}) \in \mathcal{A}$ for all j. Thus π is not a route for any one platoon but rather a routing prescription for the entire network. In other words given π the corresponding route for the p-th platoon is $g_p(X_0)g_p(X_1)\ldots$. Observe that our assumptions ensure $(g_p(X_j), g_p(X_{j+1})) \in A$ for all j. The symbol $|\pi|$ will denote the length of the route π and it will be the number of macro-nodes in the sequence π. We also use the symbol $|.|$ to denote the cardinality of a set. We also indulge in a slight abuse of notation and write $Y \in \pi$ if a macro-node Y lies on a route π.

2.2 Travel time function definition

For each π and $X \in \pi$ we define the function $t_p^\pi(X)$ to be the arrival time of the p-th platoon at X under the routing prescription π. For a route $\pi = X_0 X_1 \ldots$ the following is true

1. $t_p^\pi(X_0)$ is the trip starting time of platoon p.
2. $t_p^\pi(X_{i+1}) = t_p^\pi(X_i) + \tau_{(x_p^i, x_p^{i+1})}(t_1^\pi(X_i), \ldots t_m^\pi(X_i), \rho_1, \ldots \rho_m)$,
   where ρ_p is the size of platoon $p, x_p^i = g_p(X_i)$ and $\tau_{(x_p^i, x_p^{i+1})}(.)$ is the travel time function associated with the directed link (x_p^i, x_p^{i+1}).
3. $0 < \tau_{(x,y)}(.) < \infty$ for all $x, y \in A$.

2.3 The random traveller

Let Ω denote the set of platoons. Then we assume that each platoon is identified with a unique number in the set $\{1, \ldots, m\}$ and define a random variable $Z : \Omega \rightarrow \mathbb{R}$ with a probability distribution $P_Z : \mathbb{R} \rightarrow [0, 1]$ such that

1. $P_Z(Z = p) = \gamma_p > 0, \quad 1 \leq p \leq m,$
   i.e., each platoon can be picked as the random platoon with positive probability.

2. $\sum_{p=1}^{m} \gamma_p = 1.$

Based on the random variable Z we define for any route π and macro-node $X \in \pi$ the random variable $T_X^\pi : \mathbb{R} \rightarrow \mathbb{R}$ with the probability distribution

$$P(T_X^\pi(p) = t_p^\pi(X)) = \gamma_p.$$

2.4 The consistency assumption

Let $(X, Y) \in \mathcal{A}$ and π, π' be two routes to X from some origin node X_o. Then we assume the following property.

$$E[T_X^\pi] \leq E[T_X^{\pi'}] \Longrightarrow E[T_Y^\pi] \leq E[T_Y^{\pi'}].$$

An immediate consequence of this assumption is that if an optimal route exists it must be an acyclic route. The precise definition of a cyclic route is as below.

π is a cyclic route to a macro-node X if for $\pi = X_0 \ldots X_{|\pi|-1}$ there exists i, j with $0 \leq i < j \leq |\pi| - 1$ such that $X_i = X_j$.

In the subsequent develpment is will be assumed that X_o denotes the macro-node from which all trips originate. For any node $X \in \mathcal{X}$ we assume Π_X denotes the set of routes from X_o to X. We assume that $X_o, (\mathcal{X}, \mathcal{A})$ are such that all nodes in $\mathcal{X}$ are reachable from X_o, i.e. Π_X is non-empty for all X. We also define the projection function $\lambda_k(\pi)$ to be the prefix of length k of the route π.

An optimal route in a set Π_X is any route $\pi^* \in \Pi_X$ such that

$$E[T_X^{\pi^*}] = \min_{\pi \in \Pi_X} E[T_X^\pi].$$

Proposition 2..1 If an optimal route exists then there also exists an optimal route that is acyclic.

Proof: Let there exist $\pi^* = X_0^* \ldots X_{|\pi^*|-1}^* \in \Pi_X$ such that π^* is cyclic. Then pick i, j with $i < j$ such that $X_i^* = X_j^*$. Then from $\tau_{(x,y)}(.) > 0$ we get $t_p^{\pi^*}(X_i^*) < t_p^{\pi^*}(X_j^*)$ for all platoons p which implies that $E[T_{X_i^*}^{\pi^*}] < E[T_{X_j^*}^{\pi^*}]$. Consider the route $\pi' = \lambda_i(\pi^*) X_{j+1}^* \ldots X_{|\pi^*|-1}^*$. Then by the consistency assumption $E[T_{X_{j+1}^*}^{\pi'}] \leq E[T_{X_{j+1}^*}^{\pi^*}]$. Repeated application of the consistency assumption gives

$$E[T_{|\pi^*|-1}^{\pi'}] \leq E[t_{|\pi^*|-1}^{\pi^*}],$$

which implies that π' is also an optimal route.

We now use this argument for the following inductive construction. Assume that all optimal routes are cyclic. Pick some optimal $\pi^* \in \Pi_X$. We know $|\pi^*| < \infty$. Set $\pi_0 = \pi^*$ and generate a sequence $< \pi_n >$ in the following manner. Given an optimal $\pi_n = X_0^n \ldots X_{|\pi_n|-1}^n$, since it is cyclic, there exists $i, j, i < j$ with $X_i^n = X_j^n$. Thus as before construct $\pi_{n+1} = \lambda_i(\pi_n) X_{j+1}^n \ldots X_{|\pi_n|-1}^n$. By the prior argument π_{n+1} is also optimal and hence by hypothesis it is cyclic. Moreover $|\pi_{n+1}| < |\pi_n|$

and the construction can continue. Since $|\pi_0| < \infty$ this implies that for $n > |\pi_0|$, $|\pi_n| = 0$. This is absurd. Thus there exists some optimal route that is acyclic. ■

In the subsequent development we use Π_X to denote the set of all acyclic routes from X_o to X. Since each route is of finite length and $\mathcal{X}$ is finite, Π_X (the set of acyclic routes) is also finite. This together with the fact that $\tau_{(x,y)} < \infty$ implies that for any $X \in \mathcal{X}$ an optimal route exists. Note that by existence we understand that the minimum exists and is finite.

2.5 Efficient Routes

Definition 2..2 For all $X \in \mathcal{X}$ such that X is reachable from X_o define $\Pi_X^e \subseteq \Pi_x$ to be the set of all $\pi = X_0 \ldots X_{|\pi|-1} \in \Pi_X$ such that

1. $\lambda_{|\pi|-2}(\pi) \in \Pi^e_{X_{|\pi|-2}}$
2. $E[T_X^\pi] = \min_{(X',X)} \min_{\pi' \in \Pi^e_{X'}} E[T_{X'}^{\pi'}] + \sum_{p=1}^{p=m} \gamma_p \rho_p \tau_{(x'_p,x_p)}(t_1^{\pi'}, \ldots t_m^{\pi'}, \rho_1, \ldots \rho_m)$

Then Π_X^e is the set of all effcient routes to the node X.

Proposition 2..3 Efficient routes have the following properties.

1. Π_X^e is non-empty for all X.
2. If $\pi = X_0 \ldots X_{|\pi|-1} \in \Pi_X^e$ then $\lambda_k(\pi) \in \Pi^e_{X_k}$ for all k such that $0 \leq k \leq |\pi| - 1$.
3. If $\pi, \pi' \in \Pi_X^e$ then $E[T_X^\pi] = E[T_X^{\pi'}$.

Proof: The proofs of parts (2) and (3) are immediate from the definition of effcient routing. Therefore we prove only the first part i.e., that the definition of efficient routing is not vacuous.

For any $X \in \mathcal{X}$ the minimum in the definition of efficient routing is over the set

$$S(X) = \{\pi X : \pi \in \Pi^e_{X'}, (X', X) \in \mathcal{A}\}.$$

Thus if $\Pi_X^e = \emptyset$ then either $S(X)$ is empty or $S(X)$ is non-empty and no minimum exists over the set $S(X)$. Consider the case $S(X) \neq \emptyset$. We show later that this must indeed be true. Since $\Pi^e_{X'} \subseteq \Pi_{X'}$ and $|\Pi_{X'}| < \infty$, we get $|\Pi^e_{X'}| < \infty$. Moreover $\mathcal{X} < \infty$ implies that $|\{X' : (X', X) \in \mathcal{A}\}| < \infty$. From these facts $|S(X)| < \infty$. Moreover $S(X)$ non-empty implies that there exists X' such that $\Pi^e_{X'}$ is non-empty which in turn implies that there exists some $\pi \in \Pi^e_{X'}$ such that $E[T^\pi_{X'}] < \infty$. Since $\tau_{(x,y)}(.) < \infty$ for all $(x, y) \in A$, we get $E[T_X^{\pi X}] < \infty$. From this fact and $|S(X)| < \infty$ we get that the minimum over $S(X)$ exists and is finite.

We now show that $S(X)$ is indeed non-empty. Pick any $\pi = X_0 \ldots X_{|\pi|-1} \in \Pi_X$. Obviously $\Pi^e_{X_0} = \{X_o\} \neq \emptyset$. Let $\Pi^e_{X_i} \neq \emptyset$. Then $S(X_{i+1}) \neq \emptyset$ because $\{\pi X_{i+1} : \pi \in \Pi^e_{X_i}\} \subseteq S(X_{i+1})$. Then by the prior argument $\Pi^e_{X_{i+1}} \neq \emptyset$. By induction $\Pi^e_X \neq \emptyset$. ■

Theorem 2..4 There exists an optimal route to X_D that is also efficient.

Proof:

Let $\pi^* = X_0^* \ldots X_{|\pi^*|-1}^* \in \Pi_{X_D}$ be an optimal route. We show by induction that there exists an optimal route (not necessarily π^*) that is also efficient.

Induction hypothesis: For all $k, 0 \leq k \leq |\pi^*|-1$ there exists an optimal route $\tilde{\pi} = \lambda_k(\tilde{\pi}) X_{k+1}^* \ldots X_{|\pi^*|-1}^*$ such that $\lambda_k(\tilde{\pi}) \in \Pi^e_{X_k^*}$.

The base case is $k = 0$. In this case we choose $\tilde{\pi} = \pi^*$ itself, since $\pi^* \in \Pi^e_{X_0} = \{X_0\}$.

Assume w.l.o.g. that the induction hypothesis is true at some k but not at $k+1$. Pick $\tilde{\pi}$ as in the hypothesis. Then $\lambda_{k+1}(\tilde{\pi}) \notin \Pi^e_{X_{k+1}^*}$. By the definition of efficient routing there exists some $\pi' \in \Pi^e_{X_{k+1}^*}, \pi' \neq \lambda_{k+1}(\tilde{\pi})$ such that

$$E[T^{\pi'}_{X_{k+1}^*}] < E[T^{\tilde{\pi}}_{X_{k+1}^*}].$$

Consider now the route $\hat{\pi} = \pi' X_{k+2}^* \ldots X_{|\pi^*|-1}^*$. By the consistency assumption

$$E[T^{\hat{\pi}}_{X_{k+2}^*}] \leq E[T^{\tilde{\pi}}_{X_{k+2}^*}]$$

and by applying the consistency assumption repeatedly to the sequence $< X_{k+j}^* >_{j=1}^{j=|\pi^*|-1}$ we get

$$E[T^{\hat{\pi}}_{X_{k+j}^*}] \leq E[T^{\tilde{\pi}}_{X_{k+j}^*}]$$

for all j. In particular for $j = |\pi^*| - 1$ we get

$$E[T^{\hat{\pi}}_{X_{|\pi^*|-1}^*}] \leq E[T^{\tilde{\pi}}_{X_{|\pi^*|-1}^*}]$$

which implies that $\hat{\pi}$ is an optimal route. Since $\lambda_{k+1}(\hat{\pi}) = \pi' \in \Pi^e_{X_{k+1}^*}$ is also true, $\hat{\pi}$ satisfies the induction hypothesis for $k+1$. By induction the hypothesis is true for all k.

In particular if we choose $k = |\pi^*| - 1$ then we have an optimal route $\tilde{\pi}$ such that $\lambda_{|\pi^*|-1}(\tilde{\pi}) \in \Pi^e_{X_{|\pi^*|-1}^*} = \Pi^e_{X_D}$. Thus $\tilde{\pi}$ is both an optimal and an efficient route to X_D. This proves that within the class of efficient routes there exists an optimal route.

■

Theorem 2..5 Under *Strong* consistency assumption, all optimal routes are efficient.

Proof: We show the contrapositive. Assume $\pi = X_0 X_1 \ldots X_{D-2} X_{D-1} X_D \notin \Pi^e_{X_D}$, then there are two possibilities :

(1) $X_0 \ldots X_{D-1} \in \Pi^e_{X_{D-1}}$, and then by definition, for any $\pi' \in \Pi^e_{X_D}$, we have:

$$E[T^{\pi}_{X_D}] > E[T^{\pi'}_{X_D}]$$

hence, it is certainly not optimal.

(2) $X_0 \ldots X_{D-1} \notin \Pi^e_{X_{D-1}}$, then we go backwards, and again we have two cases :

(2.1) $X_0 \ldots X_{D-2} \in \Pi^e_{X_{D-2}}$, then let $\lambda_{D-1}(\pi') \in \Pi^e_{X_{D-1}}$, for some π', it follows that: $E[T^{\pi}_{X_{D-1}}] > E[T^{\pi'}_{X_{D-1}}]$ and we set

$$\pi' = \lambda_{D-1}(\pi') X_D$$

By strong consistency, it follows that :

$$E[T^{\pi}_{X_D}] > E[T^{\pi'}_{X_D}]$$

(2.2) $X_0 \ldots X_{D-2} \notin \Pi^e_{X_{D-2}}$, we go backwards and again have two posibilities :

(2.2.1) $X_0 \ldots X_{D-3} \in \Pi^e_{X_{D-3}}$, then let $\lambda_{D-2}(\pi') \in \Pi^e_{X_{D-2}}$, for some π', it follows that: $E[T^{\pi}_{X_{D-2}}] > E[T^{\pi'}_{X_{D-2}}]$ and we set

$$\pi' = \lambda_{D-2}(\pi')X_{D-1}X_D$$

By strong consistency, it follows that :

$$E[T^{\pi}_{X_{D-1}}] > E[T^{\pi'}_{X_{D-1}}]$$

One more iteration yields :

$$E[T^{\pi}_{X_D}] > E[T^{\pi'}_{X_D}]$$

(2.2.2) $X_0 \ldots X_{D-3} \notin \Pi^e_{X_{D-3}}$,We go backwards ...

This inner loop will eventually reach the limiting situation :

(2.2...1) $X_0X_1 \in \Pi^e_{X_1}$, then let $\lambda_2(\pi') \in \Pi^e_{X_2}$, for some π', it follows that: $E[T^{\pi}_{X_2}] > E[T^{\pi'}_{X_2}]$ and we set

$$\pi' = \lambda_2(\pi')X_3 \ldots X_{D-1}X_D$$

By strong consistency, it follows that :

$$E[T^{\pi}_{X_3}] > E[T^{\pi'}_{X_3}]$$

Now iterating as above, we have :

$$E[T^{\pi}_{X_D}] > E[T^{\pi'}_{X_D}]$$

(2.2...2) $X_0X_1 \notin \Pi^e_{X_1}$, then let $\lambda_1(\pi') \in \Pi^e_{X_1}$, for some π', it follows that: $E[T^{\pi}_{X_1}] > E[T^{\pi'}_{X_1}]$ and we set

$$\pi' = \lambda_1(\pi')X_2 \ldots X_{D-1}X_D$$

By strong consistency, it follows that :

$$E[T^{\pi}_{X_2}] > E[T^{\pi'}_{X_2}]$$

Now iterating as above, we have :

$$E[T^{\pi}_{X_D}] > E[T^{\pi'}_{X_D}]$$

We finally conclude that $\pi \notin \Pi^*_{X_D}$.

■

3. Implementation of the Dynamic Programming "Efficient" Heuristic.

In this section, we provide an algorithm to solve for an efficient routing. We leave behind the random traveler scheme, and in order to fully specify the mechanics of the algorithm, we use more notation :

3.1 More Notation

- We define $L_{i,j}(.) : Z^+ \rightarrow Z^+$to be the dynamic load function for link $(i,j) \in A$, i.e evaluated at some time epoch it yields the total number of vehicles currently on the link.
- We let $I_{i,j}(.) : Z^+ \rightarrow Z^+$ be the link impedance funtion, i.e for a given number of vehicles in the network it provides the travel time to be experienced if the link is to be entered. The relation between this entities is the following :

$$\tau_{i,j}(.) = I_{i,j} \circ L_{i,j}(.) \; (i,j) \in A$$

- Let the m-tuple $X = (x_1, x_2, .., x_m)$ be a*macronode*, as defined in the previous section, where $x_p \in N$, $1 \leq p \leq m$, i.e a feasible distribution of platoons across the network, for instance platoon 1 located at node x_1, platoon 2 located at node x_2 etc... Finally let $f(x_1, x_2, .., x_m)$ be the total trip time experienced for platoons leaving origins $X_o = (O_1, O_2, ..., O_m)$ to intermediate destinations $X = (x_1, x_2, .., x_m)$ through an **Efficient** routing, as defined above.

We restate the definition of Efficient routings as follows :

$$f(X) = \min_{\substack{\pi' \in \Pi^{\epsilon}_{X'} \\ \text{s.t } (X', X) \in \mathcal{A}}} \{f(X') + \sum_{p=1}^{m} \rho_p \tau_{(x'_p, x_p)}(t_1^{\pi'}, \ldots t_m^{\pi'}, \rho_1, \ldots \rho_m)\}$$

Then, Π^{ϵ}_X is the argument of the above minimization problem. Now, suppose that the macronode $X^* = (x_1^*, x_2^*, ..., x_m^*)$ is the attained by the argmin of the definition above, then the time updating goes as follows:

$$t_p(X) = t_p(X^*) + \tau_{(x_p^*, x_p)}(t_1(X^*), \ldots t_m(X^*), \rho_1, \ldots \rho_m),$$

To complete the formulation we clearly have $f(O_1, O_2, ..., O_m) = 0$ and our problem is to find $f(D_1, D_2, ... D_m)$.
One last set of equations to fully describe the recursion is the way dynamic loadings on links are updated. Then, the loadings on the links reached by an efficient routing are updated as follows :

$$L_{(x_p^*, x_p)}(t) = L_{(x_p^*, x_p)}(t) + \rho_p \;\; ; \; t_p(X^*) \leq t < t_p(X)$$

This last equation simply states the fact that to whatever the number of vehicles in link $(x_p^*, x_p) \in A, 1 \leq p \leq m$, we add the size of the platoon entering the link, namely ρ_p, and since the platoon will take $\rho_l \cdot \tau_{(x_p^*, x_p)}(t_1(X^*) \ldots t_m(X^*), \rho_1, \ldots \rho_m)$ time units to traverse the link, this addition must be carried out for time periods $t_p(X^*) \leq t_p(X)$.

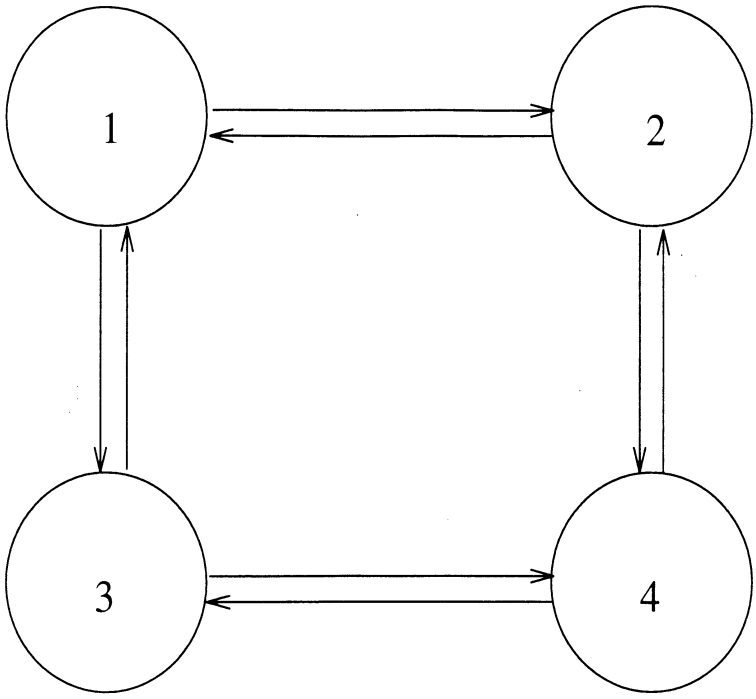

Figure 1: Network 1

3.2 Example.

We try to illustrate the notation with the following example. Consider the network on figure 1. Suppose that we have four platoons to consider, each of these leaving a different node to the node located in the "diagonal" respectively. Since the order in which we write the macronode m-tuple is arbitrary we set for instance, the order defined by node labels. Then our macronode origin is the four-tuple $(O_1, O_2, O_3, O_4) = (1, 2, 3, 4)$ and the macronode destination is $(D_1, D_2, D_3, D_4) = (4, 3, 2, 1)$. So it follows that our problem is to find $f(4, 3, 2, 1)$ with the boundary condition $f(1, 2, 3, 4) = 0$.

The first iteration will examine one step macronodes reachables from (O_1, O_2, O_3, O_4) such as : $(2, 4, 1, 2)$, that is, the platoon leaving node 1 is routed to node 2 , the platoon leaving node 2 is routed to node 4 , the platoon leaving node 3 is routed to node 1, and the platoon leaving node 4 is routed to node 2. There are then a total of 2^4 of such macronodes. The first iteration of the procedure is trivial.

To illustrate the updating procedure, let us assume all platoons consist of one vehicle , all link travel times are constant and equal to 10 time units, and that all platoons leave their origins at time zero except for the platoon leaving node 4 at two time units. Then :

$$
\begin{aligned}
& t_4(1, 2, 3, 4) = 2 \\
& t_l(1, 2, 3, 4) = 0 \quad l = 1, 2, 3 \\
& t_4(2, 4, 1, 2) = t_4(1, 2, 3, 4) + 1 \cdot \tau_{(4,2)}(0) = 12 \\
& t_1(2, 4, 1, 2) = t_1(1, 2, 3, 4) + 1 \cdot \tau_{(1,2)}(0) = 10
\end{aligned}
$$

Now, if we further assume that to reach (D_1, D_2, D_3, D_4) the efficient intermediate destination m-tuple is $(2, 4, 1, 2)$, the updating is as follows :

$$L_{(1,2)}(t) = 1 \quad 0 \leq t < 10$$

$$L_{(4,2)}(t) = 1 \quad 2 \leq t < 12$$

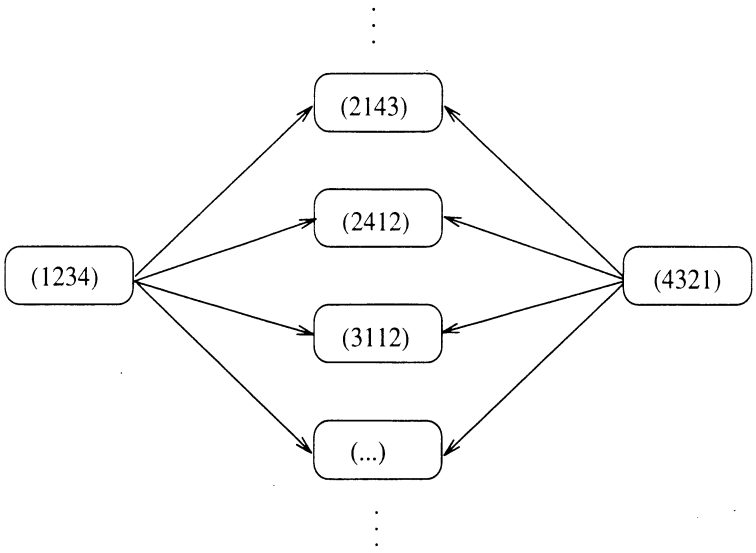

Figure 2: Decision Network for example.

3.3 Implementation

The example given in section 1, shows that to solve for efficient routing requires a lot of updating and possibly a lot of memory depending upon the size of the problem considered. For instance, the number of macronodes reachable from the origin in that example is exponential on the number of platoons to route. It is required then to avoid unnecessary updating, loadings and times along non-efficient paths, and it is vital to reduce the requirements of memory. For the latter concern, we will then consider one platoon at a time, i.e a different decision tree. In our example there will be then only two reachable macronodes from (O_1, O_2, O_3, O_4). These are $(2, 2, 3, 4)$ and $(3, 2, 3, 4)$, since we first consider the platoon leaving node 1 . From each of these we then have macronodes $(2, 1, 3, 4)$ and $(2, 4, 3, 4)$, and $(3, 1, 3, 4)$ and $(3, 4, 3, 4)$, respectively. These are obtained by considering routing choices for the platoon leaving node 2. For the former concern we will use a heuristic function that will help in the forward solving to prune paths that are not efficient.

3.3.1 The Algorithm

We briefly explain the use of a heuristic function in solving for the efficient routing. We define :

- $d : N \times N \rightarrow R^+$ the euclidean geographical distance between nodes in the network, e.g $d(x_1, D_1)$ is the distance between nodes x_1 and D_1.
- We now define a heuristic function, that yields a lower bound on a system-optimal routing total trip time from any macronode to the destination macronode as follows :

$$h(x_1, x_2, ..., x_m) = \sum_{p=1}^{m} \rho_p \cdot \frac{d(x_p, D_p)}{MAXSPEED}$$

where $MAXSPEED$ is the maximum speed allowed in the network.
We now define the order in which platoons are to be considered, that is, how to expand the decision tree. Since the intention is to use the procedure on real-time and the dynamic routing problem has no fixed horizon, rolling horizon procedures are to be considered. In view of this, we are mainly interested in system-optimal routing decisions for the first (in time) platoons in the network.

Hence, the sequence according to which we expand the network is deduced from the times platoons enter the network.
The algorithm then is the following :

1. Perform best first search to get a first feasible routing with related total trip time. Initialize a pointer at the macronode origin.

2. From pointer say $(x_1, x_2, ... x_m)$ on :

   - Fathom all reachable macronodes $(x'_1, x'_2, \ldots x'_m)$ from pointer that satisfy :

     $$\sum_{p=1}^{m} \rho_p \cdot \tau_{(x'_p, x_p)}(t_p(X')) + h(X') \geq \text{Total Trip Time}$$

     Expand all dangling macronodes (if accrued total trip time surpasses the current total trip time, then fathom).

   - Since by the previous step we have identified a part of the efficient routing, update all the entities.

1. If pointer is at $(D_1, D_2, ... D_m)$ stop. Else, update pointer and go to step 2 .

On the appendix, we give further detailed information on the computer code developed.

4. Computational Experiment

In order to test the validity of the procedure we compared its performance, in a problem with known optimal solution. For that, we use the same network as in Kaufman et al. [4](the same as in figure 1), where an optimal solution to the system-optimal dynamic routing problem was computed. Their model is richer in that they allow for platoon splitting but considerably limited to "small" examples. To test the size of problems that could be solved with the procedure, we enlarged that network, into networks 2 and 3. All the detailed information is given in the appendix.

Comparison w.r.t Optimal Solution			
	Routing time(min)	c.p.u(sec)	Ratio
Network 1	4816.2	22	1.0003
Comparison w.r.t Random Routings			
	Routing time(min)	c.p.u(sec)	Ratio
Network 1	6930	22	0.71
Network 2	9631.5	26	0.73
Network 3	19498.5	74	0.79

For network 1, Kaufman et al. were constrained to solve the problem to a very short time span. In contrast, our procedure is not constraint by time spans and it could solve until full clearance of the network.Best yet, when compared up to a common time span, the "efficient" routing came 0.003% short of optimality.It required 25 c.p.u seconds to solve it, whereas Kaufman et al. required more than 6 c.p.u minutes. For networks 2 and 3, the comparison is made to the average total trip time of a sample of 200 routings picked at random. In the table, the percentage value represents how good it was when compared to this average. It is interesting to note that under increasing congestion the procedure seems to perform better. One possible explanation is that under heavy congestion this assumption is more likely to hold.

References

[1] Carey,M. Optimal time Varying flows on Congested Networks . *Operations Research* **35** (1987) 1 58-69

[2] Friesz T.L, J. Luque, R.L.Tobin, and B. Wie. Dynamic Network Traffic Assignment considered as a Continuous Time Optimal Control Problem. *Operations Research* **37** (1989) 6 893-901

[3] Janson B.N. , Dynamic Traffic Assignment for Urban Road Networks . *Transportation Research B* **25B** 2/3 143-161

[4] Kaufman D.E, and R.L. Smith . Fastest Paths in Time Dependent Networks for IVHS Application .*IVHS Journal* **1** 1 1-11

[5] Kaufman D.E , R.L Smith and J. Nonis. A Mixed Integer Linear Programming Model for Dynamic Traffic Assignment.*ITS report*, University of Michigan.

[6] Bean,J.C, Birge, J and Smith R.L. Aggregation in Dynamic Programming. *Operations Research* **35** (1987) 1 58-69

5. Appendix 1

In this section we provide detailed information on the computer implementation of the procedure. Because of time constraints this implementation is not efficient at all, and needs further improvement. To avoid confusion, in the following we will assumme that the efficient routing is optimal. so that no distinctions apply.

5.1 General Setting

The following definitions are used throughout the code.

- **MEMORY**(Total memory requirement),**MAXs**(Number of entries in link impedance functions).
  MAXPLAT(maximum number of plattons departing from any node).
  MAXSPEED (maximum speed allowed in the network).
  HORIZON(number of time periods to be considered).
  PERIOD(time equivalence of one period unit).
- **struct_node**{} contains all information referenced by nodes.
- **struct_link**{} all information for links.
- **struct liste_link**{} information on network topology.
- **struct_macronode**{} information for macronodes as defined above.

The details of each structure contents is carefully explained in the code. As an illustration :

```
struct _node{
        int code;                       /* Code number for node */
        int Nbpred;                     /* total number of sucessors */
        int platoon;                    /* total number of plattons departing */
        int destination[MAXPLAT];       /* Array with codes for destinations */
        int size[MAXPLAT];              /* Sizes of plattons leaving node */
        double departure[MAXPLAT];      /*departure times for plattoons leaving */
        double *distance;               /* pointer to array of geographical dist. */
        struct liste_link *pred_link;   /* pointer to list of outward links */
        };
```

Pointers to array of each of these structures are then defined; e.g. ***tabnode** for the array of **struct_node**{}.
***tablink** and ***tab_mnode**. While the size of the node and link arrays are fixed(we will denote **NbN**, total number of nodes and **NbL**, total number of links) the size of the macronode array is chosen to be the constant **MEMORY** as defined above. This is due to the fact that we do not know in advance how much pruning will be necessary to find the efficient routing. However, we consider that this step can be improved by using the *c*-command **realloc()** in order to ask for just about the memory requirements, as we prune the macro-network.

5.2 Input Data

All the data that defines the problem setting must be provide through two input files, namely;**node_data** and **link_data**. We give instances of these files, in the exemple given in the paper. Nodes and links are listed in increasing code number. The codes are arbitrarily chosen.
For **node_data**:

```
4           /* Total number of nodes */
1 2 1       /* node code-number of sucessors-number of platoons departing*/
3           /* destination(s) code(s) for platoons departing */
1           /* Total number of vehicles in platoon(s)*/
0           /* platoon(s) departure(s) time(s) */
2 2         /* code sucessor node—code link joining them */
3 7         /* code sucessor node—code link joining them */
0 15 5 30   /* Geographical distance between this node and others*/
2 2 1       /* node code-number of sucessors-number of platoons departing */
4           /* destinations for platoons departing */
...         /* ... iterate */
```

For **link_data** :

```
8                       /* Total number of links */
1 0 2                   /* link code-current link loading-freeflow travel time */
0 0 101.3 160.7 222.2   /* link impedance function (defined for MAXs) */
2 0 1.768               /* link code-current link loading-freeflow travel time */
0 24.2 55.1 85.2 116.8  /* link impedance function (defined for MAXs) */
...                     /* ... iterate */
```

5.3 Subroutines

We give a list and a brief explanation of subroutines used.

- **lecture_node()** and **lecture_link()** read and fill with the input data the array of structures pointed by***tabnode**and***tablink** respectively. **verification()** prints to the standard output the contents of these structures to check for consistency.

- **init_structures()** utilizes the information now pointed by ***tabnode**and***tablink** to adequately initialize pointers between entries of the array of structures(that is, the network topology).
  The subroutine **init_tab_mnode()** initializes the array of structures pointed by ***tab_mnode**(the macronetwork). It is here that when demanding too much memory, the execution can be aborted. There is a special message for these error.

- **graph_search(root)** performs a best first depth search on the macronetwork from the macronode indexed by **root** in the macronode array(pointed by **tab_mnode**). The integer variable **counter** serves to index free space in ***tab_mnode** to write on as it goes through a macropath. Since the first entry on ***tab_mnode** (i.e *tab_mnode[0]*) is the macro-origin, **Root** takes on the initial value of 0, and **counter** is set initially to be one. Since the procedure **graph_search(root)** is to be used extensively when pruning, the real numbers **value1** and **value2** denote respectively, the total routing time through the macronode reached by

the procedure and the best total routing time found so far. Then, when pruning one can stop the depth search when **value1** exceeds **value2** or adequately update.

- **check_optimality()** is the pruning procedure. It traverses the macronetwork (i.e ***tab_mnode**), fixing optimal routing decisions from the macro-origin and on, by calling repeatedly **graph_search(root)**. In this case, the integer variable **counter2**, index the last optimal routing decision on the macronetwork. The procedure returns 1 if an optimal routing has been identified.

With these procedures then the core of the code is :

```
optimality = 0;                    /* setting initial values for basic variables */
counter = 1;
root = 0;
counter2 = 0;

graph_search(root);                /* First Depth Search for a feasible routing */
value2 = value1;                   /* First Routing total cost is stored in value2 */

for(i=0;optimality != 1;i++)       /* pruning sequentially to optimality */
{
optimality = check_optimality();

}
```

5.4 Output

The pruning procedure **check_optimality** marks as it goes through the macronodes reached by the efficient routing (i.e it modifies tab_mnode[].optimality). So the **print_solution()** procedure goes through the macronetwork reconstructing the efficient path. We recompute the travel times accrued along the path have more information on its qualitative features.(Otherwise, one simply has the optimal routing cost stored in **value2**).

6. Appendix 2

In this appendix we provide all the information on the networks considered.

6.1 Links

We use four different types of links, to be denoted A, B, C, D respectively. The following table gives the link impedance funtion for each link type(it is defined in 1.5 minute time units).

Link Impedance Function						
	Number of Vehicles					
Time Units	s = 0	s = 1	s = 2	s = 3	s = 4	freeflow time
Type A	0	0	101.3	160.7	222.2	2
Type B	0	24.2	55.1	85.2	116.8	1.768
Type C	0	0	118.1	198.7	279.5	2.348
Type D	0	22.9	54.3	84.3	115.8	1.808

6.2 Network 1

This is the same network solved in Kaufman et al.[5].It is a four node network with sixty platoons departing along six time periods.In figure 3, the different codes assigned to links and nodes are shown.For further information, the input files are annexed and can be interpreted according to the guidelines given in appendix 1. We now list links codes according to types :

- **Type A** :1, 2
- **Type B** :3, 7
- **Type C** :4, 8
- **Type D** :5, 6

6.3 Network 2

This is ten node network with fifty platoons departing along one time period.In figure 4, the different codes assigned to links and nodes are shown.For further information, the input files are annexed and can be interpreted according to the guidelines given in appendix 1. We now list links codes according to types :

- **Type A** :1, 2, 9, 17, 22, 24
- **Type B** :3, 7, 13, 19, 23, 26
- **Type C** :4, 8, 10, 11, 12, 18
- **Type D** :5, 6, 14, 15, 20, 21, 16, 25

6.4 Network 3

This is twelve node network with eighty four platoons departing along one time period.In figure 5. the different codes assigned to links and nodes are shown.For further information, the input files are annexed and can be interpreted according to the guidelines given in appendix 1. We now list links codes according to types :

- **Type A** :1, 2, 9, 17, 22, 24, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36
- **Type B** :3, 7, 13, 19, 23, 26
- **Type C** :4, 8, 10, 11, 12, 18
- **Type D** :5, 6, 14, 15, 20, 21, 16, 25

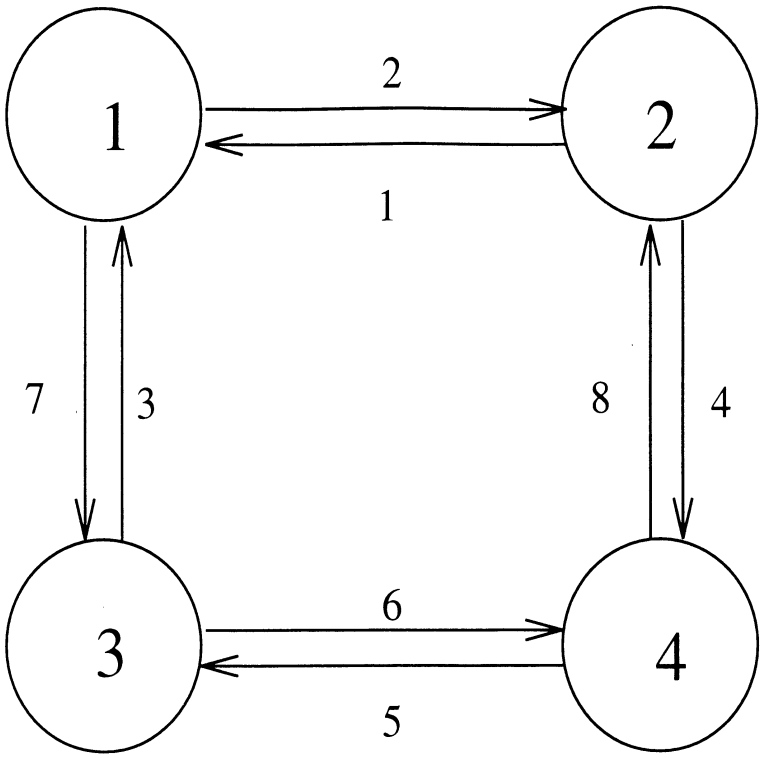

Figure 3: Network 1.

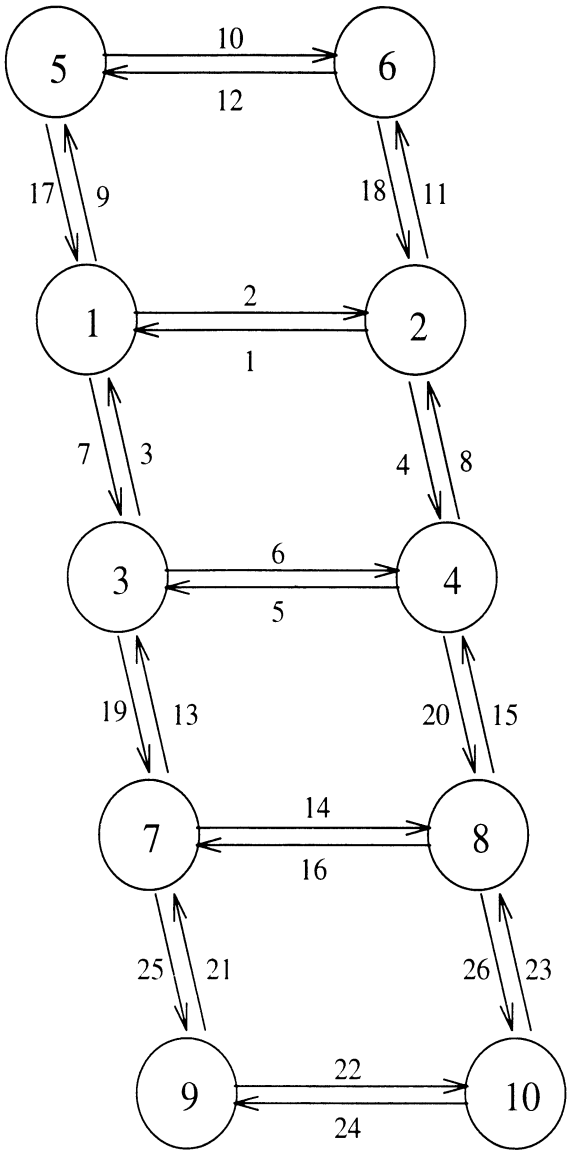

Figure 4: Network 2.

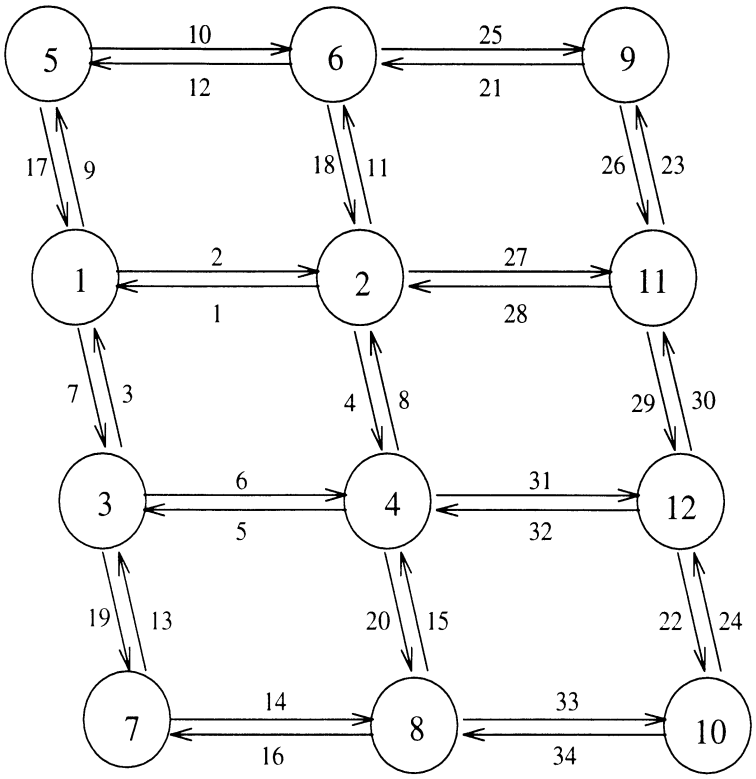

Figure 5: Network 3.