Normative Model for Control of Vehicle Trajectory In an Emergency Maneuver

THOMAS B. SHERIDAN and R. DOUGLAS ROLAND
Massachusetts Institute of Technology

This paper proposes a mathematical norm against which driving simulation results (one driver and one vehicle interacting with an environment of fixed and moving obstacles) can be directly compared. The norm is, in effect, an optimal control strategy (in terms of steering, braking and accelerating) given specific constraints on driver's visual span, dynamic equations of the vehicle, and the cost function or trading relation between penalty for colliding with an obstacle and the penalty function of increased control effort or time. The mathematical model, embodied in a digital computer, is being used by the authors in conjunction with both a closed-circuit TV laboratory simulator and actual road tests. To illustrate the idea, some empirical trajectories obtained on the driving simulator are presented together with various computed trajectories which are optimal for particular cost functions and a simplified dynamic model of the automobile. Three appendixes describe the simulator, give comparable data from both simulator runs and road tests with a standard automobile, and provide a simple numerical example of dynamic programming.

•A major hurdle to the satisfactory engineering of private vehicles and highways is the lack of quantitative criteria for driver performance. How far can we trust the driver? Under what conditions do we permit autonomous control of the vehicle by the driver with a modicum of help by signs and traffic laws and policemen, and under what conditions should we by-pass the driver and impose active control on the vehicle from outside, on the basis that the expected behavior of the driver does not fall within satisfactory tolerances?

Two theoretical approaches have borne some relevance to this problem. The theory of traffic flow, deriving from classical fluid mechanics plus developments in applied probability such as theory of queues, predicts flow phenomena based on average behavior of large numbers of vehicles and drivers in relatively steady streams (1). The micro-behavior of one vehicle relative to its environment is not tractable by this approach.

The second approach, the theory of automatic control, has been applied to the individual driver, but in a somewhat misguided way. While quasi-linear differential equations are now available which predict the human controller's response surprisingly well in the context of flying aircraft and spacecraft along smooth, well-charted paths (2), no such success has been achieved in the automobile driving context. The explanation is simple. In the former case the human controller's inputs are well defined, continuous, single-valued functions of time, and "tracking" or "error nulling" in the closed-loop servo-mechanism sense is not too bad an explanation of what the human is actually doing (though there is some room for argument here, and the aircraft takeoff and landing operations do not fit the servomechanism paradigm). If automobile driving were a matter of following a white line across the California salt flats, the servomechanism model would fit.

Paper sponsored by Committee on Driving Simulation and presented at the 45th Annual Meeting.

THE PROBLEM

Unfortunately, keeping the vehicle on the road is not the crux of the driver's problem, and the driver's more difficult tasks are to:

1. Visually identify the obstacles in his immediate environment as to the penalty for hitting them, relative to costs of effort and time expenditure;
2. Predict the future course of those obstacles which are in motion, i.e., other vehicles and pedestrians;
3. Chart a course through all the obstacles which will minimize the expected penalty over some "interval of predicament"; and
4. Control his vehicle to effect the programmed trajectory.

It is these requirements for preview of the input and preplanning of control which make driving a car a more sophisticated control task than keeping an aircraft or spacecraft on a command course by error nulling.

The problem, it would appear, can be formulated in terms of the currently developing theory of optimal control. Given an initial state (set of initial conditions), a terminal state or range of allowable terminal states, and a cost function which specifies the incremental costs and constraints in moving from any one particular state to another in the space between initial and terminal states, the minimum cost trajectory (and the optimal control strategy) can be straightforwardly and uniquely specified. Included in the cost function are (a) the costs of colliding with each of the fixed or moving obstacles; (b) the cost of time in getting between initial and terminal states; (c) the constraints on maximum forward accelerating and decelerating forces and sideward accelerating force; (d) the manual control dynamics (steering wheel or pedal displacement to vehicle position). The suggested means for computing the optimal trajectory, because of its generality, is the dynamic programming algorithm of Bellman (3), and the means for implementing this algorithm is a digital computer. These techniques will be described by example.

Having once determined an optimal trajectory, one can observe in a simulated or controlled vehicle experiment with a human subject, how much and in what way the human driver deviates from the optimal control strategy (or the vehicle deviates from the optimal trajectory). Alternatively one can assume the human is an optimal controller (we know in the simple servomechanism case he is not far off) but that he is subject to internal constraints additional to those posed by the vehicle and environment. In this case one can try to find under what additional constraints the human driver is optimal. (This so-called "indirect problem" of optimal control is less straightforward than the "direct problem.") With either orientation, the aim is to discover the human's cost trade-offs in a particular context of vehicle control. This cost or criterion problem is not a consideration of available models of the human operator. In the preview situation it would seem to be the governing factor.

The theoretical problem will be formulated under one set of assumptions, then the computation technique will be described. Several alternative formulations will then be given, and finally some associated problems of experimentation will be discussed.

CASE 1: COMPLETE PREVIEW, DETERMINISTIC PREDICTION

Physically, the problem is to start from the origin of an xy Cartesian space at some initial time t_o (Fig. 1) and chart a best course down the road. It will be convenient to let x, y and t have values only at discrete points at regular intervals Δx, Δy, Δt. There are k obstacles, some fixed (beer cans, parked cars, etc.) and some moving (vehicles, pedestrians, bicycles, etc.). We will assume in this first case that all obstacles are within the driver's preview and that all obstacles maintain constant derivatives so that the positions of all obstacles at each time t are determined (can be predicted) by the driver at t_o:

$$\begin{aligned} x_k(t) &= x_k(o) + \dot{x}_k(o)t + \ddot{x}(o)\,\frac{t^2}{2} + \cdots \\ y_k(t) &= y_k(o) + \dot{y}_k(o)t + \ddot{y}_k(o)\,\frac{t^2}{2} + \cdots \end{aligned} \tag{1}$$

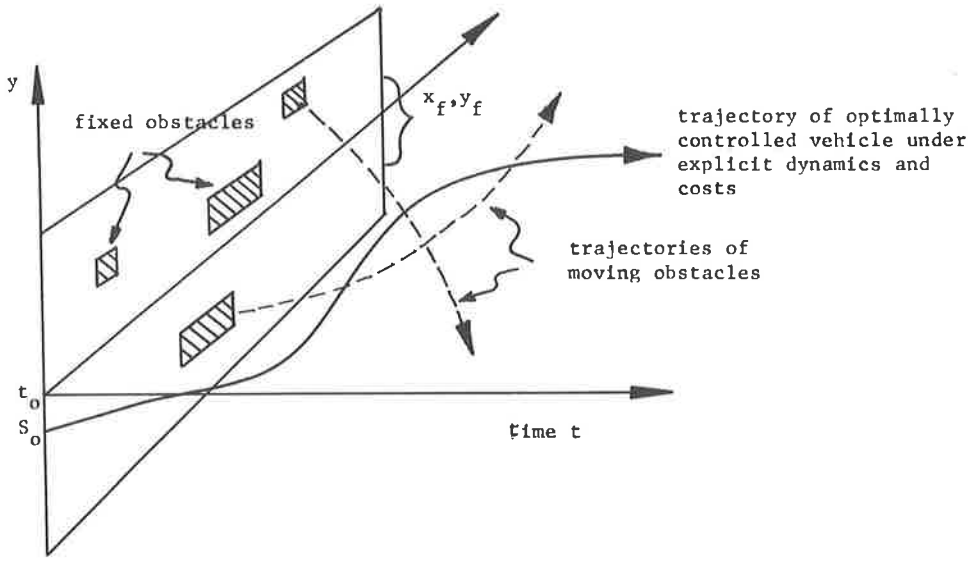

Figure 1. Physical space of desired trajectory and obstacles.

where x_k and y_k are the forward and lateral positions of the kth obstacles. For each obstacle in any given increment of time Δt we assume the driver perceives a cost C_k, which may have a value only for a collision, i.e.,

$$C_k = c_k, \quad | x-x_k | < w_k \text{ and } | y-y_k | < \ell_k; \; C_k = 0 \text{ otherwise} \tag{2a}$$

or C_k may be a function of the miss distance from some obstacles or targets, i.e.,

$$C_k = K_{kx} | x-x_k |^{\alpha} + K_{ky} | y-y_k |^{\beta} \tag{2b}$$

where x and y specify position of the controlled vehicle and w_k and ℓ_k represent effective length and width of the kth obstacle.

We assume some equations of motion for the vehicle, such as

$$\begin{aligned} A\ddot{x} + B\dot{x} &= f_x \\ C\ddot{y} + D\dot{y} &= f_y \end{aligned} \tag{3}$$

where f_x and f_y are tire forces. A simplest assumption for the sake of our example is that steering dynamics are sufficiently "tight" and accelerating or braking sufficiently fast that the driver can command f_x and f_y directly, i.e., the dynamic lags of the human neuromuscular response and steering, tracking and accelerating mechanisms are not appreciable. (The elemental human reaction time of, say, 0.25 sec is compensated by the driver's preview and anticipation.)

Since the driver is constrained by fuel expenditure, tire wear and social criticism from too violent use of the steering wheel, brake and accelerator pedals, we must assume an "effort" cost over a unit time increment, such as

$$C_e = K_{ex} | f_x^3 | + K_{ey} | f_y^3 | \tag{4}$$

The values f_x and f_y may be constrained to be less than certain limiting values, $f_{x\,max}$ and $f_{y\,max}$, which represent skidding.

There is one more cost to consider, a cost C_T per unit of time to reach a terminal state $x(t_f)$, $y(t_f)$, $\dot{x}(t_f)$, $\dot{y}(t_f)$ or one of a set of such states. This represents how big a hurry the driver is in.

It is convenient to consider the total cost function as the cumulative cost up to last time increment plus the incremental costs for collision, for effort and for time

$$C(t) = C(t-1) + \Sigma_k C_k(t) + C_e(t) + C_T(t) \qquad (5)$$

$C(t-1)$	$\Sigma_k C_k(t)$	$C_e(t)$	$C_T(t)$
cumulative cost to last time stage	incremental collision cost	incremental effort cost	incremental time cost to reach required terminal stage

It is the ratio of the weightings on these costs which we seek to determine for the human driver in various situations. For example, we know that the weighting C_k he attaches to collisions is very high. Letting $C_k \to \infty$, there is still presumably a variety of control strategies which will trade off between how much of a hurry he is in, C_T, and his reluctance to be a wild driver, C_e.

DETERMINING OPTIMAL TRAJECTORIES BY DYNAMIC PROGRAMMING FOR GIVEN OBSTACLES, GIVEN VEHICLE DYNAMICS, AND GIVEN COST WEIGHTINGS

For purposes of computing a minimum cost trajectory it is convenient to think in terms of a state space, a space of sufficient variables (and derivatives) of the system at each stage in time that transition from given state S(t) at time t to another given state S(t + 1) at t + 1 completely specifies the cost incurred over that time interval Δt.

C_T is dependent only upon whether the present state satisfies the terminal conditions, Sf. C_k for a collision with an obstacle positioned at x_i and y_j depends only on $S_{ij}(t) = x_i(t), y_j(t)$. C_e for a jump (Fig. 2) from a state $S_{ijuv}(t-1)$ to a different state $S_{IJUV}(t)$ depends upon $S_{ijuv}(t) = x_i(t), y_i(t), \dot{x}_u(t), \dot{y}_v(t)$ since the effective $\dot{x}_{iI}$ across the intervals is $x_I(t) - x_i(t-1)$ and the effective $\dot{x}_{uU}$ across the interval is $\dot{x}_U(t) - \dot{x}_u(t-1)$, and similarly for $\dot{y}$ and $\ddot{y}$ in Eq. 3. Thus at each time stage t we need a four-dimensional space in S_{ijuv} with sufficient range of the variables chosen artfully at the outset to include those states through which the optimal trajectory has reasonable probability of passing (Fig. 2).

The basic idea of the dynamic programming algorithm is best explained by example. For the reader not at all familiar with this optimization procedure a numerical example is provided in Appendix C.

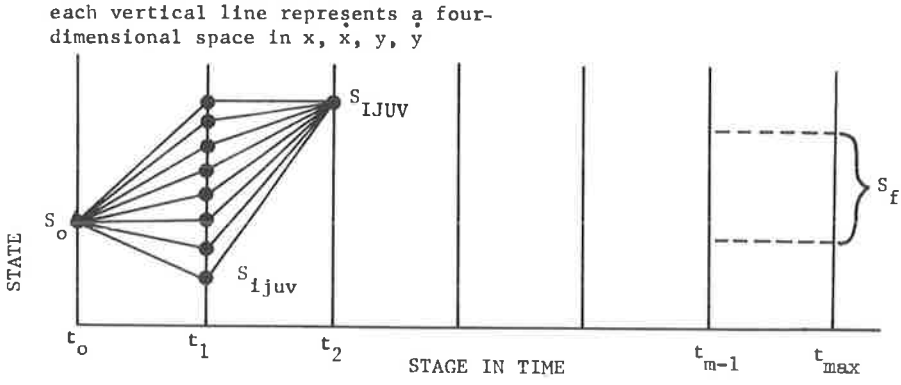

Figure 2. State space of desired trajectory and obstacles, showing trial paths to determine least cost path to $S_{IJUV}(t_2)$: IJUV is a particular state to which all ijuv states at the previous stage are considered paths.

In the present problem the dynamic programming algorithm proceeds as follows. We start at either end of the time range of interest, i.e., at $S_o(t_o)$ or at $S_f(t_{max})$; it does not matter unless there is no prespecified terminal condition. Assuming we start at t_o, the cost of jumping to each next start $S_{ijuv}(t_1)$ is determined from Eqs. 1, 2, 3, 4, and 5, and each resulting value $C_{ijuv}(t_1)$ is stored. Next we consider in turn each of the states S_{ijuv} at t_2, to determine the path by which the least cost trajectory arrives at that particular $S_{IJUV}(t_2)$. In other words we choose the one of all $S_{ijuv}(t_1)$ from which to set out for the particular $S_{IJUV}(t_2)$ under consideration in order that the cumulative cost C_{IJUV} at t_2 be least. This involves a brute force application of Eqs. 1, 2, 3, 4, and 5 to each and every S_{ijuv} at the last stage paired with the single S_{IJUV} at the present stage. Since every C_{ijuv} for the last stage is already in memory, the least cost is easily obtained following Eq. 5 as

$$C_{IJUV}(t) = \underset{ijuv}{\text{minimum}} \left[C_{ijuv}(t-1) + \underset{\substack{\text{from ijuv at } (t-1) \\ \text{to IJUV at } (t)}}{(\Sigma_k C_k + C_e)} + \underset{S \text{ not an } S_f}{C_T} \right] \tag{6}$$

where now $t = t_2$. This equation is essentially a statement of Bellman's principal of optimality. The optimal path is then simply the previous state by which to obtain least cost for the present state. This value is stored:

$$P_{IJUV}(t) = \underset{\text{minimum}}{[ijuv]} \tag{7}$$

These cost and path values are stored in memory and the process is repeated for every S_{ijuv} in the state space at t_2. We then have in memory a least cost to get to each S_{ijuv} at t_2 and the corresponding best path to get there. The cost information at stage t_1 may then be thrown away.

The whole process is repeated at each successive stage forward in time, until t_{max} is reached. At this last computation stage, either only a unique state S_f is allowed, or the least cost state within a set of states S_f is chosen. Since any stored path P(t) specifies the best state at the adjacent time stage, and P at that time specifies the best state at the next time, and so on, the optimum path is easily traced through the stored table of best paths. A computational flow chart is shown in Figure 3.

The great saving of the dynamic programming algorithm is that the amount of computation is a linear progression with T and not a geometric progression. For N states at T times only N^2T comparisons need be made, not N^T, the number of different trajectories, and, more importantly, only NT values of path P need be stored, not N^T values. The dynamic programming algorithm can handle with ease various essential nonlinearities in the vehicle equations and cost functions which some other optimization techniques cannot.

EXEMPLARY FITS OF THE MODEL TO DRIVING SIMULATOR DATA

Using a driving simulator (described in Appendix A), an experienced subject was instructed to drive a straight course down the center of an open track, and that upon reaching a certain point along the track two "targets" (negative obstacles or obstacle "holes") would suddenly appear before him at fixed positions down the road. He was to overrun the targets with equal penalty for lateral errors on both targets. He controlled only steering. His forward velocity was held constant. The subject had been trained in the same experiment and knew that the location of the targets was a random distance from the center of the track and was equally probable to the left or to the right. He also knew that in some cases both targets appeared at once (type A) and in other cases the second appeared just as he started responding to the first (type B), which was actually ½ sec after the first appeared.

Figure 4 is a plot of an ensemble average of ten runs of type A for one particular location of the targets (heavy crosses) and an average of ten runs of type B for the same

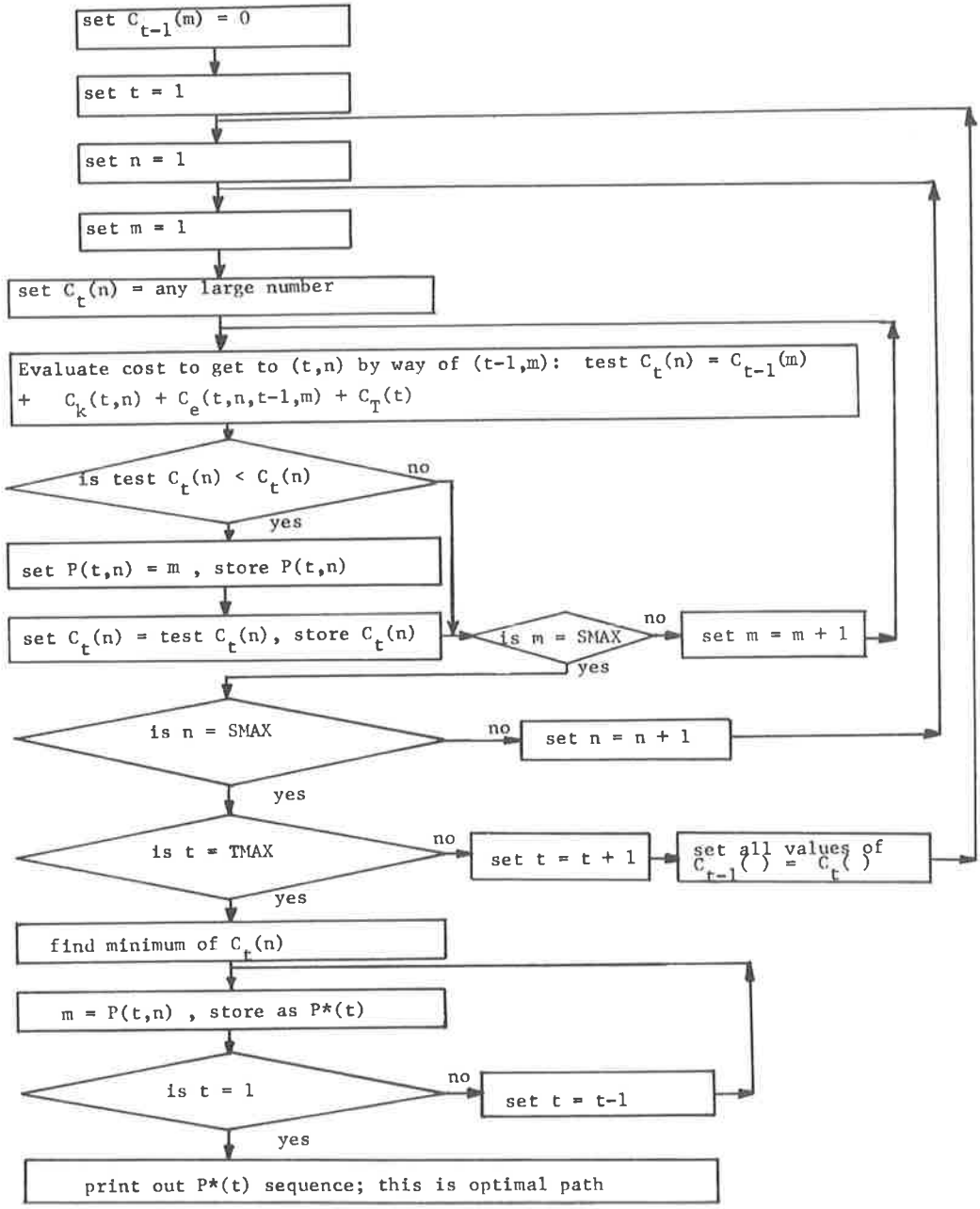

Figure 3. Computational flow chart for dynamic programming example: m is a running index for every ijuv state at stage t-1, n is a running index for every IJUV state at stage t.

target locations. Note that the ordinate and abscissa are not the same scale. As indicated, these runs had been interspersed with runs for other target locations.

On the same plot are exemplary optimal trajectories for various performance cost functions. The latter are based on the extremely simple model in which the lateral position of the vehicle (the ordinate of the graph) is the second integral of the vehicle's front wheel (or steering wheel) position. The vehicle's forward velocity (rate of motion along

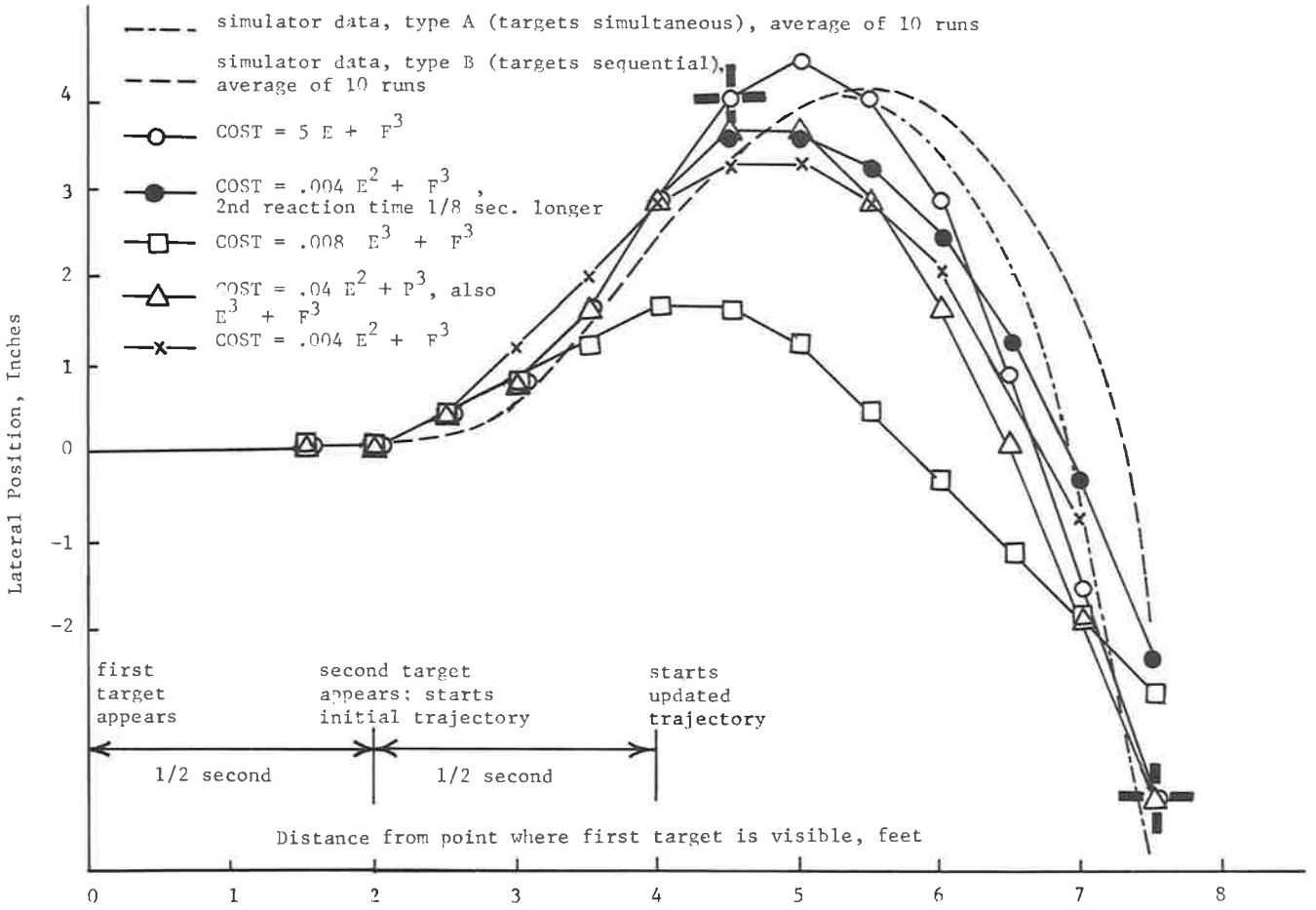

Figure 4. Examples of optimal trajectories for various cost functions of error (E) and effort (F) compared with driving simulator data.

abscissa) is constant. All of the computed optimal trajectories shown presume a type B stimulus situation where the second target appeared ½ sec after the first appeared. Our model assumes that the driver planned an optimal trajectory with respect to the first target which is to begin after his own ½ sec reaction time (after 2 ft). The finite position change is therefore at the 2½-ft point. At the same time he started his initial optimal trajectory the second target appeared, but he could not initiate an updated trajectory until after his ½ sec reaction time.

The empirical data seem to suggest that for all cost functions the trajectory updating occurred after a longer delay with respect to the appearance of the second target than with respect to the first target. However, it is not the authors' purpose to draw quantitative conclusions from these data but only to note qualitatively how different cost functions result in different forms of approximation to the empirical data.

An experiment similar to that described above was performed with a standard automobile in a large parking lot. This experiment is described in Appendix B, and some results are given which compare vehicle trajectories of the actual car with those of the driving simulator in the same type of task.

ARTFUL USE OF COMPUTER MEMORY

The dynamic programming algorithm poses severe demands on computer memory, and it is important to understand these if practical solutions are to be implemented. For example, suppose in the present problem x and y were reticulated into 40 increments, $\dot{x}$ and $\dot{y}$ into 10 increments, and t into 100 increments. Then we must keep in memory at any one time the least costs of every state at two adjacent time stages, requiring $2 \cdot 40 \cdot 40 \cdot 10 \cdot 10 = 320{,}000$ locations. In addition we must store paths for every state at all 100 time stages, or 16,000,000 locations. Clearly we cannot put all of this in the core memory. (Of course this is still better than separately evaluating the $160{,}000^{100}$ separate trajectories which are possible.)

Fortunately there are ways out. These will not be detailed here, for they would comprise at least another paper this size. Briefly, however:

1. One can store in core memory the cost information, i.e., the C_{ijuv} values of only a small part of the state space at only the two stages where least costs are being computed, retaining the rest on tape or disk files and playing it back into core when needed. Suppose one could allocate 1000 core locations to cost values at each of two adjacent stages and do tape-to-core and core-to-tape transfer in blocks of 1000. The number of block transfer cycles to make all possible comparisons of states at adjacent stages is the square of the ratio of the state space size to the size of the memory block, which is 160^2 in our present example. More block transfers would be necessary for storing paths.

2. To save storing on tape the required 16,000 blocks of best paths, the 160,000 path specifications at each stage can be printed out on paper. A human clerk should be able to trace from one stage to the next in less than a minute since he is told by the present P(t) specification precisely where to go for the next P(t). But if only a one-character printer is available at 10 numbers per sec, printing the required 160,000 best paths would require close to 5 hours of printing per stage plus the paper, clearly not a practical approach for problems of this size, but certainly feasible for smaller problems when long-term computer memory is not convenient (or when a high-speed printer is!).

3. The state space can be broken into blocks (hyperspace rectangles) and optimal trajectories determined for a number of Δt intervals within these blocks by interpolation techniques, storing in each case the costs and paths to get to the surfaces of these blocks. This technique presumes a preferred direction of motion through the state space and still requires storage of optimal costs and paths in a large number of states at different time stages, but it greatly reduces the high-speed storage requirement (4).

4. Nominal trajectories can be drawn through the state space and improved trajectories only in the neighborhood of the nominal trajectory can be considered. Thus, if we consider 100 time-units in a state space of 5 increments in both x and y and 3 increments in both x and y, centered on the nominal trajectory, we have a state space of only 225 numbers. The dynamic programming solution is now easily within reach of the smallest computer (far better than brute force comparison of 225^{100} possible trajectories).

This small state space even makes practical the print-out technique of method 2 above if it is necessary. Successive iterations can be made to keep improving the current "optimal" trajectory (though one must be careful in order to avoid converging on sub-optimal paths, a built-in limitation of all steepest ascent techniques in which the best path is not within the range of original consideration). Much experimentation remains to be done using people in roles of artfully posing nominal trajectories and working with the machine in real time to improve them.

CASE 2: COMPLETE PREVIEW, PROBABILISTIC PREDICTION

Having looked at some computational aspects of the original formulation, let us return to a slightly different formulation. In case 1 we assumed that at t_o the future positions of all moving objects are known by direct extrapolation on present states and a perfect optimal control strategy is computable from the outset. However, we may alternatively assume the human driver does not plan his control on precise extrapolations, that he knows he could not make accurate prediction anyway. Therefore, let us assume that the locations (of the centers) of various moving obstacles are known only on a probabilistic basis. In particular let us assume a two dimensional unimodal probability distribution with its mean at $x_k(t)$, $y_k(t)$ and with a variance which increases with time. Thus, in evaluating the penalty for a potential collision with obstacle k,

$$C_k = c_k' \exp \frac{-[(x - x_k)^2 + (y - y_k)^2] \quad (t + K_k)}{R_k} \tag{7}$$

This adds a growing uncertainty or smudge to the prediction of obstacle positions with parameters R_k (a "rate of smudge") and K_k (initial "smudge") in lieu of the length and width parameters ℓ_k and w_k in Eq. 2a. This formulation presumes the human driver is unable to take in new information and update the optimal control strategy during the course of any one predicament time interval, t_m.

CASE 3: COMPLETE PREVIEW, UPDATING OF TRAJECTORIES TO COMPENSATE FOR EXTERNAL INFLUENCE ON OTHER OBSTACLES

When the potential collision predicament lasts more than several seconds there is time for the driver to update his optimal control strategy based on new estimates of the positions and velocities of moving obstacles. If the kth obstacle takes some arbitrary path (subject to about the same dynamic constraints as the vehicle whose control we are interested in), after some time t_u the driver would make a new extrapolation, analogous to Eqs. 1:

$$\begin{aligned} x_k(t - t_u) &= x_k(t_u) + \dot{x}_k(t_u)\ (t - t_u) + \ldots \\ y_k(t - t_u) &= y_k(t_u) + \dot{y}_k(t_u)\ (t - t_u) + \ldots \end{aligned} \tag{8}$$

Then he would take $S_u(t_u)$ as an initial state and recompute an optimal trajectory based on the newly predicted obstacle positions at future time stages, with or without "smudge" considerations on his predictions. This might be repeated several times during the course of a total predicament interval t_{max}. The optimal trajectory in each case would be based upon the range of x remaining.

CASE 4: CONSTRAINED PREVIEW, UPDATING OF TRAJECTORY

Suppose that either because of poor visibility or limited perceptive capability the driver can attend only to obstacles less than x_{lim} ahead. If x_{lim} were short relative to the total range of x involved in the present collision predicament, we would certainly expect the driver to update his optimal trajectory as often as he could, each time basing his optimal strategy on all the obstacles in his present preview—some of those he saw on the early computation plus the new obstacles which the "moving window" of his preview

might now reveal to him. All of the exemplary trajectories given earlier included one updated computation based on new evidence when the second target (obstacle "hole") appeared.

CASE 5: SIMULTANEOUS OPTIMAL CONTROL OF TWO VEHICLES WITH UPDATING OF TRAJECTORIES

Suppose, in case 3, that one of the k obstacles, instead of taking an arbitrary path, is a vehicle being controlled optimally with respect to its driver's preview. And suppose, further, that the drivers of both vehicles update their optimal control strategies periodically during a collision predicament. The study of certain cooperation/competition, or "social" aspects, of driving would seem to be tractable in terms of the costs netted each of the drivers as a function of the various parameters of the control situation previously described plus the updating interval t_u.

AN ORIENTATION FOR EXPERIMENTS

The aim, to restate, is to determine how much and in what way the human driver deviates from an optimal controller (the direct problem), or under what additional constraints to those posed by the vehicle and environment the human driver is optimal (the indirect problem), and by either means discover the nature of the driver's control cost criteria. It has been shown that the greatest difficulty in implementing computation of optimal control strategies or trajectories by dynamic programming is the computer memory requirement, and that the use of nominal trajectories artfully posed by a human experimenter can reduce the memory requirement to a very workable level. Suppose a driving simulation is set up under careful control such that the dynamic parameters of Eqs. 3 are few and known, the states of the k obstacles are known, the constraints on $f_{x\ max}$ and $f_{y\ max}$ are represented, and the relative costs assumed at the start of the experiment for C_e ("wild driving"), C_k (collision) and C_T (elapsed time) are at least realistic and are made explicit to the driver. Then empirical trajectories recorded from the simulation can be used directly as nominal trajectories for an optimal control computation in a digital computer using the same parameters.

Having available such relative cost weightings for the human operator under a reasonable range of circumstances, one can speculate on their use for design purposes. A group of transportation designers in conference around a scale model simulation of a highway intersection (or the futuristic equivalent) could pose different configurations to the highway, the vehicles, the traffic signals, etc., by physically changing objects in the simulation, or typing into a computer conditions of speed, weather, etc., for sample vehicles. The answer to their "what would happen if" questions would be produced in short order by the computer and displayed by a mechanical plot of probable trajectories of the various vehicles.

ACKNOWLEDGMENT

The work herein presented was sponsored in part by the U.S. Public Health Service and in part by the National Aeronautics and Space Administration.

REFERENCES

1. Herman, R., and Gardels, K. Vehicular Traffic Flow. Scientific American, Vol. 209, No. 6, December 1963.
2. McRuer, D., Graham, D., Krendel, E., and Reisener, W. Human Pilot Dynamics in Compensatory Systems. AFFDL-TR-65-15, May 1965.
3. Bellman, R., and Dreyfus, S. Applied Dynamic Programming. Princeton University Press, Princeton, New Jersey, 1962.
4. Larson, R. E. Dynamic Programming With Reduced Computational Requirements. IEEE Transactions on Automatic Control, AC-10, 2, April 1965.

Appendix A

A BRIEF DESCRIPTION OF THE LABORATORY DRIVING SIMULATOR

The driving simulator consists of a 16-inch long vehicle (Fig. 5a) which carries a television camera. Its front wheels are steered remotely by means of a servomechanism, and it is accelerated and braked remotely by means of an electric motor and an electric clutch. The vehicle is designed to have steering dynamics similar to those of a conventional automobile. The vehicle is driven along a 40-ft long model roadway (Fig. 5b) and into a "test section" where a variety of fixed and moving objects can be made to suddenly appear; e.g., cardboard "obstacles" (or, alternatively, "targets") are driven up through slits in the test track at random positions by means of electronically timed solenoids. The electrical umbilical is suspended from an overhead rail in such a way that it does not cause a significant drag force.

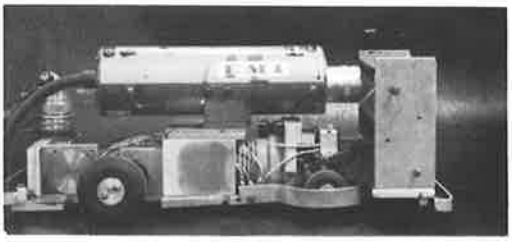

Figure 5a. Servo-controlled model vehicle with vidicon and periscope.

The experimental subject sits in a darkened cab controlling the vehicle with actual steering wheel, brake and accelerator (Fig. 5c) and has a 30-degree view in the TV monitor equivalent to what he would see if seated in the model vehicle.

Timing is accomplished by means of photocells strategically placed along the model roadway. Records of the vehicle's trajectory are generated by a device on the vehicle which makes a continuous mark on a piece of black paper laid along the test section of the model roadway.

Figure 5b. Model vehicle approaching targets in test section.

Figure 5c. Experimental subject in darkened booth viewing targets on TV.

Appendix B

COMPARISON OF TRAJECTORIES FROM SIMULATOR AND FULL-SCALE VEHICLE TESTS

A standard automobile (late model Ford Mustang) was accelerated along a straight course (in a large parking lot) to about 40 miles per hour. The driver's forward vision was blocked; he maintained lateral alignment by vision through the side window. At a predesignated point along the course the driver's forward view was restored by a passenger and the driver attempted to overrun two successive targets (rubber lane marking cones), centering them on the front bumper, with equal importance attached to lateral position errors for both targets.

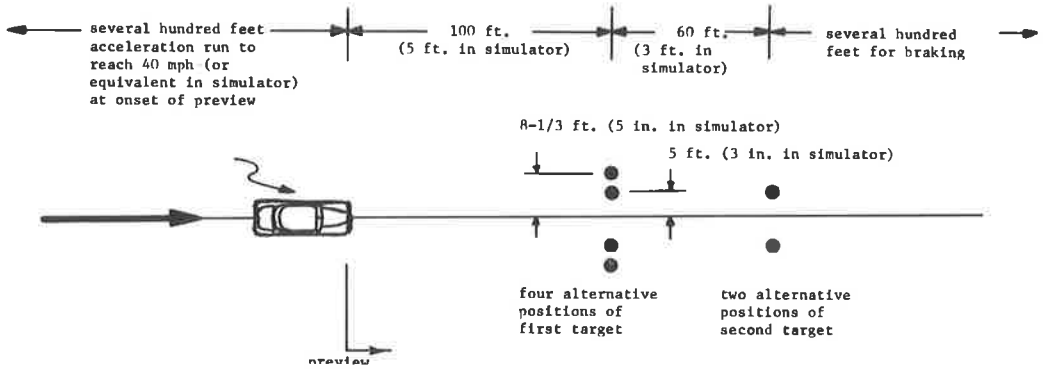

Figure 6. Plan view of experimental course.

A plan view of the experimental course is shown in Figure 6. As indicated, the analogous course and task were set up on the driving simulator described in Appendix A. (This was a different group of subjects and slightly different task from that described earlier in the body of the paper.) The first target was randomly placed in one of the four alternative positions shown and the second target in one of the two alternative positions shown.

On each run with the standard automobile a record was obtained (using stripes of toothpaste spread on the parking lot) of the position and the angle at which the vehicle crossed each of the target lines. Comparable measurements were made using an ink marking device built into the driving simulator.

Experimental results from four of the eight alternative target configurations are shown in Figure 7. This set of four all started to the left; the target configuration in the four not given was the mirror image, and the resulting trajectories were roughly comparable

Comparison of results from the full-scale tests and from the driving simulator revealed one striking difference: there was a marked tendency to oversteer the driving simulator by the relatively inexperienced subjects used. (It will be noted in reference to the simulator results given earlier for a comparable task with an experienced subject that there was no such tendency to oversteer.) This is believed to have been due primarily to the lack of kinesthetic and vestibular cues (velocity and acceleration senses) available to the driver of the simulator, for these senses are known to play the same role that artifical rate feedback does in automatic control of otherwise undamped system

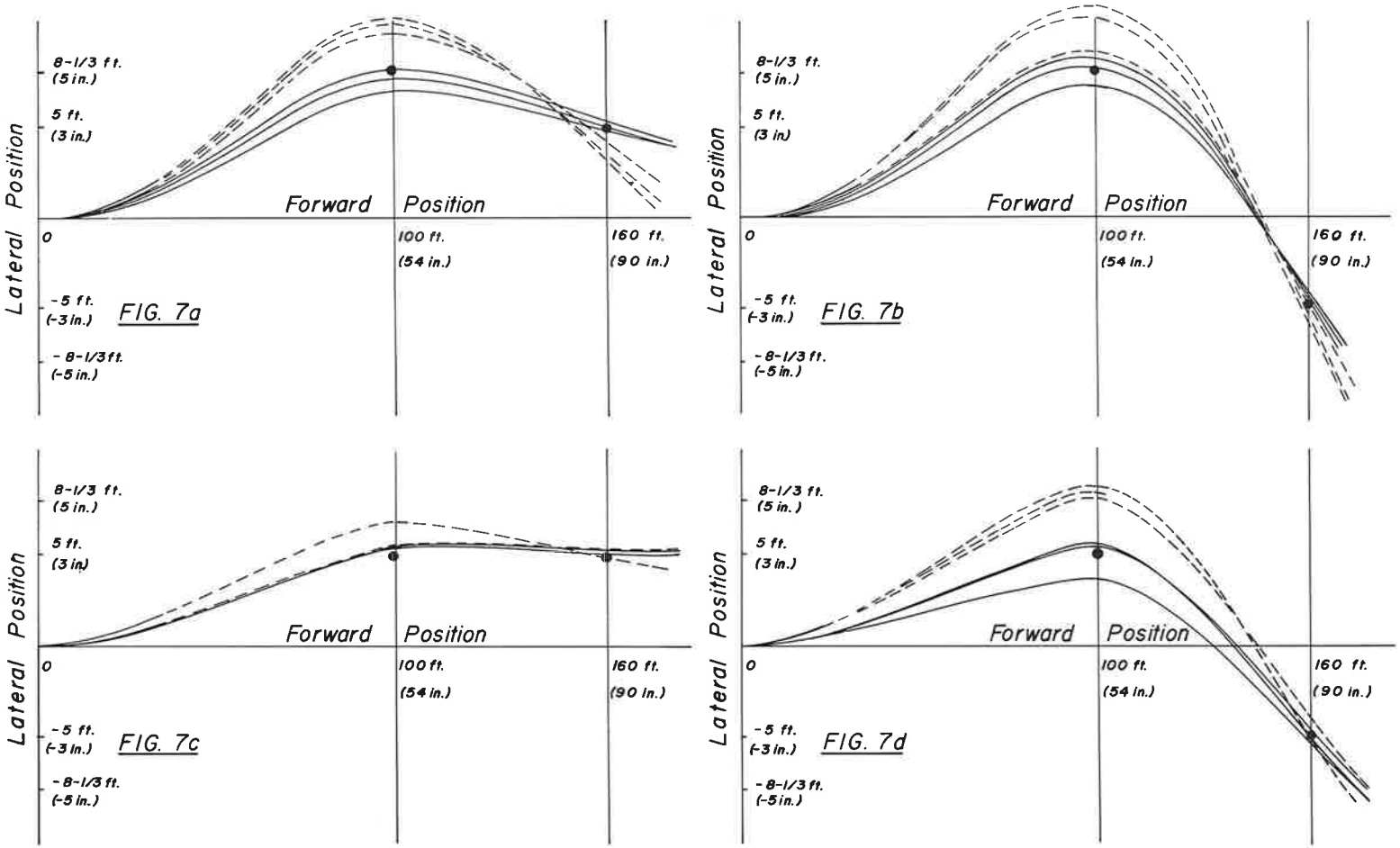

Figure 7. Trajectories from simulator and full-scale vehicle. Key: dashed line and numbers in parentheses = simulator; solid line = full-scale vehicle.

Appendix C

A SIMPLE NUMERICAL EXAMPLE OF DYNAMIC PROGRAMMING

Assume at each of four stages in time after the initial stage there are three possible states of a system, represented by circles in Figure 8, and that from each state at a given stage, the cost to get to any state at the next stage is directly a function of the two states. These costs are represented by the numbers along the lines connecting the states

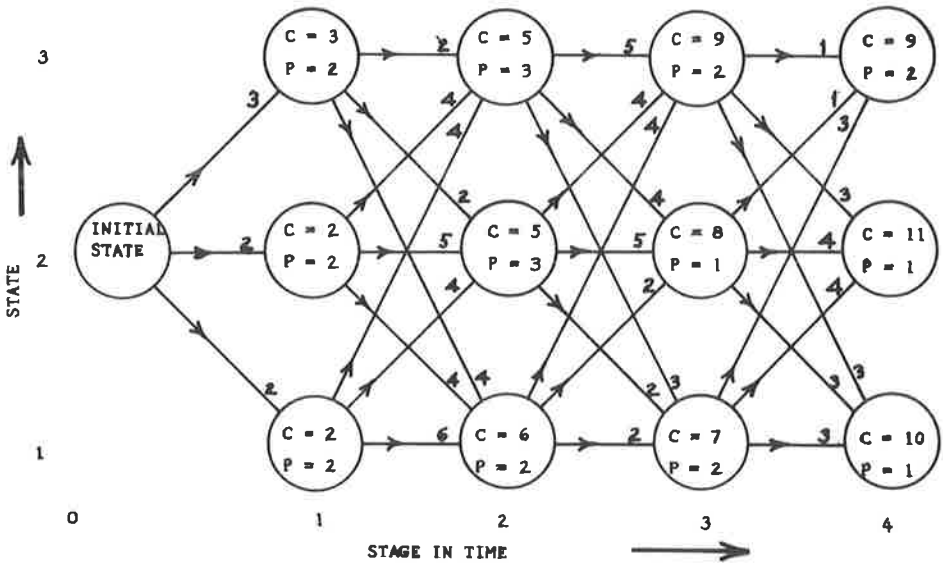

Figure 8. A simple numerical example of dynamic programming: numbers above lines represent state transition costs; numbers in circles represent cumulative least cost values C and best path values P.

Starting with stage 1, the cost, C, indicated within each circle at stage 1 is simply given by the number along the line from the initial state; the least-costs path, P, is 2 in every case since this is the only previous state available.

At stage 2, the cost to state 1 is the least of path 1 (cost = 2 + 6 = 8), path 2 (cost = 2 + 4 = 6), and path 3 (cost = 3 + 4 = 7). The state 1 circle is thus marked with the least cost and best path. Similarly the cost to state 2 is the least of path 1 (2 + 4), path 2 (2 + 5) and path 3 (3 + 2) and is thus marked as cost 5 and path 3, and similarly for stat 3 at stage 2.

Having completed the least cost and best path determination for stage 2 we can throw away all cost information about stage 1, since in determining costs at stage 3 we need know only the costs to the states at the previous stage. But the best path information must be retained.

At stage 3 we similarly obtain the least cost and corresponding path to reach each of the three states, and so with stage 4.

Now, assuming stage 4 is the terminal stage we could require that state 2 be the terminal state. We observed from our stored path information that we should have come there via state 1 at stage 3, and at state 1 at stage 3 we observe we should have come via state 2 at stage 2, thence via state 3 at stage 1, and finally back to the initial state. This is the optimal trajectory, the least-cost way of reaching state 2 at stage 4.

On the other hand we could have looked for the least-cost state at stage 4, which is found to be state 3. The optimal trajectory to reach this state is found to be entirely different: state 2 at stage 3, state 1 at stage 2, state 2 at stage 1, and back to the initia state.

All trajectories need not be different. The optimal trajectory to state 1 at stage 4 is seen to be the same as to state 2 except for the last step.

The reader should note that in determining the least-cost paths from any one stage to the next the number of trial comparisons was 9, the square of the number of states, and only the three best path values were stored. Both the total number of comparisons and the total number of stored path values was linear with the number of time stages traversed.