International Conference on Problems of Logistics, Management and Operation in the East-West Transport Corridor (PLMO), October 27-29, 2021
http://conf.cyber.az

Dynamic Models in Operational and Organizational Control Systems: Dynamic Programming Method

Aminagha Sadigov
Institute of Control System of Azerbijan
National Academy of Sciences
Baku, Azerbaijan
aminaga.sadigov@gmail.com
https://orcid.org/
0000-0002-0058-6895

Turan Isazadeh
Institute of Control System of Azerbijan
National Academy of Sciences
Baku, Azerbaijan
turan.isazadeh@yahoo.com

Lamiya Najafova
Institute of Control System of Azerbijan
National Academy of Sciences
Baku, Azerbaijan
najafova.lamiya@gmail.com

***Abstract*—Computational procedures for dynamic programming are described. The principle of optimality is considered on a specific example. The problem of terminal control is considered. The main advantages of computational dynamic programming procedures are formulated.**

Keywords*—*control systems, dynamic programming, optimality principle, optimization methods, combinatorial methods, operational control

I. INTRODUCTION

Optimization problems of operational and organizational control in modern systems, which, as a rule, have a complex hierarchical structure, are solved most often at the upper levels of the hierarchy. In this case, mainly static models of control objects are used. However, there are wide classes of problems when it is necessary to use dynamic models in the construction of operational and organizational control systems. First, these are control problems at the upper levels of the hierarchy, which are solved using dynamic models. A feature of these models is that the dynamic characteristics of objects are sufficiently inertial and the planning repetition interval exceeds the time required for calculating optimal programs on computers. Secondly, these are problems of control of dynamic objects at the lower levels of the hierarchy (technological processes and shop floor systems), which allow, mainly due to the time factor, using the principle of building operational and organizational control systems. Thirdly, these are problems of control of objects with a human in the control loop, making decisions on the choice of control modes for a dynamic object or corrective control actions. Besides, the theory of optimal control of dynamical systems is a classical and best studied area of optimal control, therefore, using the example of problems from this area, it is convenient to consider all the key features of optimization methods.

Despite the fact that the use of operational and organizational control systems allows avoiding the need to solve the problem of optimal system synthesis, the difficulties in implementing optimal programs using both analytical and numerical methods remain significant. The choice of the optimization method here is crucial.

The problems of solving two-point boundary value problems of taking into account control constraints and phase constraints can serve as an obstacle to the application of the classical calculus of variations, the maximum principle and dynamic programming (Bellman equations) [1], i.e., methods using the necessary optimality conditions. In another group of methods - direct optimization methods, which should include all gradient descent methods, various methods of purposeful enumeration (combinatorial methods), computational procedures for dynamic programming, etc., the account of control constraints and phase constraints is much less difficult. Among them, we should especially distinguish the computational procedures of dynamic programming, the presentation of which is discussed in this paper.

II. OPTIMALITY PRINCIPLE. BELLMAN EQUATION

Dynamic programming, which has two forms - analytical, associated with solving the Bellman equation, and numerical, implemented in the form of computational procedures of dynamic programming, is based on the optimality principle formulated by the American mathematician R. Bellman [1].

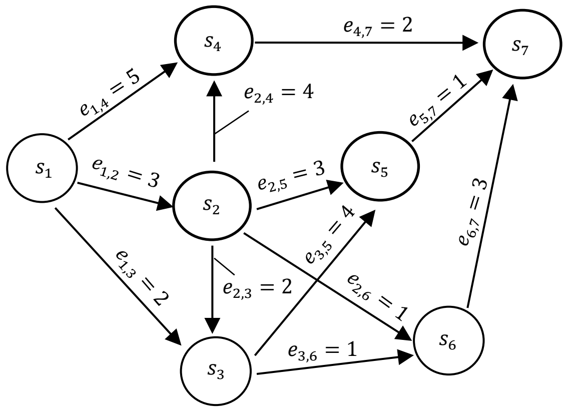

Fig. 1. The diagram explaining the optimality principle

Let us consider the optimality principle using a specific example. Suppose it is required to transfer the system from a given state s_1 to the final state s_7 (Fig.1) in such a way that during the transition a minimum cost (or time) is ensured:

$$E = \sum_{k,j} e_{k,j},$$

where $e_{k,j}$ is the costs related to the transition from the state s_k to the state s_j (or during the transition from s_k to s_j). The transition from the state s_k to the state s_j is carried out as a result of the action of the control $u_{k,j}$.

In other words, in an oriented scheme $\Gamma = \{S, U\}$, where S is the set of vertices and U is the set of edges, it is necessary to find a minimum length path between the vertices s_1 and s_7.

The solution to this problem can be obtained by complete enumeration of all paths from s_1 to s_7 [2], but if the number of possible states is large, complete enumeration may not be feasible. The dynamic programming method allows implementing the most economical enumeration procedure.

Let us introduce the function ω_k, which determines the minimum increment of the criterion E when going from the vertex s_k to the final vertex s_k. Successively determine $\omega_7, \omega_6, \omega_5, \ldots, \omega_1$. Determining

$$\omega_1 \triangleq \min_{\{k,j\}} \sum_{k,j} e_{k,j}\,,$$

and calculating ω_1 gives the sought-for result. Let us calculate

$$\omega_7 = 0;\ \omega_6 = e_{6,7} = 3;\ \omega_5 = e_{5,7} = 1;\ \omega_4 = e_{4,7} = 2.$$

From the vertex s_3, we can go to s_7 either through the vertex s_5, or through the vertex s_6. Since ω_5 and ω_6 are already known, we write

$$\omega_3 = min \begin{Bmatrix} e_{3,5} + \omega_5 = 4 + 1 = 5; \\ e_{3,6} + \omega_6 = 1 + 3 = 4 \end{Bmatrix} = 4.$$

Let us keep in mind the value of ω_3 and the vertex s_6, through which the path $s_3 \to s_6 \to s_7$ goes, and similarly calculate

$$\omega_2 = min \begin{Bmatrix} e_{2,3} + \omega_3 = 2 + 4 = 6; \\ e_{2,4} + \omega_4 = 4 + 2 = 6; \\ e_{2,5} + \omega_5 = 3 + 1 = 4 \end{Bmatrix} = 4,$$

let us also keep in mind $\omega_2 = 4$, as well as the path $s_2 \to s_5 \to s_7$.

To determine ω_1, we perform operations of the same type:

$$\omega_1 = min \begin{Bmatrix} e_{1,2} + \omega_2 = 3 + 4 = 7; \\ e_{1,3} + \omega_3 = 2 + 4 = 6; \\ e_{1,4} + \omega_4 = 5 + 2 = 7 \end{Bmatrix} = 6.$$

Thus, the path $s_1 \to s_3 \to s_6 \to s_7$ is optimal. The simple equations used to obtain it are called functional equations of dynamic programming. The effect of their application is that the general solution falls into several steps (stages) and the solution at each step is easier to obtain than from versions of a general solution. Besides, when building a general optimal solution, the results of the previous solutions are used as much as possible.

The dynamic programming method, as noted above, is based on the optimality principle. In the considered example, this principle was used in the sequential calculation of the functions ω_k, called Bellman functions. The functional equations

$$\omega_k = \min_{\{j\}} \left(e_{k,j} + \omega_j \right), \tag{1}$$

used above are applied in computational procedures. In an analytical solution of optimal control problems, the so-called Bellman equation is used, the derivation of which is convenient to demonstrate on the simplest problem of the classical calculus of variations — the Lagrange problem:

$$E = \int_0^T F(x,u)dt \to min; \tag{2}$$

$$\dot{x} = f(x,u); \tag{3}$$

$$x(0) = a. \tag{4}$$

Here, x is a state variable; u is the control action; a is the given boundary condition at the left end.

Since we need to find an analytical solution that is valid for any values of the initial state a and the integration interval $[0,T]$, we consider the parameters a and T as variables.

The Bellman function for problem (2)–(4) is defined by the expression

$$\omega(a,T) = \min_{u[0,T]} \int_0^T F(x,u)dt,$$

the notation $u[0,T]$ meaning that minimization (2) is carried out on the entire interval $[0,T]$. If we break the interval $[0,T]$ into two subintervals $[0,s]$ and $[s,T]$, then the expression for $\omega[a,T]$ can be represented as follows:

$$\omega(a,T) = \min_{u[0,s]} \min_{u[s,T]} \left[\int_0^s F(x,u)dt + \int_0^T F(x,u)dt \right].$$

or if we write it as follows:

$$\omega(a,T) = \min_{u[0,s]} \left[\int_0^s F(x,u)dt + \min_{u[s,T]} \int_0^T F(x,u)dt \right] \tag{5}$$

and carry out minimization on the interval $[0,s]$, using the entire totality of optimal trajectories found on the interval $[s,T]$. These trajectories will have initial states $a(s)$. Thus, equation (5) corresponds to the optimality principle.

By definition of $\omega[a,T]$, the second term on the right-hand side of equation (5) is the function of the initial state $a(s)$ and the interval $[s,T]$ with the duration $T - s$. Therefore,

$$\min_{u[s,T]} \int_s^T F(x,u)dt \triangleq \omega(a(s), T - s),$$

will take on the form

$$\omega(a,T) = \min_{u[0,s]}\left[\int_0^s F(x,u)dt + \omega(a(s),T-s)\right]. \quad (6)$$

Equation (6) has the same structure as equation (1), and is also called the functional equation of dynamic programming.

If we now introduce the notation $u[0,s] = \vartheta$ and select the interval $[0,s]$ as sufficiently small, that is, such that it should be possible to carry out with a high degree of accuracy the approximation

$$\int_0^s F(x,u)dt \approx F(a,\vartheta)s;$$

$$\dot{x}|_{t=0} \approx \frac{a(s)-a}{s} = f(a,\vartheta),$$

then it follows from the last formula that

$$a(s) = a + s\cdot f(a,\vartheta),$$

and equation (6) is transformed, taking the form

$$\omega(a,T) = \min_{\vartheta}[F(a,\vartheta)s + \omega(a+s\cdot f(a,\vartheta),T-s)].$$

Let us expand $\omega(a+s\cdot f(a,\vartheta),T-s)$ as a Taylor series relative to the point with the coordinates a and T in the increments $s\cdot f(a,\vartheta)$ and s, respectively. As a result, for the terms of the first-order series, we obtain

$$\omega(a+s\cdot f(a,\vartheta),T-s) =$$
$$= \omega(a,T) + s\cdot f(a,\vartheta)\frac{\partial\omega(a,T)}{\partial a} - s\frac{\partial\omega(a,T)}{\partial T}.$$

Substituting this expression into the equation for $\omega(a,T)$, we get

$$\omega(a,T) = \min_{\vartheta}[F(a,\vartheta)\cdot s + \omega(a,T) +$$
$$+s\cdot f(a,\vartheta)\frac{\partial\omega(a,T)}{\partial a} - s\frac{\partial\omega(a,T)}{\partial T}\bigg].$$

Since $\min_{\vartheta}\omega(a,T) = \omega(a,T)$, and s is a small but finite quantity, the last equation can be transformed as

$$0 = \min_{\vartheta}\left[F(a,\vartheta) + f(a,\vartheta)\frac{\partial\omega(a,T)}{\partial a} - \frac{\partial\omega(a,T)}{\partial T}\right] \quad (7)$$

This nonlinear partial differential equation is called the Bellman equation. It determines the necessary and sufficient conditions for the existence of an extremum for the original problem. It corresponds to two equations commonly used in calculations:

$$0 = F(a,\vartheta) + f(a,\vartheta)\frac{\partial\omega(a,T)}{\partial a} - \frac{\partial\omega(a,T)}{\partial T}; \quad (8)$$

$$0 = \frac{\partial F(a,\vartheta)}{\partial\vartheta} + \frac{\partial f(a,\vartheta)}{\partial\vartheta}\frac{\partial\omega(a,T)}{\partial a}. \quad (9)$$

Equation (8) was obtained by substituting the optimal control ϑ in (7), equation (9) – by differentiating (8) with respect to ϑ.

The physical meaning of the Bellman equation can be explained as follows. At the initial time instant, the control $u[0,s]$ is determined for a known value $x(0) = a$, which is equivalent to determining $\frac{dx}{dt}$. It is obvious from the fact that selecting $u[0,s]$ gives the value

$$x(s) = a + s\cdot f(a,u[0,s]).$$

At $s\to 0$, the value of f increasingly approaches the value $\frac{dx}{dt}$, since

$$f(a,u[0,s]) \approx \frac{a(s)-a}{s} \approx \frac{dx}{dt}$$

This is also the case for other time instants over the entire interval $[0,T]$. Therefore, when implementing the method of dynamic programming, the extremum is defined as the envelope of the family of tangents to the optimal trajectory. In the classical calculus of variations, the extremum is found as a geometrical locus of points that form the desired trajectory.

The Bellman equation can take a different form depending on the mathematical form of the optimization problem being solved. For example, with the minimization

$$E = \int_0^\infty F(x,u)\,dt$$

and conditions (7) and (4), we obtain

$$0 = \min_{\vartheta}\left[F(a,\vartheta) + f(a,\vartheta)\frac{\partial\omega\ (a)}{\partial a}\right].$$

Although equations of the type of (8) following from Bellman equation (7) were known in the calculus of variations as the Hamilton–Jacobi equations, the Bellman equation is a generalization of the latter, extending to the case of the presence of control constraints. In this case, it takes the form

$$0 = \inf_{\vartheta}\left[F(a,\vartheta) + f(a,\vartheta)\frac{\partial\omega\ (a,T)}{\partial a} - \frac{\partial\omega\ (a,T)}{\partial T}\right].$$

where *inf* is the exact lower bound of the backeted expression.

The Bellman equation is widely used to solve relatively simple problems of optimal control theory, in particular, problems of analytical design of controllers. For this class of problems, a linear description of control systems and a quadratic functional are used [3], which makes it quite simple to find a solution. At the same time, the Bellman equation is a nonlinear partial differential equation, and solving equations of this type is not a trivial task. From the theory of partial differential equations, we know the method of characteristics, which turns out to be the most acceptable. An example of using the method of characteristics is given in [4]. The applicability of the Bellman equation is also limited by the requirement for the existence of the derivative $\frac{\partial\omega}{\partial a}$, which is not always fulfilled. Of course, it would be possible to solve the Bellman equation by numerical methods, but that will involve no fewer difficulties in the implementation of computational dynamic programming schemes based on the solution of the functional equation, and not the Bellman equation.

III. THE FIRST COMPUTATIONAL PROCEDURE OF DYNAMIC PROGRAMMING

Earlier, a functional dynamic programming equation (6) was obtained for problem (2)-(4).

Suppose $[0, s] = \Delta$. Then equation (6) can be written as

$$\omega_0\,(x_0) = \min_{u_0}[F\,(x_0, u_0)\,\Delta + \omega_1\,(x_1)]. \qquad (10)$$

Here, the indices 0 and 1 denote the variables for the time instants $t = 0$ and $t = \Delta$, respectively; the arguments T and T have been omitted, being replaced with the indices 0 and 1. The physical meaning of equations (6) and (10) is the same, $\omega_0\,(x_0)$ is the optimal value of the criterion during the transition of the system from the state x_0 to the state x_N.

It is clear from (10) that to calculate $\omega_0\,(x_0) = \min E\,(x, u)$ we need to know $\omega_1\,(x_1)$. Obviously, $\omega_1\,(x_1)$ can be calculated similar to $\omega_0\,(x_0)$, but that requires knowing $\omega_2\,(x_2)$. To calculate $\omega_2\,(x_2)$, we need to know $\omega_3\,(x_3)$, etc., up to $\omega_N\,(x_N)$. Therefore, the search for a solution must be carried out successively step by step, completely similar to how it was done when selecting the optimal path, but taking discrete time values as the step $N\Delta, (N-1)\,\Delta, (N-2)\,\Delta, \ldots, \Delta, 0$. For any $1 \le k \le N$, functional equations of the type of (10), in which only the indexing of the variables changes, will hold true:

$$\omega_k\,(x_k) = \min_{u_k}[F\,(x_k, u_k)\,\Delta + \omega_{k+1}\,(x_{k+1})]. \qquad (11)$$

Obviously, at $t = N\Delta$, since the trajectory does not get an increment, $\omega_N \equiv 0$ for all x_N. The first step is to calculate $\omega_{N-1}\,(x_{N-1})$.

We enumerate sequentially all possible states from $x_{N-1} = M_x^-$ and to $x_{N-1} = M_x^+$ with the step δ, and for each of them, we enumerate all quantized values of the control action $M_u^-, M_u^- + \nu, \ldots, M_u^+ - \nu,\ M_u^+$. For each value of u_{N-1} at the given x_{N-1}, $F\,(x_{N-1}, u_{N-1})\Delta$ is calculated and the smallest one is selected, which corresponds to the solution of the equation

$$\omega_{N-1}\,(x_{N-1}) = \min_{u_{N-1}} F\,(x_{N-1}, u_{N-1})\Delta.$$

Thus, analyzing the state $x_{N-1} = \delta$ and five possible trajectories, each of which has its own corresponding value u_{N-1}, we should choose the one that corresponds to the minimum increment of the functional.

Let us proceed to the analysis of the second step, recording the optimal controls u_{N-1} for all possible states x_{N-1}. Let us set a specific value of x_{N-2}, e.g., $x_{N-2} = 0$, and enumerate all discrete values of u_{N-2}. As a result of the action u_{N-2}, the system will go from the state $x_{N-2} = 0$ to some state $x_{N-1} = x_{N-2} + \Delta f\,(x_{N-2}, u_{N-2})$, and the criterion will receive the increment $F\,(x_{N-2}, u_{N-2})\,\Delta$. . Optimal trajectories from any states x_{N-1} to the states x_N are already known. Therefore, the total trajectory of the transition from a given state x_{N-2} to x_N, taking into account the analyzed value of u_{N-2}, will be estimated by the expression $F\,(x_{N-2}, u_{N-2})\,\Delta + \omega_{N-1}(x_{N-1})$.

Enumerating all admissible values of u_{N-2} for the specified x_{N-2}, calculating $F\,(x_{N-2}, u_{N-2})\,\Delta$ and x_{N-1}, and also selecting the address of x_{N-1} the corresponding value of $\omega_{N-1}(x_{N-1})$, we can find

$$\omega_{N-2}(x_{N-2}) = \min_{u_{N-2}}[F\,(x_{N-2}, u_{N-2})\,\Delta + \omega_{N-1}\,(x_{N-1})].$$

Similar calculations are performed for all possible values of x_{N-2}.

These exact calculations should be carried out for all the remaining steps. The process ends with the calculation of $\omega_0(x_0)$ and u_0.

If $\vec{x}$ is an n-dimensional vector, and $\vec{u}$ is an m-dimensional vector, it is necessary to solve the problem

$$E\,(\vec{x}, \vec{u}) = \int_0^T F\,(\vec{x}, \vec{u})\,dt;$$

$$\dot{\vec{x}} = \vec{f}\,(\vec{x}, \vec{u});$$

$$\vec{x}(0) = \vec{a};$$

$$\vec{x}\,(T) = \vec{b} \text{ or not fixed;}$$

$$\vec{x}\,(t) \in E_x;$$

$$\vec{u}\,(t) \in E_u,$$

which is reduced to a mathematical programming problem of the form

$$E\,(\vec{x}_k, \vec{u}_k) = \sum_{k=0}^{N-1} F\,(\vec{x}_k, \vec{u}_k)\,\Delta;$$

$$\vec{x}_{k+1} = \vec{x}_k + \Delta\dot{\vec{f}}\,(\vec{x}_k, \vec{u}_k);$$

$$\vec{x}_0 = \vec{a};$$

$$\vec{x}_N = \vec{b} \text{ or not fixed;}$$

$$\vec{x}_k \in E_x, \qquad k = \overline{0, N};$$

$$\vec{u}_k \in E_u, \qquad k = \overline{0, N},$$

where k is the number of the quantization step of the variable t, then the structure of functional equation (11) remains the same:

$$\omega_k\,(\vec{x}_k) = \min_{\vec{u}_k}[F\,(\vec{x}_k, \vec{u}_k)\,\Delta + \omega_{k+1}\,(\vec{x}_{k+1})].$$

However, since each $\vec{x}_k$ is now a vector with the coordinates $x_1, x_2, \ldots, x_n$, and $\vec{u}_k$ is a vector with the coordinates $u_1, u_2, \ldots, u_m$, the amount of computation increases significantly.

The above computational procedure of dynamic programming has two shortcomings.

1. Let the state $x_k == M_x^+$ be analyzed for some k-th step. Then, when calculating x_{k+1} from the formula $x_{k+1} = M_x^+ + \Delta f\,(M_x^+, u_k)$, if $f\,(M_x^+, u_k) > 0$, the state $x_{k+1} > M_x^+$. This case is called the expanding variable grid. Computer programs that implement this dynamic programming procedure should include special operations that either really expand the variable grid or stop the computation at $x_{k+1} > M_x^+$ and $x_{k+1} < M_x^-$.

2. An assumption was made earlier that the quantization of the variables u and x was carried out in such a way that discrete values of the control u correspond to discrete values that coincide with the quantized $M_x^-, M_x^- + \delta, \ldots, M_x^+ - \delta, M_x^+$. However, in most cases, this assumption is not fulfilled. Therefore, it is necessary to use interpolation formulas to select such values of the control action that ensure that the calculated values of the state variables coincide with their quantized values with a specified degree of accuracy.

The computational procedure of dynamic programming, in which the second method of reducing optimal control problems to mathematical programming problems is used, does not have these two shortcomings.

IV. THE SECOND COMPUTATIONAL PROCEDURE OF DYNAMIC PROGRAMMING

Let us use the second method of reducing problems of optimal control to problems of mathematical programming in the version when the control $u_k = const$ on each of the intervals $[t_k, t_{k+1}]$ and can be calculated from difference equation for the known values of x_k and x_{k+1}. So, original problem is reduced to the following one:

$$E\,(x_k, u_k) = \sum_{k=0}^{N-1} F\,(x_k, u_k)\,\Delta;$$

$$x_{k+1} = x_k + \Delta \cdot f\,(x_k, u_k);$$

$$x_0 = a;$$

$$x_k \in [M_x^-, M_x^- + \delta, M_x^- + 2\delta, \ldots, M_x^+ - \delta, M_x^+];$$

$$M_u^- \le u_k \le M_u^+.$$

Let $\widehat{\omega}_k\,(x_k)$ denote the optimal value of the criterion $E\,(x_k, u_k)$ associated with a change in x from the value x_k at the instant $t = t_k = k\Delta$ to some arbitrary (since the right end is not fixed) value x_N at the instant $t = T = N\Delta$. Obviously, $\widehat{\omega}_N\,(x_N) = 0$ for all x_N, and $\widehat{\omega}_0 = \min E\,(x_k, u_k)$.

In accordance with the optimality principle, we successively determine

$$\widehat{\omega}_{N-1}\,(x_{N-1}), \widehat{\omega}_{N-2}\,(x_{N-2}), \ldots, \widehat{\omega}_2\,(x_2), \widehat{\omega}_1\,(x_1), \widehat{\omega}_0\,(x_0).$$

In the first step, $\widehat{\omega}_{N-1}\,(x_{N-1})$ is calculated. To do this, it is necessary to enumerate all the values of x_{N-1} and for each of them successively analyze all admissible values of x_N, calculating the corresponding u_{N-1}, increments of the criterion $F\,(x_{N-1}, u_{N-1}\,(x_N)\,)\Delta$ and selecting the smallest of them. Formally, this process can be presented by the formula

$$\widehat{\omega}_{N-1}\,(x_{N-1}) = \min_{(x_N)}[F\,(x_{N-1}, u_{N-1}(x_N))\,]\,\Delta;$$

$$x_N = x_{N-1} + \Delta \cdot f(x_{N-1}, u_{N-1}).$$

In the second step, for each possible state x_{N-2}, we calculate:

a) for all possible values of x_{N-1}, the corresponding values of u_{N-2} from the formula

$$x_{N-1} = x_{N-2} + \Delta \cdot f\,(x_{N-2}, u_{N-2});$$

b) from the known values of x_{N-2} and u_{N-2}, the increments of the criterion $F\,(x_{N-2}, u_{N-2}\,(x_{N-1}))\Delta$;

c) at the address x_{N-1}, the values of $\widehat{\omega}_{N-1}\,(x_{N-1})$;

d) the minimum value of $\widehat{\omega}_{N-2}\,(x_{N-2})$ from the formula

$$\widehat{\omega}_{N-2}\,(x_{N-2}) = \min_{(x_{N-1})}\Big[F\,\big(x_{N-2}, u_{N-2}(x_{N-1})\Delta + \widehat{\omega}_{N-1}(x_{N-1})\big)\Big].$$

Similar operations are carried out in all other steps. Thus, for an arbitrary k-th step, the functional equation has the form

$$\widehat{\omega}_k\,(x_k) = \min_{(x_{k+1})}\Big[F\,\big(x_k, u_k(x_{k+1})\big)\Delta + \widehat{\omega}_{k+1}(x_{k+1})\big)\Big]. \qquad (12)$$

In the final step, we calculate

$$\omega_0(x_0) = \min_{(x_1)}\Big[F\,\big(x_0, u_0(x_1)\Delta + \widehat{\omega}_1(x_1)\big)\Big],$$

In the considered computational procedure of dynamic programming, the order of calculation of $\widehat{\omega}_k$ from $k = N$ is used, i.e., from $t = T$ to $k = t_0 = 0$.

Suppose $\widehat{\omega}_k\,(x_k)$ is the optimal value of the criterion obtained during the transition of the system from the arbitrary state x_0 (the boundary condition at the left end is considered unspecified) to the state x_k. The process of calculating $\widehat{\omega}_k$ is carried out starting from $\widehat{\omega}_1\,(x_1)$. For this purpose, for each of the possible states of x_1 all possible states of x_0 are enumerated and the corresponding value of u_0 is calculated from the formula $x_1 = x_0 + + \Delta f\,(x_0, u_0)$, and then

$$\widehat{\omega}_1\,(x_1) = \min_{x_0}[F\,(x_0, u_0)\Delta].$$

is found.

If the value $x_0 = a$ is specified, the calculation process in the first step is simplified. When carrying out the calculations in the second step, the relations $x_2 = x_1 + \Delta f\,(x_1, u_1)$ are used, from which u_1 is calculated using the specified x_2 and x_1. The functional equation for the two-step process will have the form

$$\widehat{\omega}_2(x_2) = \min_{x_1}[F(x_1, u_1)\Delta + \widehat{\omega}_1(x_1)],$$

the values of $\widehat{\omega}_1(x_1)$ are taken from the table filled in the first step.

In the general case, for any $0 \le k \le N$, we will have the equation

$$\widehat{\omega}_k(x_k) = \min_{x_{k-1}}[F(x_k, u_k)\Delta + \widehat{\omega}_{k-1}(x_{k-1})]. \quad (13)$$

In cases when $\vec{x}$ is a vector and $\vec{u}$ is a vector, functional equations (12) for the second computational procedure of dynamic programming take the form

$$\widehat{\omega}_k(\vec{x}_k) = \min_{\vec{x}_{k-1}}[F(\vec{x}_k, \vec{u}_k)\Delta + \widehat{\omega}_{k-1}(\vec{x}_{k-1})]; \; k = \overline{0, N}.$$

It can be seen from the above that when using the second computational procedure, we do not need to perform interpolation and take into account the possibilities of expanding the variable grid, unlike the first computational procedure. Nevertheless, it also has some shortcomings, the main of which may be the difficulty of solving the system of algebraic equations $\vec{x}_{k+1} = \vec{x}_k + \Delta f(\vec{x}_k, \vec{u}_k)$ with respect to $\vec{u}_k$, especially if the dimensions of the vectors $\vec{x}$ and $\vec{u}$ do not coincide.

V. TAKING INTO ACCOUNT CONTROL CONSTRAINTS, PHASE CONSTRAINTS, AND BOUNDARY CONDITIONS

Compared with other optimization methods, computational procedures of dynamic programming have the advantage that they make it easy to take into account all possible forms of constraints on control and phase coordinates, as well as boundary conditions at both ends of the trajectory.

Let us first consider the first computational procedure. We assumed earlier that $M_u^- \le u(t) \le M_u^+$. Following this constraint, when carrying out calculations in each step, it is necessary to consider only those quantum values of u_k that are limited by the interval $[M_u^-, M_u^+]$. To do this, in a program that implements the method on a computer, we introduce either the entire set of values $[M_u^-, M_u^- + \delta, M_u^- + 2\delta, \ldots, M_u^+ - \delta, M_u^+]$, or conditions prohibiting the use of those u_k, that lie outside the interval $[M_u^-, M_u^+]$. If this interval is not constant in time, then the values $M_u^-(k\Delta)$ and $M_u^+(k\Delta)$ ($k = \overline{0, N-1}$) are fixed and before checking whether u_k belongs to the interval $[M_u^-(k\Delta), M_u^+(k\Delta)]$, the corresponding values $M_u^-(k\Delta)$ and $M_u^+(k\Delta)$, refined at each step, are introduced into the formula for verification. Obviously, the introduction of these values will not significantly complicate the computation program, even if the dependences $M_u^-(k\Delta)$ and $M_u^+(k\Delta)$ cannot be specified by analytical expressions and it will be necessary to store their tabulated values in the computer memory in the form of a table.

Let us consider the process of taking into account phase constraints. When calculating $x_{k+1} = x_k + \Delta f(x_k, u_k)$ for given x_k and u_k, it is necessary to check whether the calculated x_{k+1} belongs to the interval $[M_x^-(k\Delta), M_x^+(k\Delta)]$.

Let us now consider the process of solving a two-point boundary value problem. The presence of the boundary condition $x_N = b$ along with the condition $x_0 = a$ leads to a change in the calculations of only ω_{N-1} in the computational procedure of dynamic programming, and this change is related not to the complication, but to the simplification of the calculations. Indeed, if the following formula was used in the problem with two variable end-points

$$\omega_{N-1}(x_{N-1}) = \min_{u_{N-1}} F(x_{N-1}, u_{N-1})\Delta,$$

then for a fixed $x_N = b$ for each possible state x_{N-1} there will be a unique control that transfers the system from the state x_{N-1} to $x_N = b$. Thus, when calculating $\omega_{N-1}(x_{N-1})$, it is no longer necessary to choose the best of the controls, but to find the only control that transfers the system from x_{N-1} to $x_N = b$. Going through all possible values of u_{N-1}, we check the correspondence of the obtained x_N to the value $x_N = b$ and calculate $\omega_{N-1}(x_{N-1}) = F(x_{N-1}, u_{N-1})\Delta$. The condition $x_N = b$ has no effect on the further course of calculations.

Let us turn to the second computational procedure. By enumeration the values for the given value of x_k, the values of u_k are calculated here. Obviously, to take into account the control constraints for each calculated value of u_k, it is necessary to check the condition $u_k \in [M_u^-(k\Delta), M_u^+(k\Delta)]$, and, if it is not satisfied, exclude this value of u_k from consideration.

When selecting the values of x_{k+1} to calculate u_k for a given x_k, it is necessary to check the conditions $x_k \in [M_x^-(k\Delta), M_x^+(k\Delta)]$ as well, and once this condition is violated, proceed to the consideration of the new value of x_k (it is assumed that the enumeration of x_{k+1} is performed successively starting from $M_x^-(k\Delta)$ or in the opposite direction).

Taking into account the boundary condition $x_N = b$ in the second computational procedure turns out to be even simpler than in the first. In the first step, the value of u_{N-1} is calculated for each of the valid x_{N-1} from the formula

$$b = x_{N-1} + \Delta f(x_{N-1}, u_{N-1}),$$

and then $\widehat{\omega}_{N-1}(x_{N-1}) = F(x_{N-1}, u_{N-1})\Delta$. The same order of calculation can be used in the first computational procedure to determine $\omega_{N-1}(x_{N-1})$, as it allows foregoing the enumeration of all admissible controls u_{N-1}.

To illustrate the nature of the computational process implemented in dynamic programming, let us consider a simple problem, which, after being reduced to a mathematical programming problem, takes the form

$$E = (x_k, u_k) = \sum_{k=0}^{N-1}(x_k^2 + u_k^2)\Delta + \lambda(x_N - 2)^2 \rightarrow min;$$

$$x_{k+1} = x_k + \Delta u_k;$$

$$u_k \in E_u = [-2, -1, 0, 1, 2];$$

$$0 = M_x^- \le x_k \le M_x^+ = 8;$$

$$0 \leq x_N \leq 2.$$

We select quantization steps $\delta = 1$ and $\Delta = 1$ and use the second computational procedure of dynamic programming. For $\Delta = 1$ at $T = 10$, we will get $N = 10$.

Since the values of x_N are subject to a constraint and a fine $\lambda\,(x_N - 2)^2$ (specified $\lambda = 2.5$) is imposed for the non-compliance with the condition $x_N = 2$, then another row must be entered into the table of results to store the values of $\widehat{\omega}_N(0) = 2.5 \cdot 4 = 10, \widehat{\omega}_N(1) = 2.5, \widehat{\omega}_N(2) = 0$. The states $x_N > 2$ are inadmissible. We will write the results of the computations in Table 1.

In the first step, we fix the state $x_{N-1} = 0$ and start the enumeration of the states $x_N = 0, x_N = 1, x_N = 2$. For $x_N = 0$, we find the value $u_{N-1} = 0$ from the equation $x_N = x_{N-1} + \Delta\, u_{N-1}$. For $x_N = 1$, we get $u_{N-1} = 1$, and for $x_N = 2$, we determine $u_{N-1} = 2$. Now we calculate

$$\widehat{\omega}_{N-1}\,(x_{N-1}) = \min_{x_N}[(x_{N-1}^2 + u_{N-1}^2) + \widehat{\omega}_N\,(x_N)].$$

For $x_N = 0, x_N = 1, x_N = 2$, we write successively

$$\widehat{\omega}_{N-1}\,(0) = \min_{x_N}\left\{\begin{matrix}(0^2 + 0^2) + 10 = 10;\\ (0^2 + 1^2) + 2.5 = 3.5;\\ (0^2 + 2^2) + 0 = 4.\end{matrix}\right\}$$

Thus, for the state $x_{N-1} = 0$, the transition to the state $x_N = 1$ is optimal. Similar calculations are performed in all steps of the computation. For instance, in the second step for the state $x_{N-2} = x_8 = 2$, it is necessary to analyze the states $0 \leq x_9 \leq 4$. Successively considering $x_9 = 0, 1, 2, 3, 4$, we find the values $u_8 = -2, -1, 0, 1, 2$ from the equation $x_9 = x_8 + \Delta\, u_8$. For each of them, we need to calculate the increments of the criterion $(x_8^2 + u_8^2)\Delta$. Successively substituting $u_8 = -2, -1, 0, 1, 2$, we find $(x_8^2 + u_8^2)\Delta = 8, 5, 4, 5, 8$. These values will correspond to the states $x_9 = 0, 1, 2, 3, 4$, which are characterized by the values $\widehat{\omega}_9 = 3.5; 2, 4, 10, 20$. The values of $\widehat{\omega}_8\,(2)$ will be found from the formula

$$\widehat{\omega}_8\,(2) = \min_{x_N}\,[(2^2 + u_8^2) + \widehat{\omega}_9\,(x_9)] = \begin{bmatrix}8 + 3.5\\ 5 + 2\\ 4 + 4\\ 5 + 10\\ 8 + 20\end{bmatrix} = 7.$$

Therefore, the optimal transition from the state $x_8 = 2$ is into the state $x_9 = 1$ under the influence of the control $u_8 = 1$. By means of calculations of this type, we fill Table 1.

TABLE I.

x	0	1	2	3	4	5	6	7	8
$\widehat{\omega}_{10}$	10	2,5	0	×	×	×	×	×	×
$\widehat{\omega}_9$	3,5	2	4	10	20	×	×	×	×
x_{10}	1	2	2	2	2	×	×	×	×
$\widehat{\omega}_8$	3	3	7	14	24	39	60	×	×
x_9	1	1	1	2	2	3	4	×	×
$\widehat{\omega}_7$	3	4	8	16	27	43	64	92	128
x_8	0	1	1	1	2	3	4	5	6
$\widehat{\omega}_6$	3	5	9	17	28	45	67	96	132
x_7	0	0 or 1	1	1	2	3	4	5	6
$\widehat{\omega}_5$	3	5	10	18	29	46	68	98	135
x_6	0	0	1	1	2	3	4	5	6
$\widehat{\omega}_4$	3	5	10	18	30	47	69	99	136
x_5	0	0	1	1	2	3	4	5	6
$\widehat{\omega}_3$	3	5	10	18	30	47	70	100	137
x_4	0	0	1	1	2	3	4	5	6
$\widehat{\omega}_2$	3	5	10	18	30	47	70	100	138
x_3	0	0	1	1	2	3	4	5	6
$\widehat{\omega}_1$	3	5	10	18	30	47	70	100	138
x_2	0	0	1	1	2	3	4	5	6
$\widehat{\omega}_0$	3	5	10	18	30	47	70	100	138
x_1	0	0	1	1	2	3	4	5	6

VI. THE PROBLEM OF LOGICAL-DYNAMIC SYSTEMS

The need to solve increasingly complex problems of control of real objects not only stimulates the development of traditional methods of control, but also gives rise to completely new methods of solving them with the involvement of an unconventional mathematical apparatus. One of the classes of such problems is the class of control problems for logical-dynamic systems. The main quality that characterizes a logical-dynamic system is the presence of a finite set of states and a jump-like transition from one state to another [5, 6, 7]. At the same time, outside the jump-like (logical) transitions, the system functions as a dynamic one. In other words, a logical-dynamic system should be considered such a system, the continuous dynamic nature of operation of which is significantly influenced by logical conditions leading to a change in its dynamic characteristics.

Let us consider the problem of logical-dynamic systems in the light of the construction of operational and organizational control systems. Following [7, 8], we write the equation of a logical-dynamical system in the form

$$\dot{\vec{x}}(t) = AL^A\vec{x}\,(t) + BL^B\vec{u}\,(t),$$

where $\vec{x}\,(t)$ and $\vec{u}\,(t)$ are, as before, phase vector function and control vector function, respectively; AL^A and BL^B are logical-operator matrices, the values of the elements of which can change abruptly depending on the logical conditions.

This equation can be represented in a more general form:

$$\dot{\vec{x}}(t) = \vec{f}(\vec{x}\,(t), \vec{u}\,(t), L(\vec{x}, \vec{u}\,)), \tag{14}$$

where $L(\vec{x}, \vec{u}\,)$ is a totality of logical conditions that change the equation parameters depending on $\vec{x}$ and $\vec{u}$ or transfer the system in a jump-like manner from one state to another.

When using the principle of operational and organizational control, as noted earlier, there is no need to solve the synthesis problem, but the problem of obtaining optimal program controls remains. We will use the second method of reducing problems of optimal control to problems of mathematical programming. For this purpose, we introduce the quantization step of the variable t equal to the constant Δ, and select in the space of variables $x_1, x_2, \ldots, x_n, t$ those points where the action of logical conditions leads to an abrupt change of states or a change in the parameters of equation (14). If there were no logical conditions, it would be possible to introduce an elementary operation, but the presence of logical conditions forces us to write an elementary operation in the form

$$B\ (x_k, x_{k+1}) = \begin{Bmatrix} \gamma_{k,k+1}, & u_k \\ \hat{L}\ (x_k, x_{k+1}) \end{Bmatrix}, \qquad (15)$$

i.e., along with the existence of a continuous trajectory $\gamma_{k,k+1}$ we suppose the possibility of a jump-like transition from x_k to x_{k+1} along the trajectory $\hat{L}\ (x_k, x_{k+1})$ under the influence of the logical condition $L\ (x_k, x_{k+1})$. When using the second computational procedure of dynamic programming, the transition to (15) requires additional saving all conditions $L\ (x_k, x_{k+1})$ in the computer memory and calculating $\widehat{\omega}_k\ (x_k)$ with account for these conditions.

Thus, the problem of accounting for logical conditions in operational and organizational control systems is associated only with an increase in the amount of computation and does not cause any fundamental problems. It should be added that, as the calculation of simple problems has shown, even the increase in the amount of computation turns out to be insignificant.

VII. AVERAGING CONTROL AND TERMINAL CONTROL

The optimal control problems considered previously far belong to the class of so-called averaging control problems[9, 10, 11, 12]. An integral functional was used as a criterion in them, and the choice of control at any moment of time affected this functional. At the same time, there is a large class of problems in the practice of optimal control, in which the behavior of an object on the interval $[0, T]$ is not of interest, but the result achieved at the finite time instant is important. This result depends on the final state $\vec{x}\ (T)$ and is determined by the criterion $\Phi\ (\vec{x}\ (T))$. Problems of this type are called terminal control problems. In the classical calculus of variations, these problems are called Mayer problems, and the criterion $\Phi\ (\vec{x}\ (T))$ is called the Mayer criterion. (Problems of reaching the extremum of a mixed-type criterion

$$E\ (\vec{x}, \vec{u}) = \int_{t_0}^{T} F\ (\vec{x}, \vec{u})\ dt + \Phi\left(\vec{x}\ (T)\right)$$

in the classical calculus of variations are called Bolza problems).

The Lagrange problem

$$E\ (\vec{x}, \vec{u}) = \int_{t_0}^{T} F\ (\vec{x}, \vec{u})\ dt\ \rightarrow min;$$

$$\dot{\vec{x}}(t) = \vec{f}(\vec{x}, \vec{u});$$

$$\vec{x}\ (t_0) = \vec{x}^0$$

can always be reduced to a Mayer problem by introducing an additional coordinate x_{n+1} that satisfies the equation

$$\dot{x}_{n+1} = f_{m+1} = F\ (\vec{x}, \vec{u}), \quad x_{n+1}(t_0) = 0.$$

Using the augmented vector $\vec{\bar{x}} = |\vec{x}, x_{n+1}|$ instead of $\vec{x}$ and the vector $\vec{\bar{f}}$ instead of $\vec{f}$, we arrive at the following Mayer problem:

$$\Phi\ (\vec{x}, \vec{u}) = x_{n+1}\ (T)\ \rightarrow min;$$

$$\dot{\bar{x}} = \bar{f}\ (\bar{x}, \vec{u});$$

$$\bar{x}\ (t_0) = \bar{x}_0.$$

However, it is possible to reduce a Mayer problem to a Lagrange problem only in rare special cases, and therefore it becomes necessary to develop specific methods for solving Mayer problems. In this respect, let us consider the possibilities of computational procedures of dynamic programming.

So, we have the following terminal control problem:

$$\left.\begin{array}{c} E\ (x, u) = \Phi\left(x\ (T)\right) \rightarrow min \\ \dot{x} = f\ (x, u); \\ x_0 = a; \\ u(t) \in E_u. \end{array}\right\} \qquad (16)$$

Using the first method of reducing problems of optimal control to problems of mathematical programming, we proceed from problem (59) to problem

$$E\ (x_k, u_k) = \Phi\ (x_N);$$

$$x_{k+1} = x_k + \Delta f\ (x_k, u_k),\ \ k = \overline{0, N-1};\ x_0 = a; \qquad (17)$$

$$u_k \in E_u.$$

Let us introduce into consideration the function $\omega_k\ (x_k)$, which expresses the optimal value of the criterion $\Phi\ (x_N)$, which can be achieved from the state x_k, and calculate in advance the values $\omega_N\ (x_N) = \Phi\ (x_N)$ for all possible values of x_N. At the same time, we had to set the range of possible values of x again, limiting it to the interval $[M_x^-, M_x^+]$. Of course, among the values $x \in [M_x^-, M_x^+]$, it is easy to find the one that guarantees $\min \Phi\ (x_N)$.. But the question is whether this value is reachable from the state $x_0 = a$,, and if it is, then in what way, and if it is not, then what will be the optimal reachable state and how to arrive at it.

VIII. THE MAIN ADVANTAGES AND SHORTCOMINGS OF COMPUTATIONAL PROCEDURES OF DYNAMIC PROGRAMMING. ASSESSMENT OF COMPUTATIONAL DIFFICULTIES

Let us formulate the key advantages of computational procedures of dynamic programming using the above results.

1. Computational procedures of dynamic programming have some properties of an "analytical" nature. First, it is possible to recover the optimal trajectory from tables of optimal solutions for any initial conditions in the interval $[M_x^-, M_x^+]$. Second, from the same tables it is possible to obtain the optimal solution for any time interval $[0, T']$ at $T' \leq T$. To do this, it is necessary to start restoring the

optimal strategy, not from $x_0 = a$, but from $x_l = a$, where $l = (T - T') / \Delta$.

For instance, suppose the problem has $T = 7$ and $x(0) = = 1$. Then $l = 3$. In this case, we need to start restoring the optimal trajectory from the seventh row from the bottom of Table 3, selecting the value $x_4 = 0$ at the address $x = 1$. Then we will determine the state $x_5 = 0$, and so on.

2. The presence of various kinds of constraints does not complicate, but facilitates the solution process.

Suppose, for instance, that the control constraints have the form $|u_k| \leq \gamma$. Then, if the computational scheme for the first method of reducing the optimal control problem to a mathematical programming problem is used, the smaller γ, the smaller the number of running values of u_k and, therefore, the faster the computation process.

If phase constraints $x_k \in [M_x^-(k\Delta), M_x^+(k\Delta)]$, are specified, then their fulfillment for each u_k is verified. The stricter the phase constraints, the fewer cells in the table, the shorter the computation process.

The presence of boundary conditions at the right end practically does not change the computational procedure of the dynamic programming method. The exception is the first step, where ω_{N-1} or $\widehat{\omega}_{N-1}$, are determined, but the calculations for the two-point boundary value problem there turn out to be even simpler than in the problem with two variable endpoints.

3. Computational procedures of dynamic programming are well adapted for taking into account various kinds of constraints, not only given in the simplest form (e.g., in the form of inequalities), but also for constraints of a more complex form, in particular, those including logical conditions.

4. Using computational procedures of dynamic programming, it is possible to solve averaging optimization problems (with interval functional), but also terminal optimization problems, with the computational difficulties of the solution not increasing.

5. Both computational procedures can be used to solve optimization problems with practically any criterion in terms of form, both linear and nonlinear. Moreover, the criterion may not even be specified analytically, but only formulated in the form of a table for the discrete values of the variables.

6. Finally, using computational procedures of dynamic programming, it turns out to be possible to investigate both deterministic and stochastic processes. Taking into account the effects of random parameters does not affect the essence of the method, but only leads to the need to analyze a much larger number of control options.

It follows from the above that it is difficult to find a problem that could not be solved in principle using computational procedures of dynamic programming. However, the theoretic possibility in many practical cases is not supported by the practical possibility. This is explained by a very serious drawback of the dynamic programming method, which is that in order to obtain an optimal solution, it is sometimes necessary to analyze so many control options that the most modern computers will be incapable of doing it. The main difficulty in the practical implementation of computational procedures of dynamic programming is overcoming the dimensionality problem, or, in the figurative expression of R. Bellman, "the curse of dimensionality". Usually, in dynamic programming, the dimensionality is understood as the number of solution options required for analysis during all N calculation steps. The dimensionality of the problem is closely related to the dimensionality of the phase vector, the number of quantization steps of variables and time, i.e., proportional to the number of nodes in the grid of variables determined as

$$\left(\prod_{j=1}^{n} M_{x_j}\right) N = M_{\vec{x}} N, \tag{18}$$

where $M_{\vec{x}}$ is the number of levels of quantization of the variable x_j carried out with the step δ_j;

$$M_{x_j} = \left[\frac{M_{x_j}^+ - M_{x_j}^-}{\delta_j}\right] + 1, \tag{19}$$

the square brackets denoting the integer part of the number enclosed in them.

Let us characterize quantitatively the labor intensity of calculations for the first computational procedure of dynamic programming.

The number of possible control variations for each of the states $\vec{x}_k$ is determined from the expression

$$M_{\vec{u}} = \prod_{i=1}^{m} M_{u_i}.$$

Since at one step the total number of possible states $\vec{x}_k$ is equal to the number of nodes of the grid of variables at this step, i.e., $M_{\vec{x}}$, the total number of local control options to be investigated at each step is equal to $M_{\vec{u}}\, M_{\vec{x}}$, and at all N steps

$$W^1 = M_{\vec{u}}\, M_{\vec{x}} N. \tag{20}$$

Taking into account (20), we can write a formula for an approximate, in this case, the upper-bound, estimate of the time of solution of the optimization problem using the first computational procedure

$$W_t^1 = \alpha^1 W^1 = \alpha^1 M_{\vec{u}}\, M_{\vec{x}} N, \tag{21}$$

where α^1 is proportionality coefficient, taking into account that to calculate one local solution at each step it is necessary to:

1) take one of the possible options of the control vector$\vec{u}_k$;

2) calculate the value of $\vec{x}_{k+1}$;

3) determine the increment of the criterion caused by the transition from $\vec{x}_k$ to $\vec{x}_{k+1}$, i.e., $F(\vec{x}_k, \vec{u}_k)\,\Delta$;

4) take from the table the value of ω_k corresponding to the calculated $\vec{x}_{k+1}$;

5) add $F(\vec{x}_k, \vec{u}_k)\,\Delta$ and ω_k together, and then compare the result with the previous option and choose the smallest one.

The specific numeric value of the coefficient α^1 should be determined based on the analysis of the real calculation process for a specific computer, its operating system and the language in which the program is written.

An assessment of the amount of memory required to implement the dynamic programming procedure can be carried out by the formula

$$W_n^1 = (m+1)\, M_{\vec{x}} N. \qquad (22)$$

This is a lower-bound estimate that essentially determines only the size of the table for storing the values of $\omega_k\,(x_k)$ and the corresponding m components of the vector $\vec{u}_k$.

Let us compare the results obtained with those estimates of the computational difficulties that are characteristic of the complete enumeration method. For one fixed initial state, when using the complete enumeration method, an analysis of $M_{\vec{u}}$ options is required at the first step, an analysis of $(M_{\vec{u}})^2$ options is required at two solution steps, and an analysis of $(M_{\vec{u}})^N$ options is required at N steps. Since the total number of initial states is $M_{\vec{x}}$, the total number of solution options in the complete enumeration method is $M_{\vec{x}}\,(M_{\vec{u}})^N$. Now, similarly to [13], we can write a formula for determining the solution time:

$$W_t^{11} = \alpha^{11} M_{\vec{x}}\,(M_{\vec{u}})^N \qquad (23)$$

where α^{11}, just as α^1, is a proportionality coefficient characterizing the time of calculation of one solution by the complete enumeration method.

If we compare both methods in terms of the number of options, then the first computational procedure of the dynamic programming method turns out to be $(M_{\vec{u}})^{N-1}\,/\,N$ times more efficient than the complete enumeration method. Suppose, for instance, $m = 2; n = 2;\ M_{u_i} = M_{x_j} = 10;\ \forall i, \forall j; N = 10$. Then, assuming $\alpha^{\Pi} = 1$ in (23) and using estimate (20), we find that in the computational procedure of dynamic programming it is necessary to analyze 10^5 options of the solution, whereas in the case of the complete enumeration method $- 10^{22}$ options.

It should be noted that estimates (20), (21) were obtained under the assumption that all of the possible combinations of the parameters of the vector $\vec{u}$ are admissible, just like all possible combinations of $\vec{x}$. The presence of additional constraints on $\vec{u}$ and $\vec{x}$ leads to a decrease in estimates (20) and (21).

To assess the computational difficulties of the second computational procedure of dynamic programming, we take into account that the number of solution options required for the analysis for a given $\vec{x}_k$ will be $M_{\vec{x}}$, and for all possible x_k-th steps $- (M_{\vec{x}})^2$. As before, $M_{\vec{x}}$ is calculated from formulas (18) and (19). For all N steps of the analysis, the number of options to be calculated will be

$$W^{11} = (M_{\vec{x}})^2 N. \qquad (24)$$

Similarly to (21), one can introduce a coefficient α^{11}, which determines the time spent on calculating one control option, and obtain an upper-bound estimate for the calculation time

$$W_t^{11} = \alpha^{11} W^{11} = \alpha^{11} (M_{\vec{x}})^2 N. \qquad (25)$$

When implementing a computational procedure, to store a table with optimal results, we will need

$$W_n^{11} = (n+1)\, M_{\vec{x}} N. \qquad (26)$$

memory cells. Formula (26), like formula (22), gives a lower-bound value for the memory size.

So, in comparison with the method of complete enumeration, the computational procedures of dynamic programming have immense advantages, since they can significantly reduce the computational difficulties. However, the dimensionality problem remains the main obstacle to the widespread use of dynamic programming. It will be shown further by what means and to what extent the dimensionality problem can be overcome.

REFERENCES

[1] Richard Bellman, Dynamic Programming, Princeton University Press, 2010, 392 pp.

[2] A.B. Sadigov, Models and technologies for solving problems of management in emergency situations, Baku, "Elm", 2017, 372 p. (in Russian)

[3] A.V. Mattis, Optimal control of the movement of marine mobile complexes, Automation of management processes, 1 (23), 2011, pp. 88-92. (in Russian)

[4] T.G. Sotnikova, D.G. Gulakov, Dynamic programming method for tasks of modes optimization by decrease size of ore granules implementation, Bulletin of the Volodymyr Dahl East Ukrainian National University, No. 7 (224), 2015, pp. 91-95. (in Russian)

[5] A.S. Bortakovskiy, Optimal control of logic-dynamical systems, Dissertation for the degree of Doctor of Physical and Mathematical Sciences, HAC RF 05.13.01, 2010, 169 p. (in Russian)

[6] O.A. Slavin, A.V. Solovyov, An.V. Solovyov, Development of a methodology for creating a logical and mathematical model of traffic movement at the stage of creating a concept of a smart city, Proceedings of the ISA RAS, Volume 63, 3/2013, pp. 31-41. (in Russian)

[7] A.A. Kadyrov, A.A. Kadyrova, Structuring and graph modeling of logic-dynamic control systems, Bulletin of BSTU im. V.G. Shukhova, 2014, No. 1, pp. 185-188. (in Russian)

[8] A.B. Sadigov, Modeling of Appearance of Instability of Complex Systems. Reports of IV International conference «Problems of Cybernetics and Informatics», September 12-14, 2012, Baku, Azerbaijan, pp. 70-73.

[9] A.V. Lezhnyov, Dynamic programming in economic problems, Tutorial, Moscow, BINOM, 2006, 176 p. (in Russian)

[10] A.V. Brodsky, Automation of the solution of optimization problems in the design of aerospace technology, Automation of the solution of optimization problems in the design of aerospace technology, Electronic journal, "Proceedings of the MAI", Issue 71 (in Russian)

[11] D.A. Karpov, V.I. Struchenkov, Dynamic Programming as a Method of Spline Approximation in the CAD Systems of Linear Constructions, Russian Technological Journal, 2019, 7(3), pp. 77-88. (in Russian)

[12] Yaofei Ma, Xiaole Ma, Xiao Song, A Case Study on Air Combat Decision Using Approximated Dynamic Programming. Hindawi Publishing Corporation, Mathematical Problems in Engineering, Volume 2014, Article ID 183401, 10 pages.

[13] P.I. Kishteev, Design decisions making based on the principle of complexity, Dissertation for the degree of candidate of technical sciences, HAC RF, 05.13.01, 1983, 203 p. (in Russian)