

Chapter 2

Mathematical Basics of LTS Sustainable Development Dynamic Management

Abstract This chapter describes the basic concepts of mathematical basis for LTS sustainable development management: LTS development actions, LTS development state set, and LTS development process. Using these terms, mathematical model of development process was elaborated, and algorithmic aspects of development process management were analyzed. This chapter presents the concepts of *development process management recursion equation and optimal initial state*. The analysis of the model has shown that it is possible to calculate LTS optimal development plan considering only OIS set.

2.1 Principal Concepts

Development period of LTS may be represented as a multi-step D-process. To elaborate mathematical basics of dynamic management methods, we use dynamic programming created by Bellman [1, 2] as a starting point, with whose help optimal multi-step tasks solutions may be determined. Dynamic programming is based on the following optimum characteristics: technical system optimal operation depends only on the system D-state at the specific time moment but not on its previous D-plan until the specific moment. This kind of assumption may be applied if optimization criterion is expressed as a sum. Many LTS development management criteria are represented by sum, for example, the total discounted cost criterion.

This chapter deals with selecting dynamic programming recursive equation for LTS development optimization.

In the book, we apply graph and set theory concepts and methods for modeling LTS D-plan and development states (D-states) [5].

Graph theory investigates graph characteristics and related interconnections, as well as elaborates algorithms for its characteristics identification and sub-graphs determination. Initial systematic researches in this area belong to L. Euler. As an

independent subject, graph theory was established in 1936 when the world's first book on graph theory written by Hungarian mathematician D. König was published.

Graph is a mathematical model showing interconnections among objects. Graph is described with the following sets: nodes or vertex set V and edge set E . If all the edges are various, that is a unigraph, if edges are repeated—a multigraph. The significant characteristic is node incidence level. Graphs can be guided (see Fig. 2.1) and non-guided. Special graphs are loop, tree, path and planar graphs (see Fig. 2.1).

Set theory is the branch of mathematics that studies sets, which are collections of objects. Although any type of object can be collected into a set, set theory is applied most often to objects that are relevant to mathematics. Set theory, however, was founded by a single paper in 1874 by Georg Cantor: “On a Characteristic Property of All Real Algebraic Numbers”.

Set theory begins with a fundamental binary relation between an object o and a set A . If o is a member (or element) of A , we write $o \in A$. Since sets are objects, the membership relation can relate sets as well. A derived binary relation between two sets is the subset relation, also called set inclusion. If all the members of set A are also members of set B , then A is a subset of B , denoted $A \subseteq B$. For example, $\{1, 2\}$ is a subset of $\{1, 2, 3\}$, but $\{1, 4\}$ is not. From this definition, it is clear that a set is a subset of itself; in cases where one wishes to avoid this, the term proper subset is defined to exclude this possibility. Just as arithmetic features binary operations on numbers, set theory features binary operations on sets. These are union, intersection, complement, symmetric difference, Cartesian product, and power set.

The mathematical logic [4] as well as combinatorial [3, 6] concepts and methods are also used for system D-plan and D-states creation and analysis provided in the book.

LTS D-plan may be described with the following basic concepts.

LTS D-actions. Basic concept definition *D-actions* is a specific feature of LTS development optimization model; it is not applied in classical dynamic programming model where only concept definitions *D-state* and *optimization steps* are used. We consider that the application of *D-actions* provides opportunity to solve real systems development optimization tasks.

D-actions form development model. Development model is used to determine optimal plan with dynamic optimization model as well as by applying plans

Fig. 2.1 Illustration of graph theory basic characteristics: **a** Guided graph, **b** path, **c** loop, **d** tree

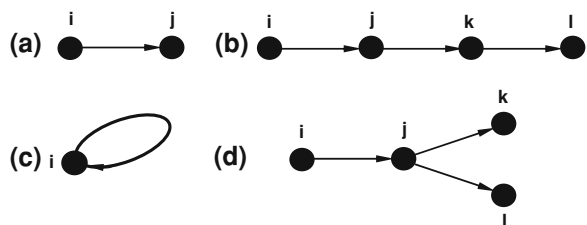
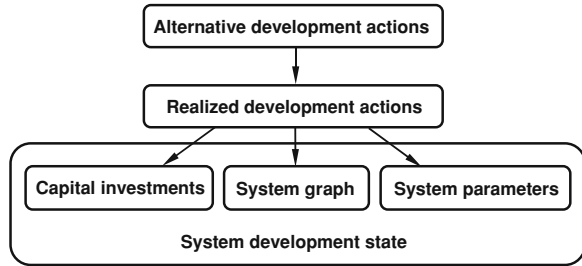


Fig. 2.2 D-states formation scheme



Fig. 2.3 Formation of information system calculation for D-state $e(t)$



estimation method. Important concept related to system economic analysis complex is possible (alternative) actions.

Let us define D-action. Generally, D-action reflects activity, new system object construction, as well as existing objects extension, reconstruction, and modernization.

D-action is assigned along with capital investments and relevant network elements (elements that are included or excluded from the estimation calculation scheme because of the D-action). The D-actions are assigned by user according to the specific task, experience, etc. Dynamic optimization of user software assigned D-actions is employed as “bricks” for creating D-plan and synthesizing optimal plan. D-actions are systems development dynamic optimization task options. All possible D-actions set constitute this task optimization area. “D-actions” may be interpreted as “purchases”. D-action is always interrelated with activity. Such variables differ from traditional approaches in many other optimization models where elements parameters are modeled: we are searching for “what to do” and “how to do”; elements parameters are created because of activities.

D-actions are structured as follows: elementary and compound. The compounds are formed from elementary actions, applying *logical conditions*: “one from” and “after”.

D-states formation scheme is depicted in Fig. 2.2, but information scheme of D-state $e(t)$ calculation is shown in Fig. 2.3.

Definition *D-action realization* denotes respective object construction, reconstruction or liquidation completion (commissioning or decommissioning), but not time period of object construction, reconstruction, or liquidation. It is assumed in the model that D-action is realized at the beginning of the first year of D-step. Constructions, reconstructions, or dismantling are finished within this year. This time interval duration is observed when the respective D-action costs (capital investments) are calculated. As soon as the D-action is realized, estimation calculation scheme is altered.

Possible realization period. The beginning of this period is usually determined by time length required for object construction, reconstruction, or liquidation. Usually, its completion is concurrent to estimation period completion, but under specific conditions, eventual realization period may be finished faster, if, for example, on plot where construction of new substation is envisaged, other civil works might be started prior to calculation period completion.

D-actions will be marked with $x(i)$, number of D-actions—with n .

LTS D-state. Technical system D-state in optimization task is specified by system graph, its elements parameters and other technical, economic, ecological, etc. parameters. Let us mark D-state with $e(j)$, where j is a sign or sign group, which determines D-state reciprocal dissimilarities. As a distinctive sign, D-step serial number t can be used. Then D-plan is

$$g(k) = e(0), e(1), e(2), \dots, e(t), \dots, e(T). \quad (2.1)$$

D-state $e(0)$ is existing D-state, where no any D-action is realized. In common, case D-state may be defined as realized D-actions within the period $0-t$:

$$e(t) = e(0) \cup \{x(1)\} \cup \{x(2)\} \dots \cup \{x(t)\}, \quad (2.2)$$

where $\{x(t)\}$ —all step t initial realized D-actions set.

D-state can be classified as technically valid or technically non-valid—if at least one technical limit in D-state is not observed.

LTS D-step. LTS D-step is time period when none of D-actions is realized—system graph and its parameters are not changed. In D-step period, it is assumed that annual consumption of the first year of D-step, 24 h, seasonal and monthly consumption is preserved for the period of all further years. Performing mathematical calculations, we assume that all D-step period realized D-actions are realized simultaneously, going on from D-step $t - 1$ to D-step t . In such a way, real continuous calculation process is modeled by discrete D-plan. In general, D-step length is from 1–3 up to 10–13 years. In the initial phase of estimation period D-steps are shorter, the next D-steps length is longer, but the last D-steps are the longest.

LTS D-plan. In estimation period, which is divided into T D-steps, may be various D-states $e(t)$. A sequence of D-states $e(t)$, if $t = 1, 2, \dots, t, \dots, T$, is called large technical system D-plan, which will be marked with $g(k)$. Graphical model of D-plan is guided path graph (see Fig. 2.4).

Graph vertices and edges can be logical admissible or non-admissible (if at least one logical condition in D-plan is not observed).

D-plan $g(k)$ may be unambiguously characterized by realized D-actions and their realization time as well as estimated by summary, at the initial moment reduced, system quality criterion—objective function (investments, generation,

Fig. 2.4 Large technical system D-plan



consumption, generating unit production range, transport network flows and overloads).

D-plans are classified into two groups:

1. Logically admissible D-plans—there is no D-state where all logical conditions are observed;
2. Logically non-admissible D-plans—in any graph vertex or edge at least one logical condition is not observed; in order to operate non-admissible D-states, it is required to realize more supplementary D-actions, i.e. to create another D-plan $g(k)$.

LTS D-states set. In optimization process, LTSs are required to be reviewed not only as separate D-states but also as a set of different D-states. Let us mark D-states set with E . To distinguish different types of D-states, let us use symbol $E(i)$. The major types of D-states are the following:

1. D-states set $E(t)$, which integrates all of D-step t possible D-states;
2. Subsets of set $E(t)$: (a) D-states subset, in which D-states are integrated forming development sequel process, preserving the existing D-states or realizing additional D-actions; such subset is marked with $E[t, e(i) \rightarrow]$ and (b) D-states subset $E[t, e(i) \leftarrow] \subset E(t)$, in which D-states are integrated in D-step $t - 1$, from which by realizing one or several D-actions (or not realizing any), D-state $e(i)$ can be formed. It must be taken into consideration that in specific D-state $e(i)$ only one D-action $x(j)$ is allowed, because D-action is the realized fact and cannot be cancelled;
3. Subsets of set $E(t)$ that integrate D-states with equal number of realized D-actions.

Finally, let us review *operations with D-states sets* used in dynamic optimization process:

1. Membership of D-state e in set E : $e \in E$;
2. Calculation of function $F(e)$ for all $e \in E$: $F(e)$;
3. Union of two D-states sets $E(1)$ and $E(2)$: $E(3) = [E(1) \cup E(2)]$.

2.2 Graphical Model of Development Process

LTS D-states $e(i)$ can be regarded as graph G vertices, but graph edges are lines that connect two D-states (graph G vertices). Graph edges represent probable transition from D-state $e(t - 1)$ to D-state $e(t)$. Taking into consideration that LTS D-state is a set of realized D-actions, $e(t) = \{x\}$ [$e(0) = es$] (where es —empty set), the transition from $e(t - 1)$ to $e(t)$ is possible if $e(t - 1) \subset e(t)$. Thus, graph G edge corresponds to realized D-actions set $\{x\}_p$, within transition time; in particular cases there is only one D-action or that is also an empty set $\{x\} = es$. For Graph G edges there is only one direction—from $e(t - 1)$ to $e(t)$.

Fig. 2.5 LTS graph; existing elements are represented by *uninterrupted line*, but new constructed system elements—by *interrupted line*

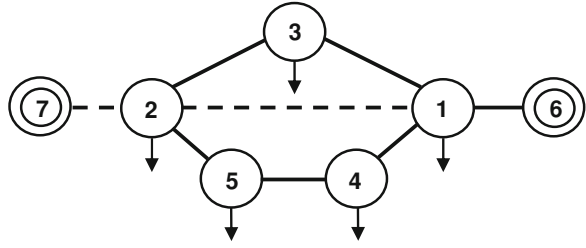


Figure 2.5 shows a system with five consumption nodes and one generation node. The highest consumption is in node 2, which is located within the furthest distance from generation node. Due to consumption rise in transport network, links 1-3-2 and 1-4-5-2 loads are close to the valid range. In addition, generation node is almost fully loaded. In order to prevent critical situation in the system, four alternative D-actions may be applied:

1. Reconstruction of existing links 1-3-2;
2. Reconstruction of existing links 1-4-5-2;
3. Construction of new links 1-2;
4. Construction of new generation node 7 at a higher load of node 2.

D-action combination set is given in Table 2.1.

Let us mark various D-states with $e(t, \chi)$. Development graph of technical system considered (if $T = 5$) is depicted in Fig. 2.6.

System development by T D-steps is illustrated with a succession of D-states (graph G vertices connected with graph edges), that is with graph loop g , which starts in vertex $e(0, 0)$ and ends in one of vertices $e \in E(T)$. For example, one of admissible graph loops is a line, which connects graph vertices sets $e(1, 3)$, $e(2, 3)$, $e(3, 3)$, $e(4, 6)$ and $e(5, 6)$. For this D-plan summary reduced system quality criterion can be calculated for five D-steps $F(5, g)$.

If system graph G is given, then all possible system D-plans are given too.

System development graph vertices set. Let us consider these sets:

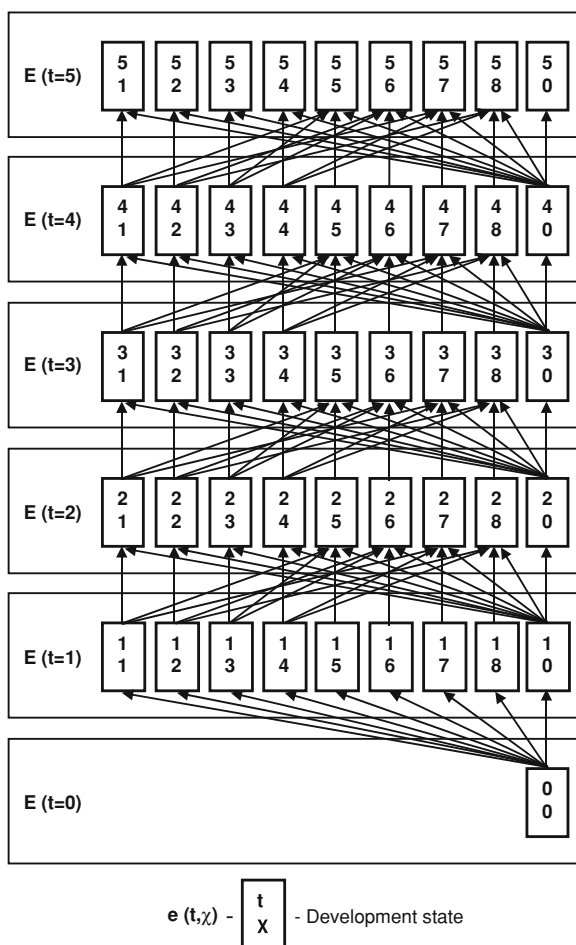
1. Vertices set in D-step $t - E(t)$;
2. Vertices set in D-step t subsets, in which all vertices have characteristics $i - E(t, i)$;
3. D-step t vertices subset $E[t, E(t - 1, i) \rightarrow]$, where all vertices are related to D-step $t - 1$ vertices $e(t - 1) \in E[(t - 1, i)]$ set $E[t, E(t - 1, i) \rightarrow]$, we mark the vertices that are related to sub-set $E(t - 1, i)$. In special case, a set may contain a single vertex, then vertices $e(t) \in E(t - 1, i)$ are interrelated only with vertex $E(t - 1, i)$;
4. Initial vertices set $E(t - 1, i)$ related to sub-set $E(t, i)$.

Let us explain definitions of the subsets considered with four examples which are illustrated in Fig. 2.3.

Table 2.1 LTS D-actions sets

No.	Code χ	Set file
1	0	Empty set
2	1	1
3	2	2
4	3	3
5	4	4
6	5	3; 1
7	6	3; 2
8	7	4; 1
9	8	4; 2

Fig. 2.6 Technical system development graph



- Example 1. A vertex set with a single vertex $e(2, 3)$ is given. Then with the given set interrelated vertices set is $\{e(3, 3), e(3, 7), e(3, 8)\}$.
- Example 2. A vertex set $\{e(2, 4)\}$ is given. Then with the given set related vertices (development follow-up) set is $\{e(3, 4), e(3, 6), e(3, 7), e(3, 10)\}$.
- Example 3. A vertex set $E(t - 1, i) = \{e(2, 2), e(2, 6)\}$ is given. Then with the given set related vertices set is $\{e(3, 2), e(3, 6), e(3, 8)\}$.
- Example 4. A vertex set $E(t - 1, i) = \{e(3, 3), e(3, 5)\}$ is given. Then initial D-states set is $E[(t - 1, i) \rightarrow] = \{e(2, 3), 2(2, 5), e(2, 0)\}$.

The following conclusions result from the reviewed concept definitions:

1. In all D-steps, all D-states and transitions must be preserved. Otherwise, D-plan is terminated.
2. Usually, the reason why D-states are not preserved is that technical constraints are not met. Generally, irrelevant D-states are occurring by incomplete D-states set.
3. In order to prevent the occurrence of irrelevant D-states, it is not allowed to utilize rigid constraint method; instead, fuzzy constraint method and penalty function must be applied. Then the first optimization can be finalized. After optimization, the results can be analyzed and D-action set can be supplemented.

2.3 Mathematical Formulation of Development Process Optimization Task

The target of LTS development optimization is to determine and identify an admissible D-plan, in which the maximal reduced system development quality criterion—objective function would be met in T D-steps. Such D-plan will be marked with $\text{optg}[T, E(T)]$ and respective objective function will look like this

$$f[T, E(T)] = \max_{g \in G} F(g). \quad (2.3)$$

The right-hand side of expression 2.3 shows that objective function maximization is taking place (if objective function incorporates also income from the sold production) over all graph G loops (routes). It may seem that the most elementary optimization is to consider in succession all probable graph loops. Still this kind of method is only admissible when D-plans scope is not considerable; but as shown in Fig. 1.8, in real tasks, astronomic figures may be obtained and calculation of D-plans might take several years. Optimization method that does not cover analysis of all D-plans $g \in G$ would be more efficient.

Let us assume that in given D-step t information available is:

1. Valid D-states set, $e(t)$ for step t ;
2. Objective function $F[e(t)]$ for development from $e(0)$ to $e(t) \in E(t)$;

3. Valid D-states set, $E(t - 1)$ for step $t - 1$;
4. D-states set from which transition is probable to $e(t, i)$:

$$E[t, e(t - 1) \rightarrow]. \quad (2.4)$$

The transitions from step $t - 1$ D-states to step t D-states $e(t)$ are determined for all $e(t - 1) \in E(t - 1)$ or all graph edges are given that connect $e(t - 1)$ with $e(t)$, as well as respective D-actions characteristics k and system graph changes;

5. Step $t - 1$ D-states set from which the given D-state $e(t)$ may be attained;

$$E[(t - 1), e(t) \leftarrow]. \quad (2.5)$$

Sets 2.5 are given for all D-states $e(t) \in E(t)$.

Besides, let us assume that for all $e(t - 1) \in E(t - 1)$ there is a certain optimal D-plan $\text{optg}[t - 1, e(t - 1)]$.

Let us mark with $f[t, e(t)]$ the maximal summary reduced criterion value within the period up to D-step t and D-state $e(t)$, i.e. is for optimal development $\text{optg}[t, e(t)]$. Let us call this value functional. It required to calculate for all D-states $e(t) \in E(t)$, if values $f[t - 1, e(t - 1)]$ are known for all D-states $e(t - 1) \in E(t - 1)$.

Functional $f[t, e(t)]$ must be expressed observing income (+) from production sold and expenditures (−):

$$f[t, E(t)] = \max_{e(t) \in E(t)} f[t, e(t)]. \quad (2.6)$$

Formula of functional calculation may be expressed as follows:

$$f[t, e(t)] = F(e(t)) + \max_{e(t-1) \in E[t-1, e(t) \leftarrow]} f[t - 1, e(t - 1)], \quad (2.7)$$

where $F(e(t))$ —reduced system quality criterion in D-step t and D-state.

Formula 2.7 is dynamic programming recursive equation that is used to optimize LTS D-plan.

In the first D-step of estimation period functional $f[t - 1, e(t - 1)]$ is system summary reduced quality criterion for previous D-steps that is independent of system further development and may be omitted in calculation. Assuming that $f[0, e(1)] = 0$, we have

$$f[t - 1, e(1)] = F(e(1)). \quad (2.8)$$

Using recursive formula 2.7, starting from the first D-step $t = 1, 2, 3, \dots, T$, in inductive way, one may calculate functional $f[t, e(t)]$ values for all D-steps and D-states and determine optimal D-plan $f[t, e(t)]$.

2.4 Algorithmization Aspects of Development Management Process

LTS development optimization applying dynamic programming recursive Eq. 2.7 can be performed in different ways depending on how recursive Eq. 2.7 is written down.

First, let us review the option in writing:

$$f[t, e(t)] = F[e(t)] + \max_{e(t-1) \in E[t-1, e(t) \leftarrow]} f[t-1, e(t-1)]. \quad (2.9)$$

Recursive Eq. 2.9 is used if all D-states are considered $e(t) \in E(t)$ for each D-state $e(t)$:

1. To calculate step t objective function item $F[e(t)]$ —to perform system D-state technical, economic and ecological criteria calculation.
2. To define D-states set $E[t-1, e(t) \leftarrow]$, from which integrated D-states $e(t-1)$ may reach D-state $e(t)$.
3. To calculate the set functional maximal value $\max_{e(t-1) \in E[t-1, e(t) \leftarrow]} f[t-1, e(t-1)]$.

Thus, optimal initial state is defined $e(t-1)$ and optimal development up to D-state $e(t) - \text{optg} \{t-1, E[t-1, e(t) \leftarrow]\}$.

In order to realize this algorithm, it is required to form operative memory data array $f[t-1, e(t-1)]$ for all D-states $e(t-1) \in E(t-1)$ and $f[t, e(t)]$ all D-states $e(t) \in E(t)$. Comparing with all D-plans calculation, the algorithm observed essentially reduces calculation time. Still, capacious computer memory consumption is required for storage of calculation interim results. It considerably limits capabilities of method application.

Let us review modifications of such recursive equation which enable are to use calculation algorithm without saving in data array functional $f[t-1, e(t-1)]$ and $f[t, e(t)]$ values on all D-states sets $E(t-1)$ and $E(t)$.

In order to elaborate such algorithm, the answer to the question must be known. By what indications the D-states, which can be used in optimal D-plan, must be identified? If such indications are available, there is no need to bear in mind functional values for all D-states.

Let us mark with $\Omega(t-1) \subset E(t-1)$ D-states $e(t-1)$ set that contains step t optimal initial states.

D-states that belong to set $\Omega(t-1)$, will be marked with $e(t-1, \omega) \in \Omega(t-1)$, where $\omega = 1, 2, 3, \dots, \omega(\max)$.

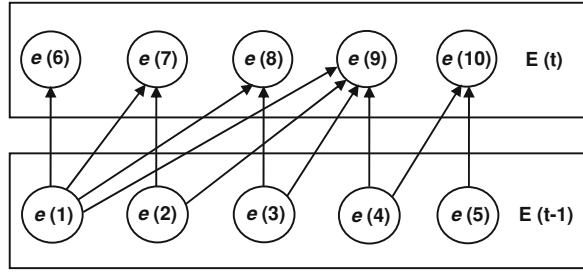
The set $\Omega(t)$ may be defined by the following method (see Fig. 2.7).

D-state $e(t-1, \omega = 1)$ is determined by formula

$$f[t-1, e(t-1, \omega = 1)] = \max_{e(t-1) \in E(t-1, \omega=1)} f[t-1, e(t-1)], \quad (2.10)$$

where $E(t-1, \omega = 1) = E(t-1)$.

Fig. 2.7 D-states sets $\Omega(t-1)$ (required for optimal development estimation/calculation) elaboration algorithm illustration



$$E(t-1, \omega=1) = \{e(1), e(2), e(3), e(4), e(5)\}; \quad e(t-1, \omega=1) = e(1);$$

$$\Omega(t-1, \omega=1) = \{e(t-1, \omega=1), \dots\};$$

$$E[t, \Omega(t-1, \omega=1) \rightarrow] = \{e(6), e(7), e(8), e(9)\};$$

$$\overline{\Omega}(t-1, \omega=1) = \{e(1), e(2), e(3)\}; \quad \overline{\overline{\Omega}}(t-1, \omega=1) = \{e(1), e(2), e(3)\};$$

$$E(t-1, \omega=2) = \{e(4), e(5)\}; \quad e(t-1, \omega=2) = e(4);$$

$$\Omega(t-1, \omega=1) = \{e(t-1, \omega=1), e(t-1, \omega=2)\};$$

$$E[t, \Omega(t-1, \omega=2) \rightarrow] = \{e(6), e(7), e(8), e(9), e(10)\};$$

$$\overline{\Omega}(t-1, \omega=2) = \{e(1), e(2), e(3), e(4), e(5)\};$$

$$\overline{\overline{\Omega}}(t-1, \omega=2) = \{e(1), e(2), e(3), e(4), e(5)\}; \quad \Omega(t-1) = \{e(1), e(4)\}.$$

D-state $e(t-1, \omega=1)$ is optimal development by $t-1$ steps final state.

For D-state $e(t-1, \omega=1)$, related set is defined as

$$E[t-1, e(t-1, \omega=1) \rightarrow]. \quad (2.11)$$

Sets 2.11 initial set is the set

$$\text{initial } E[t-1, e(t-1, \omega=1) \rightarrow]. \quad (2.12)$$

D-states set 2.12 comprises D-state $e(t-1) \in \Omega(t-1, \omega=1)$, from which transition is only possible to set 2.11.

In general case, the transition from set $\Omega(t-1, \omega=1)$ to set $E(t)$ is possible in many ways, i.e. to several D-states $e(t)$. Each D-state $e(t)$ has its own system quality criterion value:

$$F[t, e(t)] = f[t-1, e(t-1)] + F[e(t)]. \quad (2.13)$$

Formula 2.13 shows that optimal development may not occur through D-state $e(t-1) \in \Omega(t-1, \omega=1)$, for which the following condition is valid

$$f(t-1, e(t-1)) + F(e(t)) < f(t-1, e(t-1, \omega=1)) + F(e(t)). \quad (2.14)$$

This condition can be verified even without calculating objective function component $F(e(t))$ of step t for D-states $e(t)$. Excluding from set $\Omega(t-1, \omega=1)$ all D-states that do not conform to condition 2.14, we obtain D-states set $\Omega(t-1,$

$\omega = 1$), in which optimal development can be determined only from D-state $e(t - 1, \omega = 1)$.

It often happens that condition 2.14 is valid for all D-states $e(t - 1) \in \Omega(t - 1, \omega = 1)$. In such cases $\bar{\Omega}(t - 1, \omega) = \bar{\Omega}(t - 1, \omega)$.

D-states $e(t - 1, \omega = 1)$ are calculated by formula:

$$f[t - 1, e(t - 1, \omega = 2)] = \max_{e(t-1) \in E(t-1, \omega=2)} f[t - 1, e(t - 1)], \quad (2.15)$$

where

$$E(t - 1, \omega = 2) = E(t - 1) - \bar{\Omega}(t - 1, e(t - 1)). \quad (2.16)$$

Let us mark with $\Omega(t - 1, \omega)$ D-states set

$$\Omega(t - 1, \omega) = \{e(t - 1, \omega - 1), e(t - 1, \omega - 2), \dots, e(t - 1, \omega)\}. \quad (2.17)$$

The related D-states set we express as

$$\bar{\Omega}(t - 1, \omega = 2) = \bar{\Omega}(t - 1, \omega = 1) \cup OIS E[(t, e(t - 1, \omega = 2)) \rightarrow]. \quad (2.18)$$

From set 2.18, we exclude all D-states which do not meet this condition:

$$f[t - 1, e(t - 1)] + F(e(t)) < f[t - 1, e(t - 1, \omega)] + F(e(t)). \quad (2.19)$$

The reviewed analysis allows us to draw the conclusion: technical system development graph loop that is relevant to optimal D-plan may comprise only vertices $E(t - 1)$, which are included in the subset of optimal initial states set $\Omega(t - 1)$:

$$e(t - 1, \omega) \in \Omega(t - 1). \quad (2.20)$$

D-state $e(t - 1, \omega)$ can be defined using formula:

$$f[t - 1, e(t - 1, \omega)] = \max_{e(t-1) \in E(t-1, \omega)} f[t - 1, e(t - 1)], \quad (2.21)$$

where

$$E(t - 1, \omega) = E(t - 1) - \bar{\Omega}(t - 1, \omega - 1). \quad (2.22)$$

In order to calculate functional $f[t, E(t)]$ and large technical system optimal development up to any D-step t , it is required in D-step $t - 1$ to keep data on all $[\omega(\max)]$ optimal initial states $e(t - 1)$.

D-states set E is set's $E(t, \omega)$ subset, where is D-state $e(t, \omega)$:

$$E = E(t, \omega) \cap E[t, e(t - 1, \omega)]. \quad (2.23)$$

Functional $f[t - 1, e(t - 1, \omega)]$ value can be calculated using the following recursive equation:

$$f[t, E(t, \omega)] = \max_{e(t-1, \omega) \in \Omega(t-1)} f[t-1, e(t-1, \omega)] + \max_{e(t) \in E} F(e(t)), \quad (2.24)$$

where

$$E(t, \omega) = E(t) - \overline{\Omega}(t, \omega - 1). \quad (2.25)$$

Recursive Eq. 2.24 may be used to calculate functional value for the given D-states set $E(t, \omega)$ with the algorithm described below.

1. Review all D-states $e(t-1, \omega) \in \Omega(t-1)$ from $\omega = 1$ up to $\omega = \omega(\max)$.
2. For each D-state $e(t-1, \omega)$ define $\max_{e(t) \in E} F[e(t)]$. This task corresponds to LTS

statistical optimization, if optimization area is D-states set E. The task solution is D-state with calculable $F[e(t)]$ maximal value. The D-state must conform to these two conditions:

- D-state is relevant to set $E(t, \omega)$,
- The transition from D-state $e(t-1, \omega)$ to $e(t)$ must be possible.

3. From all sums $\max_{e(t-1, \omega) \in \Omega(t-1)} \sum (F(e(t)) + f[t-1, e(t-1), \omega])$ the maximal sum shall be calculated, considering all sets $\Omega(t-1)$ D-states.

Simultaneously applying formulas 2.24 and 2.25, functionals $f[t, E(t, \omega)]$ must be calculated for all ω values $\omega = 1, 2, \dots, \omega(\max)$.

Among D-states $e(t, \omega) \in \Omega(t)$ there is also an optimal D-plan from $t = 1$ up to $t = T$ (in estimation period) D-state $e(t, \omega)$ in D-step t .

LTS optimization task solution is D-state $E(T, \omega = 1)$ with summary reduced system quality criterion $f[T, E(t, \omega = 1)]$.

Comparing both considered quality optimization algorithms; the following conclusions can be drawn:

1. The application of recursive Eq. 2.9 makes algorithms execution more simple. Algorithm drawback is the demand for all potential D-states $E(t-1)$ and $E(t)$ to keep calculation interim results and to calculate system quality criteria. D-states number in set $E(t)$ is

$$V_E = 2^n, \quad (2.26)$$

where n —D-actions number. If $n = 20$, which is possible in real optimization tasks, then $V_E = 2^{20} = 10^6$.

2. Applying recursive Eq. 2.24 and expression 2.25, it is not required to keep calculation interim results for all D-states $E(t-1)$ and $E(t)$. This algorithm allows one to simultaneously utilize dynamic optimization methods, applying summary reduced system quality criterion, and system D-state static optimization to one consumption level (one D-step). If static optimization methods of LTSD-states are effective, then recursive Eq. 2.24 would be effective.

Unfortunately, such methods have not been established yet and therefore the authors are not familiar with them.

References

1. Bellman RE (1957) Dynamic programming. Princeton University Press, Princeton, New Jersey
2. Bellman RE (1961) Adaptive control processes: A. Guided Tour. Princeton University Press, Princeton, New Jersey
3. Hall MJR. (1967) Combinatorial theory. Blaisdell Publishing Company, Waltham, Massachusetts-Toronto-London
4. Mendelson E (1997) Introduction to mathematical logic (4th ed.), Chapman & Hall, London, ISBN 978-0-412-80830-2
5. Ore O (1962) Theory of graphs. American mathematical society, Colloquium Publications, Volume XXXVIII, Providence, Rhode Island
6. Riordan J (1958) An introduction to combinatorial analysis. John Wiley & Sons, Inc., New York, Chapman & Hall, Limited, London

Dynamic Management of Sustainable Development
Methods for Large Technical Systems

Krishans, Z.; Mutule, A.; Merkuryev, Y.; Oleinikova, I.

2011, XXXII, 164 p., Hardcover

ISBN: 978-0-85729-055-7