Title
LANE-OPT Users Manual Version 1.0

Authors
Lotspeich, David
Hall, Randolph W.

Publication Date
1996-03-01

CALIFORNIA **PATH PROGRAM**
INSTITUTE OF **TRANSPORTATION STUDIES**
UNIVERSITY OF CALIFORNIA, BERKELEY

LANE-OPT Users Manual Version 1.0

David Lotspeich
Randolph W. Hall
University of Southern California

California PATH Working Paper
UCB-ITS-PWP-96-2

This work was performed as part of the California PATH Program of the University of California, in cooperation with the State of California Business, Transportation, and Housing Agency, Department of Transportation; and the United States Department Transportation, Federal Highway Administration.

March 1996

ISSN 1055-1417

LANE-OPT Users Manual
Version 1.0

October, 1995

David Lotspeich
Randolph W. Hall

Department of Industrial and Systems Engineering
University of Southern California
Los Angeles, CA 90089-0193

ABSTRACT

This document is the user manual for LANE-OPT. LANE-OPT is a linear program based software package that optimally assigns traffic to lanes on an automated highway. The program is based on a workload model, such that each lane/segment has a fixed capacity than can be allocated among: **(1)** straight traffic, **(2)** lane changes into the lane, **(3)** lane changes out from the lane, and **(4)** lane changes that pass through the lane. Each of these four movements produces a user-specified workload. The computer program models the highway as a multi-commodity flow network, where commodities represent destinations. The linear program maximizes total network flow, accounting for lane/segment capacities and a fixed origin/destination pattern, which is expressed **on** a proportion basis.

Keywords: **Automated Highway Systems**
Lane Assignment
Link Layer

EXECUTIVE SUMMARY

LANE-OFT is a linear program based software package that optimally assigns traffic to lanes on an automated highway. The program is based on a workload model such that each lane/segment has a fixed capacity than can be allocated among: (1) straight traffic, (2) lane changes into the lane, **(3)** lane changes out from the lane, and (4) lane changes that pass through the lane. Each of these four movements produces a user-specified workload. The computer program models the highway as a multi-commodity flow network, where commodities represent destinations. The linear program maximizes total network flow, accounting for lane/segment capacities and a fixed origin/destination pattern, which is expressed on a proportional basis.

LANE-OFT can be used in a number of ways:

- Using highway models provided with the system to examine flow differences between different ready-made highway models.
- Examining the effect of lane change penalty parameters upon the highway's performance.
- Modifying the origin-destination matrix that determines the proportion of traffic flowing from each origin to each destination.
- Creating a new highway model for experimentation.
- Making minor modifications to the source code to incorporate unique physical features in a new highway model.

The software was written in the UNIX environment with C programs that can be compiled with the **GNU** C compiler. The optimization code used is CPLEX, which is available from CPLEX Optimization. The linear program (LP) creation program creates a MPS format file that is readable by most optimizers. Some differences in format exist between systems, which may require minor modifications to the MPS file before it is readable.

CPLEX is available for IBM compatibles and many UNIX systems. The major binding factor is the amount of memory that is available for processing. The C programs can use any ANSI-C compatible compiler; however, the size of the highway is more limited in that case because at this time DOS only allows memory to be accessed in 64KB chunks, which limits the array sizes used in the C programs. The C programs have been compiled by Borland C++ v4.5 on a IBM compatible computer with 8 MB of RAM with up to 28 highway segments. Highways with 80 segments and 5 lanes have been successfully optimized on a SUN SPARCserver 1000 with 256 MB of RAM.

TABLE OF CONTENTS

1. Introduction 1
1.1 Purpose of the System 1
1.2 System Requirements 1
1.3 Problem Formulation 2

2. System Organization and Features 4
2.1 Highway Description File 5
2.2 Origin-Destination File 6
2.3 LP Formulation Program 7
2.4 LP Optimization Software 7
2.5 Output Formatting Program 7

3. Example Problem and File Formats 7
3.1 Sample Highway Description File 8
3.2 Sample Origin-Destination Matrix 9
3.3 Sample Experiment Parameters 9
3.4 Running the LP Formulation Software 9
3.5 Solving the LP using CPLEX 10
3.6 Formatting the output 11

A.1 Appendix: Batch Processing for Multiple Experiments 15

1 INTRODUCTION

This manual provides a guide for using the LANE-OPT software package. This software formulates and solves a linear program for maximizing the flow **of** traffic across an automated highway. The highway is broken into a series of contiguous segments, each containing one node for each lane. Arcs interconnect the nodes of a segment to the following segment. The solution provides a detailed trajectory for the movement of cars between lanes in each segment. Traffic is represented as multiple commodities, which are differentiated by their respective destinations. The flow between an origin and a destination is a fixed proportion of total flow set by the user.

1.1 Purpose of the System

LANE-OPT can be used in a number of ways:

- Using highway models provided with the system to examine flow differences between different ready-made highway models.
- Examining the effect of lane change penalty parameters upon the highway's performance.
- Modifying the origin-destination matrix that determines the proportion of trafiic flowing fiom each **origin** to each destination.
- Creating a new highway model for experimentation.
- Making minor modifications to the source code to incorporate unique physical features in a new highway model.

1.2 System Requirements

The software was written in the UNIX environment with C programs that can be compiled with the **GNU** C compiler. The optimization code used is CPLEX, which is available fiom CPLEX Optimization (CPLEX, 1994). The linear program (LP) creation program creates an *MPS* format file that is readable by most optimizers. Some differences in format exist between systems, which may require minor modifications to the MPS file before it is readable. Please examine your software manuals for any questions about the exact form that the *MPS* file should take.

CPLEX is available for IBM compatibles and many **UNIX** systems. The major binding factor is the amount of memory that is available for processing. The C programs can use any **ANSI-C** compatible compiler; however, the size **of** the highway is more limited in that case because at this time **DOS** only allows memory to be accessed in 64KB chunks, which limits the array sizes used in the C programs. The C programs have been compiled by Borland C++ v4.5 on a IBM compatible computer with **8** MB of RAM with up to 80 highway segments. Highways with 80 segments and **5** lanes have been successfully optimized on a **SUN** SPARCserver 1000 with 256 MB of RAM.

13 Problem Formulation

The *AHS* consists of a set of highway segments and sets of on-ramps and off-ramps. Each segment contains one or more automated lanes, which are always situated on the left-side of the highway. The AHS may also have manual lanes, which are always situated on the right-side of the highway. The number of lanes can vary from segment to segment. Lane drops and lane additions are assumed to occur on the right-side of the highway (the model is easily generalized to more complicated structures; see Hall and Lotspeich, **1996).**

As shown in Figure 1 the highway is represented by a flow network. Highway segments are indexed by location and defined by segment type (on-ramp, off-ramp, or neither), length of the segment, number of lanes, and ramp capacity (for segments containing ramps). Nodes are assigned to the end of each lane in each segment, as well as to the start of each on-ramp and end of each off-ramp. On-ramp nodes are source nodes without entering arcs, and off-ramp nodes are **sink** nodes without out-going arcs. Source nodes are also placed at the start of each lane in the first segment, and a "super-sink" node is placed after the last highway segment to absorb all continuing highway traffic. Source nodes at the start of the highway pre-assign entering traffic to a specific lane, whereas the "super-sink" node allows the assignment of continuing traffic to be optimized among lanes (however, in experiments that follow, the highway was assumed to start with zero trafiic).

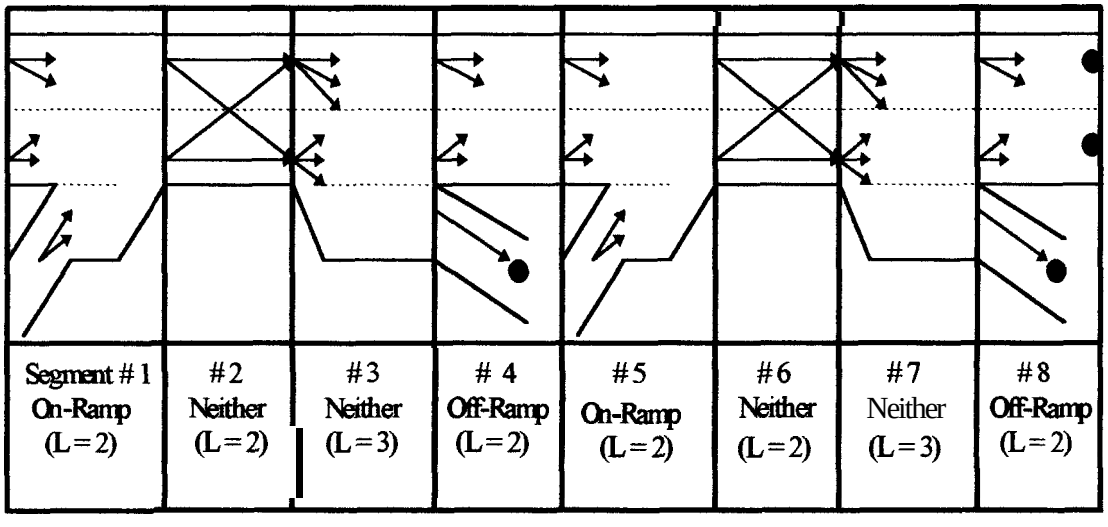

Figure 1: Example highway

Each arc represents a vehicle trajectory through a highway segment. **A** trajectory may entail staying in a lane, transitioning from one lane to another, transitioning from a lane to an off-ramp, or transitioning from an on-ramp to a lane. In each case, the arc is defined by a segment **(s),** an initial lane (i), and an ending lane (j). Hence, an arc (i,j,s) represents vehicles that begin segment **s** in lane i and end segment s in lane j. For any segment, the graph is completely connected, meaning that vehicles are allowed to transition between any pair of lanes within the segment. The exceptions are on-ramp nodes and off-ramp

nodes, which represent sources and sinks. **As** in Figure 1, arcs that are incident on these nodes only flow in one direction.

Arc capacities represent the amount of time-space available for flow to that crosses the lanes determined by the arc. The objective is to maximize the flow across the highway subject to a fixed origin/destination pattern, expressed as a proportional flow OD matrix.

The constraints of the problem are:

- Non-negativity,
- Conservation of Flow,
- Ramp flow capacities that determine the maximum rate at which vehicles can enter and exit the highway at each ramp,
- Workload constraints of each lane in each segment as detailed below.

Capacity constraints bound the flow within each lane/segment. These are bundle constraints, which account for straight traffic, as well as traffic that enters, exits and passes through each lane within each segment. These four types of flow are defined as follows:

$$F^{s}_{stay\,in\,j} = \sum_{d \in D} F^{s,d}_{j,j} \tag{1}$$

$$F^{s}_{enter\,j} = \sum_{d \in D} \sum_{i=1}^{L_s} F^{s,d}_{i,j} \tag{2}$$

$$F^{s}_{exit\,j} = \sum_{d \in D} \sum_{k=1}^{L_s} F^{s,d}_{j,k} \tag{3}$$

$$F^{s}_{pass\,through\,j} = \sum_{d \in D} \sum_{k>j} \sum_{i<j} F^{s,d}_{i,k} + \sum_{d \in D} \sum_{k<j} \sum_{i>j} F^{s,d}_{i,k} \tag{4}$$

The total workload is a linear combination of the flow variables in equations 1 - **4,** which cannot exceed the capacity of the lane/segment:

$$\alpha F^{s}_{stay\,in\,j} + \beta F^{s}_{enter\,j} + \eta F^{s}_{exit\,j} + \gamma F^{s}_{pass\,through\,j} \leq W^{s}_{j} \tag{5}$$

The workload parameters $(\alpha, \beta, \eta, \gamma)$ are all measured in units of work-time per vehicle. The left-hand side of Eq. 6 represents the total time to perform all work in lane j of segment **s** as a ratio to time available. If all parameters are measured in the same time unit, the upper bound on Eq. 6 would be 1. **LANE-OPT** assumes that flow is measured in vehicles per hour, whereas $(\alpha, \beta, \eta, \gamma)$ are measured in seconds per vehicle. Hence, **W** equals the number of seconds in an hour (3600).

$(\alpha, \beta, \eta, \gamma)$ depend on the length of the segment, as well as the workload associated with lane-changes and straight movements. Let:

c_{in} = workload for entering a lane (meter-seconds)
c_{out} = workload for exiting a lane (meter-seconds)
c_{str} = workload for continuing straight within a lane (seconds)
l = length of the segment (meters)

The parameters $(\alpha, \beta, \eta, \gamma)$ are defined by the following:

(1) A vehicle that continues straight in a lane imposes a workload of c_{str} on that lane.

(2) A vehicle that enters a lane imposes a workload of $(1/2)\, c_{str}$, plus a workload of c_{in}/l to account for turn movements into the lane (dividing by l normalizes the turn workload in units of time).

(3) A vehicle that exits a lane imposes a workload of $(1/2)\, c_{str}$, plus a workload of c_{out}/l, to account for the turn movement out from the lane.

(4) A vehicle that passes through a lane imposes no "straight" workload, but imposes a workload of $(c_{in} + c_{out})/l$ to account for turns into and out-from the lane.

Clearly, the assumptions are only approximate, especially for large l where turn-movements might not be uniformly distributed over the length of the segment. The assumptions are summarized below.

Parameter	Parameter Formula
α	c_{str}
β	$c_{in}/l + cJ2$
η	$c_{out}/l + c_{str}/2$
γ	$(c_{in} + c_{out})/l$
W_j^s	3600 seconds/hour

2. SYSTEM ORGANIZATION AND FEATURES

The system comprises 5 elements: a highway description file, an origin-destination file, the LP formulation program, the LP optimization software, and the output formatting program, as shown in Figure 2.

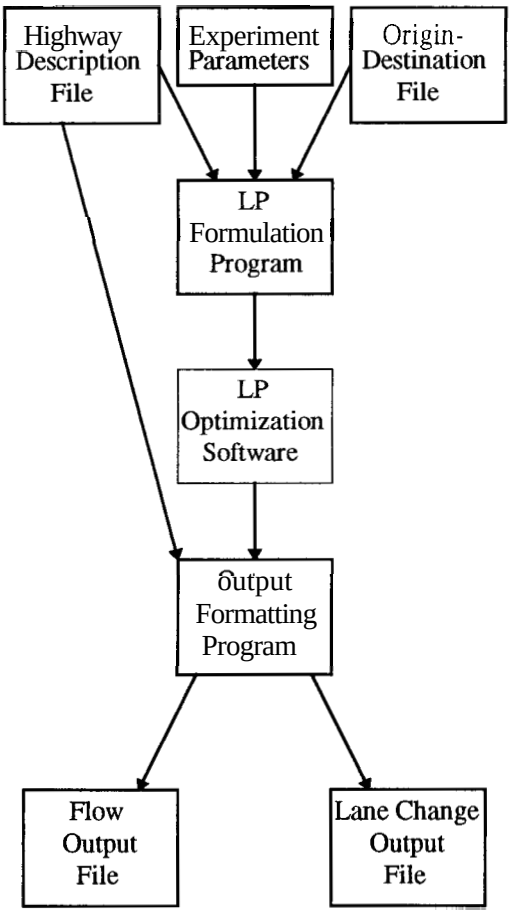

Figure 2: System structure

2.1 Highway Description File

This file provides a static physical description of the highway. It defines a series of highway segments of differing types, including on-ramps, off-ramps, and no ramp segments. Also included are the length of the segment and the number of lanes. There is also the capability to define how many (if any) of the lanes are automated and how many are manually operated. Manual and automated lanes can have different user-defined workload parameters. The highway description file is described by a set of rows, each designating a specific segment, with columns shown:

A ***B*** ***C*** ***D*** ***E*** ***F***

where each column represents the following:

A : Segment index (numbered sequentially starting at 1, 2, ...)

B : Type of segment, with 0 being an on-ramp segment, 1 being an off-ramp segment, and 2 and **3** being no-ramp segments where **3** adds a lane.

C : Length of the segment (meters),

D : Number of manual lanes,

E : Number of automated lanes. and

F : Capacity of on-ramp or off-ramp (vehicles/hour. There MUST be a number in column F even for segments without ramps. However, when there is no ramp, the number is discarded and therefore may take on any value.)

2.2 Origin-Destination File

LANE-OPT stores an origin-destination matrix that gives the proportion of total traffic that passes from each origin to each destination. The file is organized so that each row gives the proportion of total flow passing from each origin to each of the following destinations. There are two types of origins: traffic originating in a lane at the beginning of the highway, and on-ramps along the highway. With L lanes in the initial segment and On on-ramps, the file is organized as follows.

$$\begin{bmatrix} Row\,1: Rightmost\ lane \\ \vdots \\ Row\,L: Leftmost\ Lane \end{bmatrix}$$

$$\begin{bmatrix} Row\,L+1: \textbf{First}\ On-Ramp \\ \\ Row\,L+On: Last\ On-Ramp \end{bmatrix}$$

The first L rows are formatted as follows:

$$P_1\ P_2 \ldots P_{Off}\ P_{Off+2}$$

where Off is the number of off-ramps and P_i is the proportion of total flow that travels from the given origin to off-ramp i. P_{Off+2} designates the proportion traveling to a super-sink node placed at the end of the highway. The super-sink absorbs all traffic that continues on the highway beyond the last off-ramp.

The remaining *On* rows are formatted as follows:

$P_k\ P_{k+1} \cdots P_{Off}\ P_{Off+1}$

where **k** is the index for the first on-ramp after the given off-ramp. Hence, the number of elements equals the number of off-ramps that follow the on-ramp on the highway.

The sum of all proportions across all rows must equal 1.

2.3 LP Formulation Program

LANE-OPT accepts the Highway Description File and Origin-Destination File along with a set of experimental parameters as input. The experimental parameters define the space occupied by vehicles and the additional time-space used in changing lanes, for both manual and automated lanes. These parameters must be expressed in units of meters and seconds, and be consistent in dimensionality to Section 1.3. It then creates a **MPS** format file that is readable by the LP optimization software.

2.4 LP Optimization Software

The LP optimization software reads in the MPS formatted LP file created by the LP Formulation Program. It then optimizes the linear program and finally writes an **ASCII** text file containing the solution information.

2.5 Output Formatting Program

The final stage in the process is performed by the Output Formatting program, which takes the list output of the LP optimization software (see Appendix **A.2)** and formats the results in usable reports. The first report is the aggregate flow report that lists, for each segment/lane, the flow starting in each lane of each segment. This is done first in aggregate for all destinations combined, and then dis-aggregated by destination. The second report highlights the lane changes. This is done by showing for each segment/lane the volume of left and right lane changes out of the lane.

3. EXAMPLE PROBLEM

This section illustrates how to use LANE-OPT with the example highway shown in Figure 1. This example is contained on the program disk included with the manual or can be obtained by writing the authors.

As can be seen in Figure 1, there are two on-ramps (Segments 1 and 5) and two off-ramps (Segments **4** and 8). In addition, the two lanes at the start of the highway are origins and arcs allow flow to continue beyond the modeled highway to further destinations. All distances in the example are measured in meters.

Please note that the UNIX prompt is in the form

unix.prompt(x):

and the statements following the colon are entered by the user. All statements entered by the user are ***italicized.*** In addition, the following naming formats are required:

- **MPS** format file ends with a '.mps' label,
- The output of the LP solver ends with '.dive' label
- The flow output file ends with a '.out', and the lane change output file ends with a '.lr' label.

The rest of this section is organized as follows. First, the file formats and the experiment parameters are given in sections 3.1 - 3.3. Then the steps to executed **LANE-OPT** are shown for the problem. Finally, output reports are analyzed.

3.1 Highway Description File

The highway description file '8_2_0.seg' describes a highway with **8** segments, 2 automated lanes, and 0 manual lanes, as shown in Figure 1.

unix.prompt(1): ***more 8_2_0.seg***
1 0 1000.0 0 2 7200.0
2 2 1000.0 0 2 7200.0
3 3 1000.0 0 2 7200.0
4 1 1000.0 0 3 7200.0
5 0 1000.0 0 2 7200.0
6 2 1000.0 0 2 7200.0
7 3 1000.0 0 2 7200.0
8 1 1000.0 0 3 7200.0

By looking at the format of the first segment we can see the general structure of the file:

1 0 1000.0 0 2 7200.0

- 1 -> The segment index
- 0 -> The segment type, which in this case is an on-ramp.
- 1000.0 -> The segment length in meters.
- 0 -> The number of manually operated lanes.
- 2 -> The number of automatically controlled lanes.
- 7200.0 -> The ramp capacity for any ramp in this segment.

3.2 Origin-Destination File

The highway has two on-ramps and two off-ramps. Assume that 2/3 of the traffic enters at the first on-ramp, half of which exits at the first off-ramp and half of which exits at the second off-ramp. The remaining 1/3 enters at the second on-ramp and exits at the second off-ramp. Further assume that the highway begins and ends with zero flow. The origin-destination file is then formatted as follows:

unix.prompt(2): ***more 8_2.od***
0.0 0.0 0.0
0.0 0.0 0.0
0.333333 0.333333 0.0
0.333333 0.0

The zeros in the first two rows indicate that the highway's two lanes begin with zero flow. The zeros at the end of each row indicate that the aggregate destination receives zero flow (i.e., the highway ends with zero flow). The .333333 values in the third row represent flow from on-ramp one to off-ramps one and two; the .333333 value in the fourth row represent flow from on-ramp two to off-ramp two. Note that the final row only has two elements, as off-ramp one cannot be a destination for on-ramp two (because it precedes it on the highway).

3.3 Sample Experiment Parameters

We will use the following workload coefficients as experiment parameters. The workload for vehicles continuing straight down an automated lane will be set at 0.5 seconds. The additional workload for a lane change maneuver will be 500.0 meter-seconds for vehicles exiting a lane or entering the lane. With these parameters, a lane can accommodate 7,200 vehicles per hour in the absence of lane-changes. The maximum throughput declines as the number of lane changes increases. Because the highway has no manual lanes, workload parameters are not needed for manual lanes.

3.4 Running the LP Formulation Program

The LP Formulation program is titled mpswrite. The following shows how the MPS formatted problem called 'sample.mps' is created.

unix.prompt(3): ***mpswrite***

MPSWRITE
========

What is the filename you want to create?
(.mps ending is automatically attached)
sample

What is the data file containing the Segment Information?
8_2_0.seg

What is the data file containing the Origin-Destination percentages?
8_2.od

Please enter the following workload coefficients.

Of Staying in Same Automated Lane through a segment:
.5

Of Moving Into an Automated lane:
500

Of Moving Out of an Automated lane:
500

Of Staying in Same Manual Lane through a segment:
.5

Of Moving Into a Manual lane:
500

Of Moving Out of a Manual lane:
500

Your file 'sample.mps' has been created.

The resulting 'sample.mps' file created is shown in the appendix A.3.

3.5 Solving the LP using CPLEX

The LP optimization software used for this example is CPLEX. The following statements read in the problem, optimize it, and then write out the solution.

unix.prompt(4): ***cplex***

Welcome to CPLEX Linear Optimizer 3.0
with Mixed Integer Solver
Copyright (c) CPLEX Optimization, Inc., 1989-1994
CPLEX is a registered trademark of CPLEX Optimization, Inc.

Type 'help' for a list of available commands.
Type 'help' followed by a command name for more

information on commands.

CPLEX> read *sample.mps*
Specified Objective Sense: MAXIMIZE
Selected Objective Name: Objectv
Selected RHS Name: RHS
Selected Bound Name: WBOUND
Problem 'sample.mps' read.
Read Time = 0.03 sec.

CPLEX> optimize
LP Presolve eliminated 35 rows and 57 columns.
Aggregator did 8 substitutions.
Reduced LP has 28 rows, 27 columns, and 128 nonzeros.
Presolve Time = 0.02 sec.

Iteration Log . . .
Iteration: 1 Objective = 0.000000

Primal - Optimal: Objective = 7.2000072000e+03
Solution Time = 0.03 sec. Iterations = 2 (0)

CPLEX> write *sample.dive txt*
Solution written to file 'sample.dive'.

CPLEX> quit

The 'readsample.mps' command reads in the **MPS** format file from disk into memory. The 'optimize' command instructs the software to find an optimal solution to the program, and the 'write sample.dive txt' instructs the program to write the output to a file called 'sample.dive' in text format. Remember to have the '.dive' label attached to the end of this filename. Otherwise the output program will not find it. The format of the file 'sample.dive' is listed in the appendix **A.2.**

3.6 Formatting the Output

The final step **is** to format the output of the LP solution to be more easily understood.

unix.prompt(5):***output***

What is the filename (no extension)?
sample
What is the segment file name? [sample.seg]
8_2_0.seg

We are now ready to view the solution.

unix.prompt(7): ***more sample.out***

Variable F
has value of 7200.006836
Variable W
has value of 3600.000000

This section of the output provides the summation of the origin-destination flows across the highway, which is 7,200. The additional fraction of vehicles is due to representation of the flow as a floating point value instead of an integer; it can be disregarded. W echoes the workload capacity, which was set at 3600 seconds/hour.

File = sample.out

Flows to **All** Destinations

Segment		Ramp	Lane 1	2
1	4800	0	0	
2			4800	0
3			4800	0
4	2400	2400	0	
5	2400	2400	0	
6			4800	0
7			4800	0
8	4800	0	0	

The output above shows that all traffic stays in the first lane until it leaves the highway.

Flows to Destination 4

Segment	Ramp	Lane 1	2
1	2400	0	0
2		2400	0
3		2400	0
4	2400	0	0

The flow from the on-ramp in segment 1 to the off-ramp in segment 4 is 2400. This traffic got onto the freeway, went into the first lane, and remained there for the duration of the trip.

Flows to Destination 8

Segment	Ramp	Lane 1	2
1	2400	*0*	0
2		2400	0
3		2400	0
4	0	2400	0
5	2400	2400	0
6		4800	0
7		4800	0
8	4800	0	0

2400 vehicles/hour headed to the off-ramp at segment 8 from the first on-ramp and from the second on-ramp. These vehicles entered the highway in the right-most lane and remained there until they exited.

Flows to Destination 10

Segment	Ramp	Lane 1	2
1	0	*0*	*0*
2		*0*	*0*
3		*0*	*0*
4	*0*	*0*	*0*
5	*0*	*0*	*0*
6		*0*	*0*
7		*0*	*0*
8	*0*	*0*	*0*

The flows to destination 10 (the aggregate destination) are all zero as can be seen.

Output is also provided for workload slack., Each segment has 3600 seconds/hour of capacity. **A** portion of this capacity is used to serve traffic. The slack is the remaining capacity, measured in seconds per hour.

SL's

Segment	Lane 1	2
1	0	3600
2	0	3600
3	600	3600
4	2400	3600
5	600	3600
6	0	3600
7	0	3600

In this example, excess capacity of 600 seconds/hour exists in the first lane in the segment preceding the first off-ramp and in the segment at the second on-ramp. 2400 seconds/hour of capacity remained in the first off-ramp (segment 4). The second lane was unoccupied; therefore, it had slack equal to its maximum value of 3600 seconds/hour.

The "left/right file" provides a table of left and right lane changes by segment and lane (column numbers represent lanes; R and L represent right and left). Not included are flows between the right-most lane and on-ramps/off-ramps. The file also totals the number of lane changes for each segment and keeps **a** cumulative total as well. The final row is another total, for the number of left/right movements in each lane. The file can be examined with the command:

unix.prompt(7): ***more sample.lr***

sample - Left & Rights

Seg 1R	1L	2R	2L	Total–R		Total–L	Cum–R	Cum_L_
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
Tot	0	0	0	0				

As can be seen in this example, no traffic entered the leftmost lane, so there were no lane changes at all.

A. APPENDIX: BATCH PROCESSING FOR MULTIPLE EXPERIMENTS

The authors utilized the Expect interactive program automation toolkit (Libes, 1995) to run multiple experiments very rapidly. This process automated the user response side of program execution. By specifying a file containing all the needed information about each experiment, a set of experiments could be run without user interaction beyond starting the process. The file structure is shown below.

This technique should be used with caution. In larger experiments the mpsfile generation and optimization phases of execution exceeded a waiting time limit imposed by Expect. When this waiting time limit was exceeded the experiment being currently executed was prematurely terminated and the following experiment specified began. Sometimes the results of this did not appear until the next phase when the required input files would either not exist, or not be complete. In general, the automation procedure worked for problems with highways of lengths less than **48** segments.

The Expect script for each stage of the process looked like the following and were executed by typing:

unix.prompt(8): ***expect scriptfile datafile***

where scriptfile is the name of the Expect scriptfile that is to be run, and the datafile contains the parameters such as the highway description filename, the origin-destination filename, and the workload coefficient values. The following scriptfile executes the program using **CPLEX** optimization software.

```
set listfile [open "[lindex $argv 0]" "r"]
set name ""
gets $listfile name
puts %name
while {[gets $listfile name] != -1} {
        if (1) {
        puts $name
        set mpsproc [spawn mpswrite]
        expect "What is the filename you want to create?"
        send "[lindex $name 0]\r"
        expect "What is the data file containing the Segment Information?"
        send "[lindex $name 1]\r"
        expect "What is the data file containing the Origin-Destination percentages?"
        send "[lindex $name 2]\r"
        expect "Of Staying in Same Automated Lane through a segment:"
        send "[lindex $name 3]\r"
        expect "Of Moving Into an Automated lane:"
        send "[lindex $name 4]\r"
        expect "Of Moving Out of an Automated lane:"
        send "@index$name 5]\r"
```

```
        expect "Of Staying in Same Manual Lane through a segment:"
        send "[lindex $name 6]\r"
        expect "Of Moving Into a Manual lane:"
        send "[lindex $name 7]\r"
        expect "Of Moving Out of a Manual lane:"
        send "[lindex $name 8]\r"
        expect "  Your file 'demo.mps' has been created."
        }
        if (1) {
        set cplexproc [spawn cplex]
        expect "CPLEX>"
        send "read [lindex $name 0].mps\r"
        expect "CPLEX>"
        send "optimize\r"
        expect "CPLEX>"
        send "write [lindex $name 0].divetxt\r"
        expect "CPLEX>"
        send "quit \r"
        }
        expect "next:"
        if (1) {
                set outputproc [spawn output]
                expect "What is the filename (no extension)?"
                send "[lindex $name 0]\r"
                expect "What is the segment file name?\n"
                send "[lindex $name 1]\r"
        }
        expect "next:"
}
```

A sample datafile that is specified at the command line looks like the following:

File–Name Seg_File OD–File A_Wkld_stay A_Wkld_in A_Wkld_out M_Wkld_stay M_Wkld_in M_Wkld_out
sample sample.seg sample.od 0.5 200.0 200.0 *0.5* 200.0 **200.0**

The **first** line in the datafile is ignored to allow for column headings. The second line contains the data for each experiment to be run where each line corresponds to a single experiment. **This** datafile will run a single experiment, however if additional lines followed beneath it additional experiments could be run sequentially without interruption **by** the user.

The columns represent the following:

File–Name:	The file name for the experiment.
Seg_file:	The highway description filename
OD–File:	The origin destination filename
A_Wkld_stay :	The Automated Lane workload value for staying in the lane
A_Wkld_in:	The Automated Lane workload value for entering an automated lane
A_Wkld_out:	The Automated Lane workload value for exiting an automated lane
M_Wkld_stay:	The Manual Lane workload value for staying in a manual lane
M_Wkld_in:	The Manual Lane workload value for entering a manual lane
M_Wkld_out:	The Manual Lane workload value for exiting a manual lane

References

CPLEX Optimization, Inc. (1994). ***Using the CPLEX Linear Optimizer***

Hall, R. W., and D. Lotspeich (1996). "Optimized Lane Assignment on an Automated Highway," PATH Working Paper UCB-ITS-PWP-96-03,

Libes, Don (1995). ***Exploring Expect,*** O'Reilly & Associates, Sebastapol, **CA**