URAL MATHEMATICAL JOURNAL, Vol. 4, No. 2, 2018, pp. 43–55

DOI: 10.15826/umj.2018.2.006

OPTIMIZING THE STARTING POINT IN A PRECEDENCE CONSTRAINED ROUTING PROBLEM WITH COMPLICATED TRAVEL COST FUNCTIONS[1]

Alexander G. Chentsov[†], **Alexey M. Grigoryev**[††] **and Alexey A. Chentsov**[†††]

Krasovskii Institute of Mathematics and Mechanics,
Ural Branch of the Russian Academy of Sciences,
16 S. Kovalevskaya str., Ekaterinburg, Russia, 620990

[†]chentsov@imm.uran.ru, [††]ag@uran.ru, [†††]chentsov_a_a@mail.ru

Abstract: We study the optimization of the initial state, route (a permutation of indices), and track in an extremal problem connected with visiting a finite system of megalopolises subject to precedence constraints where the travel cost functions may depend on the set of (pending) tasks. This problem statement is exemplified by the task to dismantle a system of radiating elements in case of emergency, such as the Chernobyl or Fukushima nuclear disasters. We propose a solution based on a parallel algorithm, which was implemented on the Uran supercomputer. It consists of a two-stage procedure: stage one determines the value (extremum) function over the set of all possible initial states and finds its minimum and also the point where it is achieved. This point is viewed as a base of the optimal process, which is constructed at stage two. Thus, optimization of the starting point for the route through megalopolises, connected with conducting certain internal tasks there, is an important element of the solution. To this end, we employ the apparatus of the broadly understood dynamic programming with elements of parallel structure during the construction of Bellman function layers.

Key words: Dynamic programming, Route, Sequencing, Precedence constraints, Parallel computation.

Introduction

In this paper, we consider an additive routing problem aimed at applications in nuclear power generation: the intention is to decrease the exposure of power plant staff to radiation during a sequence of work-related activities. The considered problem features precedence constraints, multiple variants of movements, and travel cost functions that could depend on the set of pending tasks. The mentioned features of the statement stem from the peculiarities of the actual engineering problem, which exhibits a qualitative difference from its prototype, the well-known intractable traveling salesman problem (TSP); see [1–6]. In a series of papers, the authors have developed solution methods based on dynamic programming (DP) combined with parallel computations, see [7–12] et al. Here, we consider the statement where, in addition to a solution in the form of a route-track pair, one also has to choose the starting point (the base) for the process of movements. We found out that the DP-based procedure used in [7–11] could be used just as well to solve such an "expanded" problem (see also the monograph [13], connected with issues of decreasing staff exposure to radiation during a sequence of operations).

1. General notation and definitions

We use the quantifiers and logical connectives; $\emptyset$ denotes the empty set and $\triangleq$ denotes the equality by definition. To arbitrary objects α and β, assign the set $\{\alpha;\beta\}$ that contains α and β and them only. If x is an object, then $\{x\} \triangleq \{x;x\}$ is the singleton that contains x. A set is an object,

[1]This work was supported by Russian Science Foundation (project no. 14-11-00109).

hence [14, p. 59], to objects y and z, one can assign an ordered pair (OP) $(y,z) \triangleq \{\{y\};\{y;z\}\}$ of these objects; y is the first and z is the second element of this OP. To every OP z, assign the first element $\mathrm{pr}_1(z)$ and the second element $\mathrm{pr}_2(z)$, which are uniquely defined by the condition $z = \big(\mathrm{pr}_1(z), \mathrm{pr}_2(z)\big)$. If a, b, and c are objects, then $(a,b,c) \triangleq \big((a,b),c\big)$ is the triple of these objects, constructed as an OP with the first element (a,b) and the second element c.

To every set S, assign the family $\mathcal{P}(S)$ of all its subsets; $\mathcal{P}'(S) \triangleq \mathcal{P}(S)\setminus\{\emptyset\}$; and $\mathrm{Fin}(S)$ is the family of all finite sets from $\mathcal{P}'(S)$. The family $\mathrm{Fin}(S)$ consists of the finite nonempty subsets of S and them only. To nonempty sets A and B, assign the nonempty set B^A of all mappings from A to B (see [14, p. 70]); for $g \in B^A$ and $C \in \mathcal{P}(A)$, in the form $g^1(C) \triangleq \{g(x) : x \in C\} \in \mathcal{P}(B)$, we have the image of C under g; $g^1(C) \neq \emptyset$ when $C \neq \emptyset$. If A, B, and C are three nonempty sets, then [15, p. 5] $A\times B\times C \triangleq (A\times B)\times C$; if, in addition, D is a nonempty set and $h \in D^{A\times B\times C}$, then, for $x \in A\times B$ and $y \in C$, we have $(x,y) \in A\times B\times C$ and the value $h(x,y) \in D$ of the mapping h at the point (x,y) is well-defined; for this value, we also have $h(x,y) = h\big(\mathrm{pr}_1(x), \mathrm{pr}_2(x), y\big)$.

As usual, $\mathbb{N} \triangleq \{1;2;\ldots\}$; set $\mathbb{N}_0 \triangleq \{0\}\cup\mathbb{N}$ and $\overline{p,q} \triangleq \{j \in \mathbb{N}_0 |\, (p \leqslant j)\&(j \leqslant q)\}$ $\forall p \in \mathbb{N}_0$ $\forall q \in \mathbb{N}_0$ (if $k \in \mathbb{N}_0$, $l \in \mathbb{N}_0$, and $l < k$, then $\overline{k,l} = \emptyset$). To every nonempty finite set K assign its cardinality $|K| \in \mathbb{N}$ and the nonempty set $(\mathrm{bi})[K]$ of all bijections of the integer interval $\overline{1,|K|}$ onto K; $|\emptyset| \triangleq 0$. By $\mathbb{R}$ we denote the real line; $\mathbb{R}_+ \triangleq \{\xi \in \mathbb{R} |\, 0 \leqslant \xi\}$; and $\mathcal{R}_+[T]$, where T is a nonempty set, denotes the set of all functions from T to $\mathbb{R}_+$, that is, $\mathcal{R}_+[T] \triangleq (\mathbb{R}_+)^T$.

2. Problem statement

Fix a nonempty set X and some $X^0 \in \mathrm{Fin}(X)$; the points X^0 are viewed as admissible starting points. Let $N \in \mathbb{N}$, $N \geqslant 2$,

$$M_1 \in \mathrm{Fin}(X), \ldots, M_N \in \mathrm{Fin}(X),$$

and let $\mathbb{M}_1 \in \mathcal{P}'(M_1\times M_1), \ldots, \mathbb{M}_N \in \mathcal{P}'(M_N \times M_N)$; assume

$$(X^0 \cap M_j = \emptyset \quad \forall j \in \overline{1,N})\&(M_p \cap M_q = \emptyset \quad \forall p \in \overline{1,N} \quad \forall q \in \overline{1,N}\setminus\{p\})$$

and set $\mathbb{P} \triangleq (\mathrm{bi})[\overline{1,N}]$. Consider the processes

$$\begin{gathered}(x^{(0)} = x^0 \in X^0) \to (x_1^{(1)} \in M_{\alpha(1)} \rightsquigarrow x_2^{(1)} \in M_{\alpha(1)}) \to \ldots \\ \to (x_1^{(N)} \in M_{\alpha(N)} \rightsquigarrow x_2^{(N)} \in M_{\alpha(N)})\end{gathered} \tag{2.1}$$

where $\alpha \in \mathbb{P}$, $z_1 \in (x_1^{(1)}, x_2^{(1)}) \in \mathbb{M}_{\alpha(1)}, \ldots, z_N \in (x_1^{(N)}, x_2^{(N)}) \in \mathbb{M}_{\alpha(N)}$. The permutation α determines the route, that is, the sequence the megalopolises are visited in while $z_1,\ldots,z_N$ determines the track of these visits; x^0 is the initial state. A complete solution (see (2.1)) is a tuple $(x^0, \alpha, z_1, \ldots, z_N)$, to be determined. The choice of $\alpha \in \mathbb{P}$ may be restricted by precedence constraints; to describe them, fix

$$\mathbf{K} \in \mathcal{P}(\overline{1,N}\times\overline{1,N}),$$

that is, the set of OPs known as “address pairs” (see [7–11, 13]); the case $\mathbf{K} = \emptyset$ denotes the absence of precedence constraints. Assume that

$$\forall \mathbf{K}_0 \in \mathcal{P}'(\mathbf{K})\ \exists z_0 \in \mathbf{K}_0 :\ \mathrm{pr}_1(z_0) \neq \mathrm{pr}_2(z) \quad \forall z \in \mathbf{K}_0.$$

In the form

$$\mathbf{A} \triangleq \{\alpha \in \mathbb{P} |\, \alpha^{-1}\big(\mathrm{pr}_1(z)\big) < \alpha^{-1}\big(\mathrm{pr}_2(z)\big) \quad \forall z \in \mathbf{K}\} \in \mathcal{P}'(\mathbb{P}), \tag{2.2}$$

we have a (nonempty) set of $\mathbf{K}$-feasible routes. Let $\mathbb{X} \triangleq X^0 \cup \big(\bigcup_{i=1}^{N} M_i\big)$; then, $\mathbb{X} \in \mathrm{Fin}(X)$. Denote by $\mathbb{Z}$ the set of all tuples $(z_i)_{i\in\overline{0,N}} : \overline{0,N} \to \mathbb{X}\times\mathbb{X}$, that is, $\mathbb{Z} \triangleq (\mathbb{X}\times\mathbb{X})^{\overline{0,N}}$. If $x^0 \in X^0$ and $\alpha \in \mathbb{P}$, then

$$\mathcal{Z}_\alpha[x^0] \triangleq \{\mathbf{z} \in \mathbb{Z} |\, \big(\mathbf{z}(0) = (x^0, x^0)\big) \& \big(\mathbf{z}(t) \in \mathbb{M}_{\alpha(t)} \quad \forall t \in \overline{1,N}\big)\} \in \mathrm{Fin}(\mathbb{Z}). \tag{2.3}$$

Therefore, for $x^0 \in X^0$, in the form

$$\widetilde{\mathbf{D}}[x^0] \triangleq \{(\alpha, \mathbf{z}) \in \mathbf{A}\times\mathbb{Z} |\, \mathbf{z} \in \mathcal{Z}_\alpha[x^0]\} \in \mathrm{Fin}(\mathbf{A}\times\mathbb{Z}),$$

we have a (nonempty finite) set of feasible solutions (FS) of the problem with the fixed initial state. Next, let us note that

$$\mathbf{D} \triangleq \{(\alpha, \mathbf{z}, x) \in \mathbf{A}\times\mathbb{Z}\times X^0 |\, (\alpha, \mathbf{z}) \in \widetilde{\mathbf{D}}[x]\} \in \mathrm{Fin}(\mathbf{A}\times\mathbb{Z}\times X^0) \tag{2.4}$$

is viewed as the set of all FSs of the complete problem.

Consider the following transportation cost functions. Let $\mathfrak{N} \triangleq \mathcal{P}'(\overline{1,N})$; $\mathbf{c} \in \mathcal{R}_+[\mathbb{X}\times\mathbb{X}\times\mathfrak{N}]$; and let $c_1 \in \mathcal{R}_+[\mathbb{X}\times\mathbb{X}\times\mathfrak{N}], \ldots, c_N \in \mathcal{R}_+[\mathbb{X}\times\mathbb{X}\times\mathfrak{N}], f \in \mathcal{R}_+[\mathbb{X}]$. In terms of the tuple

$$(\mathbf{c}, c_1, \ldots, c_N, f),$$

we define the additive criterion: for $x^0 \in X^0$ and $(\alpha, \mathbf{z}) \in \widetilde{\mathbf{D}}[x^0]$, assume

$$\begin{aligned}\mathfrak{C}_\alpha[\mathbf{z}] \triangleq \sum_{s=1}^{N} \Big[\mathbf{c}\Big(\mathrm{pr}_2\big(\mathbf{z}(s-1)\big), \mathrm{pr}_1\big(\mathbf{z}(s)\big), \alpha^1(\overline{s,N})\Big) + \\ + c_{\alpha(s)}\big(\mathbf{z}(s), \alpha^1(\overline{s,N})\big)\Big] + f\Big(\mathrm{pr}_2\big(\mathbf{z}(N)\big)\Big);\end{aligned} \tag{2.5}$$

thus, to each FS $(\alpha, \mathbf{z}, x^0) \in \mathbf{D}$, we assign the value $\mathfrak{C}_\alpha[\mathbf{z}] \in \mathbb{R}_+$, which does not explicitly depend on x^0 (x^0 affects the choice of $\mathbf{z}$). Like in [7–11], for $x^0 \in X^0$, let us introduce the problem

$$\mathfrak{C}_\alpha[\mathbf{z}] \longrightarrow \min, \quad (\alpha, \mathbf{z}) \in \widetilde{\mathbf{D}}[x^0], \tag{2.6}$$

for which the value $V[x^0]$ is determined as the least value among $\mathfrak{C}_\alpha[\mathbf{z}]$, $(\alpha, \mathbf{z}) \in \widetilde{\mathbf{D}}[x^0]$, and also the (nonempty) set

$$(\mathrm{SOL})[x^0] \triangleq \{(\alpha_0, \mathbf{z}_0) \in \widetilde{\mathbf{D}}[x^0] |\, \mathfrak{C}_{\alpha_0}[\mathbf{z}_0] = V[x_0]\} \in \mathrm{Fin}(\widetilde{\mathbf{D}}[x^0]). \tag{2.7}$$

In addition, we have the following complete problem

$$\mathfrak{C}_\alpha[\mathbf{z}] \longrightarrow \min \quad (\alpha, \mathbf{z}, x) \in \mathbf{D}, \tag{2.8}$$

with the value

$$\mathbb{V} \triangleq \min_{(\alpha,\mathbf{z},x)\in\mathbf{D}} \mathfrak{C}_\alpha[\mathbf{z}] \in \mathbb{R}_+ \tag{2.9}$$

and a (nonempty) set

$$\mathbb{SOL} \triangleq \{(\alpha^0, \mathbf{z}^0, x^0) \in \mathbf{D} |\, \mathfrak{C}_{\alpha^0}[\mathbf{z}^0] = \mathbb{V}\} \in \mathrm{Fin}(\mathbf{D}).$$

In connection with (2.8), it is also of interest to consider the problem

$$V[x] \longrightarrow \min, \quad x \in X^0; \tag{2.10}$$

(2.10) is the problem of optimizing the starting point, which is of some interest in itself. Indeed, if (2.10) is solved, we get $\mathbb{V}$ (2.9) and the point $x^0 \in X^0$ such that $\mathbb{V}$ is achieved by $V[x^0]$. One could construct heuristics (when necessitated by the problem's dimension) for solving (2.6), compare their results with $\mathbb{V}$, and thereby choose what is deemed admissible. In this connection, note that (see (2.4))

$$\mathbb{V} = \min_{x \in X^0} \min_{(\alpha,\mathbf{z}) \in \tilde{\mathbf{D}}[x]} \mathfrak{C}_\alpha[\mathbf{z}] = \min_{x \in X^0} V[x]. \tag{2.11}$$

To solve the problems of the form (2.6), one can use the broadly understood DP in the spirit of [7–11, 13]; here, we consider these procedures in their algorithmic form (see [11, 16]).

3. Dynamic programming in starting point optimization problem

This section serves to adapt the DP procedure from papers [7–11, 13, 16] to the needs of solving problem (2.10). To this end, let us introduce the crossing-out operator $\mathbf{I}$, which acts in $\mathfrak{N}$: for $K \in \mathfrak{N}$, assume

$$\mathbf{I}(K) \overset{\triangle}{=} K \setminus \{\mathrm{pr}_2(z) : z \in \Xi[K]\}, \tag{3.1}$$

where $\Xi[K] \overset{\triangle}{=} \{z \in \mathbf{K} | \big(\mathrm{pr}_1(z) \in K\big)\&\big(\mathrm{pr}_2(z) \in K\big)\}$ (note that $\mathbf{I}(\{t\}) = \{t\}$ for $t \in \overline{1,N}$). In terms of $\mathbf{I}$ (3.1), let us introduce the family

$$\mathfrak{C} \overset{\triangle}{=} \{K \in \mathfrak{N} | \forall z \in \mathbf{K} \ \ \big(\mathrm{pr}_1(z) \in K\big) \Rightarrow \big(\mathrm{pr}_2(z) \in K\big)\}$$

of feasible (task) sets and its subfamilies $\mathfrak{C}_s \overset{\triangle}{=} \{K \in \mathfrak{C} | \, s = |K|\} \quad \forall s \in \overline{1,N}$. Note that $\mathfrak{C}_N = \{\overline{1,N}\}$ and $\mathfrak{C}_{s-1} = \{K \setminus \{t\} : K \in \mathfrak{C}_s, \ t \in \mathbf{I}(K)\} \quad \forall s \in \overline{2,N}$ (we have (see [16]) a recurrence procedure for constructing $\mathfrak{C}_1, \ldots, \mathfrak{C}_N$). For $\mathbf{K}_1 \overset{\triangle}{=} \{\mathrm{pr}_1(z) : z \in \mathbf{K}\}$, we have the equality $\mathfrak{C}_1 = \big\{\{t\} : t \in \overline{1,N} \setminus \mathbf{K}_1\big\}$. Let $\mathbf{M}_t \overset{\triangle}{=} \{\mathrm{pr}_2(z) : z \in \mathbb{M}_t\} \quad \forall t \in \overline{1,N}$. In addition, let

$$\mathbf{X} \overset{\triangle}{=} X^0 \cup \big(\bigcup_{t=1}^{N} \mathbf{M}_t\big). \tag{3.2}$$

Consider the construction of layers of the state space, that is, the layers of the set $\mathbf{X} \times \mathcal{P}(\overline{1,N})$. To this end, first, denote by $\widetilde{\mathcal{M}}$ the union of all the sets $\mathbf{M}_t$, $t \in \overline{1,N} \setminus \mathbf{K}_1$; then, set

$$D_0 \overset{\triangle}{=} \{(x, \emptyset) : x \in \widetilde{\mathcal{M}}\}.$$

In addition, set $D_N \overset{\triangle}{=} \{(x, \overline{1,N}) : x \in X^0\}$; D_0 and D_N are the boundary state space layers.

Constructing intermediary layers. If $s \in \overline{1,N-1}$ and $K \in \mathfrak{C}_s$, then let us define, in a sequential fashion, the three sets

$$\begin{gathered} J_s(K) \overset{\triangle}{=} \{j \in \overline{1,N} \setminus K | \, \{j\} \cup K \in \mathfrak{C}_{s+1}\}, \\ \mathcal{M}_s[K] \overset{\triangle}{=} \bigcup_{j \in J_s(K)} \mathbf{M}_j, \\ \mathbb{D}_s[K] \overset{\triangle}{=} \{(x, K) : x \in \mathcal{M}_s[K]\}. \end{gathered} \tag{3.3}$$

In view of (3.3), for $s \in \overline{1,N-1}$, let D_s be the union of all the sets $\mathbb{D}_s[K]$, $K \in \mathfrak{C}_s$; then, $\emptyset \neq D_s \subset \mathbf{X} \times \mathfrak{C}_s$.

In view of the definitions of D_0 and D_N, we see that, in particular, $(D_s)_{s\in\overline{0,N}}$ is a tuple of subsets of $\mathbf{X}\times\mathcal{P}(\overline{1,N})$. Thus we obtain the state space layers. Let us now define the functions

$$v_0 \in \mathcal{R}_+[D_0], v_1 \in \mathcal{R}_+[D_1], \dots, v_N \in \mathcal{R}_+[D_N]$$

in a sequential fashion. Set $v_0(x,\emptyset) \overset{\triangle}{=} f(x)\ \forall x \in \widetilde{\mathcal{M}}$; thus, we obtain v_0.

Under $s \in \overline{1,N}$, $(x,K) \in D_s$, $j \in \mathbf{I}(K)$, and $z \in \mathbb{M}_j$, we obtain (see [16, (4.9)])

$$\big(\mathrm{pr}_2(z), K\setminus\{j\}\big) \in D_{s-1}.$$

In view of this property, for $s \in \overline{1,N}$, we define the transformation of v_{s-1} into v_s through [16, Proposition 4.1]:

$$\begin{gathered} v_s(x,K) \overset{\triangle}{=} \min_{j\in\mathbf{I}(K)} \min_{z\in\mathbb{M}_j} \big[\mathbf{c}\big(x,\mathrm{pr}_1(z),K\big) + c_j(z,K) + \\ + v_{s-1}\big(\mathrm{pr}_2(z), K\setminus\{j\}\big)\big] \quad \forall (x,K) \in D_s. \end{gathered} \tag{3.4}$$

This implements the recurrence procedure $v_0 \to v_1 \to \ldots \to v_N$.

Proposition 3.1. *If $x^0 \in X^0$, then $v_N(x^0,\overline{1,N}) = V[x^0]$.*

P r o o f. Fix $x^0 \in X^0$, which implies $x^0 \in X$. Consider problem (2.6). The way of solving this problem is described, in particular, in [16]; in the same paper, there are also constructed the feasible task set families $\mathfrak{C}, \mathfrak{C}_1, \dots, \mathfrak{C}_N$ similar to those mentioned in the beginning of the section. Based on that (in [16]), state space layers $\mathcal{D}_0, \mathcal{D}_1, \dots, \mathcal{D}_N$, similar to $D_0, D_1, \dots, D_N$ (see, in particular, (3.3) and [16, Section 4]), are constructed. There is a difference only for D_N and $\mathcal{D}_N$: here, we have

$$D_N = X^0 \times \{\overline{1,N}\} = \{(x,\overline{1,N}) : x \in X^0\},$$

whereas, in [16], $\mathcal{D}_N = \{(x^0,\overline{1,N})\}$, whence $\mathcal{D}_N \subset D_N$. Next, the construction of $v_0, v_1, \dots, v_{N-1}$ in [16, Section 4] and in the present section is the same (see, in particular, (3.4) and [16, Proposition 4.1]). Therefore, in particular, v_{N-1} matches that of [16, Section 4]. At the same time, $V[x^0]$ matches V [16, (3.18)]. Thus, in accordance with [16, (4.12)],

$$V[x^0] = \min_{j\in\mathbf{I}(\overline{1,N})} \min_{z\in\mathbb{M}_j} \big[\mathbf{c}\big(x^0,\mathrm{pr}_1(z),\overline{1,N}\big) + c_j(z,\overline{1,N}) + v_{N-1}\big(\mathrm{pr}_2(z),\overline{1,N}\setminus\{j\}\big)\big], \tag{3.5}$$

where $\big(\mathrm{pr}_2(\tilde{z}),\overline{1,N}\setminus\{j\}\big) \in D_{N-1}$ for $j \in \mathbf{I}(\overline{1,N})$ and $\tilde{z} \in \mathbb{M}_j$. However, $(x^0,\overline{1,N}) \in D_N$, thus, in the right-hand side of (3.5), we have (see (3.4)) $v_N(x^0,\overline{1,N})$, which completes the proof. □

From Proposition 3.1, we see that problem (2.10) takes the following form:

$$v_N(x,\overline{1,N}) \longrightarrow \min, \quad x \in X^0. \tag{3.6}$$

In (3.6), we have an exhaustive search for the minimum of the function $v_N(\cdot,\overline{1,N})$ over the finite set X^0. In this connection, we propose the following algorithm for solving problem (2.10).

4. Algorithm for optimization of starting point

Algorithm 4.1.

(1) In terms of f, define the function v_0.

(2) If $s \in \overline{1,N}$ and the function v_{s-1} have been constructed already, conduct the transformation $v_{s-1} \to v_s$ based on (3.4).

(3) After v_s has been constructed through the rule (3.4), the values of the function v_{s-1} are erased and replaced by the values of the function v_s (the Bellman function layers are overwritten).

(4) After the function v_N has been constructed, solve the problem (3.6): determine $\mathbb{V}$ and the minimum of the function $v_N(\cdot, \overline{1,N})$, which has the form

$$x \longmapsto v_N(x, \overline{1,N}) : X^0 \to \mathbb{R}_+.$$

As noted before, the solution of problem (2.10) could be used to test the heuristics, which are to be employed on larger problem instances.

Coming back to the problem (2.8), note that the aforementioned algorithm (which admits a natural analogy with [16]) must be modified: the layers $v_1, \dots, v_N$ will have to be retained in the computer's memory. We also have to select the point $x^0 \in X^0$ that is a solution of problem (2.10), that is, $V[x^0] = \mathbb{V}$. Next, use the algorithm [16, Section 4] (see also [17, § 7], where a slightly more general statement was considered). The logic of constructions here follows that of [7–11, 13].

To construct an optimal solution after the optimal starting point has been found — the pair of a route and a track — we use the algorithm [16, Section 4] (see also [7, 11]). In this case, we have to retain in the computer's memory all the layers of the corresponding (to the found starting point) "part" of the Bellman function. At this stage, it is also possible to repeat the construction of the layers of the mentioned "part" that corresponds to the solution of problem (3.6). We omit this construction and refer the reader to [7, 16, 19] for details.

Using the independent computations scheme. Returning to problem (3.6), note that its most significant step — the construction of the Bellman function layers — is conducted through the independent computations scheme (see papers [17, 18]), which transfers to problem (3.6) without significant modifications because the actual object of construction in [17, 18] (and also [9, 11]) is the function v_{N-1}. Thus, we omit the theoretical description of the independent computations scheme in the spirit of [17, 18], and the parallel algorithm itself is only briefly described in connection with its software implementation for a supercomputer. The differences with [17, 18] only appear in the final computations of the form (3.5) (in [17, 18], a single computation was required, whereas, for problem (3.6), the number of computations matches the number of elements in the set X^0). A version of parallel implementation as described in [17, 18] can be used both for solving problem (2.10), (3.6) (when the Bellman function layers get overwritten, see step (1)–(4) of the Algorithm 4.1), and in subsequent construction of the optimal solution in the form of a route-track pair "tied" to the minimum of problem (3.6). Following [17, 18], we distribute the sets from $\mathfrak{C}_{N-1}$ between the nodes, creating thus a finite collection of independent computation procedures; these procedures could, in part, overlap (the layers $D_s, s \in \overline{1, N-1}$ are covered by the "individual" state space layers, which do not normally reduce to a partition; the systems of individual layers, each connected with a fixed set $K \in \mathfrak{C}_{N-1}$, denote the "theaters" of the corresponding nodes). Each of the mentioned computational procedures yields a "part" (to be more precise, a restriction to a nonempty subset of D_{N-1}) of the function v_{N-1}.

Application to a dismantling problem. One natural version of the general statement can be connected with the problem of dismantling, one by one, in a sequence, a finite system of radiating elements. The goal is to minimize the total radiation dose incurred by a staff member, by means of selecting the starting point, the route (in the form of a permutation of indices), and the actual trajectory. In this special case, problem (2.10), (3.6) has the following sense: namely, where specifically should the agent (or a crew) be brought to minimize the total radiation dose in view of the subsequent optimization of the route and track. At the same time, technology-determined precedence constraints have to be satisfied, and the travel costs present a rather complicated form of dependence on the set of tasks that have not been completed yet at the time of travel. Let us now discuss one fragment of the construction of the mentioned function.

Cost function. Assume that, on a plane, there are given the points x and y, $x \neq y$; consider the travel from x to y assuming the set $K \in \mathrm{Fin}(X)$, where $X = \mathbb{R} \times \mathbb{R}$, is formed by the radiation sources that are have not been dismantled yet. Then, the radiation dose $\mathbf{c}(x, y, K) \in \mathbb{R}_+$ for the mentioned motion is obtained by summing the values $\mathbf{c}(x, y, \{z\}) \in \mathbb{R}_+$ for $z \in K$. Each single value $\mathbf{c}(x, y, \{u\})$, for some $u \in K$, has, in the "regular" case, the following form:

$$\mathbf{c}(x, y, \{u\}) = \gamma_u \int_0^T \frac{1}{(\rho(u, w_t))^2} dt. \tag{4.1}$$

Here ρ denotes the Euclidean distance in X; $\gamma_u \in \mathbb{R}_+, \gamma_u \neq 0$, is the coefficient that determines the intensity of the source $u \in K$; the travel time T is uniquely determined by the distance $\rho(x, y)$ given a travel speed (the latter is fixed); and

$$t \longmapsto w_t : [0, T] \to X$$

is the specific (rectilinear, in our case) trajectory of the motion. The "regularity" mentioned in discussion of the use of (4.1) has the following sense: the point u is assumed to not to belong to the interval $[x; y] \overset{\triangle}{=} \{\alpha x + (1 - \alpha)y : \alpha \in [0, 1]\}$. In absence of this regularity, the cost of travel from x to y is defined as a sufficiently large penalty constant. In definitions of the interior jobs, we follow the convention [11, Section 6] in determining the dose incurred by the agent during the local motion from the entry point (into the near zone of the radiation source; a megalopolis is its discretization) to the source itself; it is assumed that during the subsequent return travel to the exit point (from the near zone) there is no radiation from this source since it has been dismantled. This scheme is described in detail in [19, Section 4].

We use construction of [20, Section 6] for $\mathbf{c}, c_1, \ldots, c_N$. In connection with the construction of $\mathbf{c}$ we use [20, (6.17), (6.20), (6.39)]. For construction of $c_1, \ldots, c_N$, we use [20, (6.39)]. Of course, in [20, (6.17), 6.20), (6.39)], impact of the single source is considered. The function f is supposed identically equal zero. We recall also constructions of [21].

5. Software implementation and computational experiment

In this section, we describe the practical implementation of the procedure that constructs the Bellman function layers through independent computations by computational nodes and optimizes the starting point. Let us start by considering the implementation of the independent computations scheme. Assume that each layer connected with $K \in \mathfrak{C}_{N-1}$ is processed by several computational cores that share RAM. These cores together are called a computational node of the cluster; the latter is thus a union of computational nodes.

Data storage. Let us consider data storage on a single node of the cluster. For every set of objective points K, we may have to store the shortest paths set (SPS) (that is, the Bellman function layers) that pass through this set. Every shortest path in SPS differs from other paths in this SPS by its starting point and is the shortest among all other paths with the same starting point. An SPS may theoretically have as many paths as the cardinality of the corresponding set K, however, normally, there are less since not every point of this set can be initial in view of precedence constraints. In accordance with this, we store SPS in a hash table, with the aim of decreasing memory usage compared with an array. The key of the hash table is the bit mask of the set, where, if a bit at position i is set, then the point i is present in this set.

Main algorithm. The main node reads the input data from a file, which describes the objective sets that must be visited, the set of starting points, and the address pairs (precedence constraints). Next, the main node constructs the family

$$\mathfrak{C}_{N-1} = \{\overline{1,N} \setminus \{t\} : t \in \mathbf{I}(\overline{1,N})\},$$

every element of which is a cardinality $N-1$ set. The sets $K \in \mathfrak{C}_{N-1}$ are distributed between the nodes through the MPI protocol. Every node has k computational cores with shared RAM. At the node connected with the set $K \in \mathfrak{C}_{N-1}$, the Bellman function layers are shared between the cores in a uniform way. There is no exchange of data between the cores because the RAM is shared by all of them; the fragments of state space and the Bellman function layers are distributed between the cores of a single node through the OpenMP library. Then, in a parallel mode that conforms to the theoretical scheme [17, 18], which was implemented in [11] (see also [9, 10] for one-element megalopolises), the layers $v_1, \ldots, v_{N-1}$ of the "whole" (suitable for all the initial states from X^0) of the Bellman function are computed. After that, a relatively simple optimization procedure for

$$v_N(x, \overline{1,N}),$$

$x \in X^0$, is conducted, in the spirit of (3.5). This yields the grobal extremum $\mathbb{V}$ and the point $x^0 \in X^0$ with the property

$$V[x^0] = \mathbb{V}. \tag{5.1}$$

When an optimal solution in the form of a route-track pair is required, in addition to v_{N-1}, it is necessary to store all the Bellman function layers, which were determined by the main algorithm (when only the global extremum $\mathbb{V}$ and optimal initial state x^0 with property (5.1) are required, it is not necessary to store the mentioned Bellman function layers; it will suffice to have a procedure for constructing only the single layer v_{N-1} permitting the intermediary layers to be overwritten). Thus, assume $v_0, v_1, \ldots, v_{N-1}$ are known. Then, the main node constructs the optimal route for the flow where $V[x^0] = \mathbb{V}$ by means of finding the local extrema in a way similar to [7, 16, 19].

Computational experiment. In this section, we describe the solution of the routing problem on plane on the Uran supercomputer. The travel cost function is assumed to depend on the set of pending tasks; it is determined through relations similar to (4.1) (see also [20]); the function f is assumed to be zero since after all the megalopolises are visited and the corresponded interior jobs consisting of dismantling the radiating elements are conducted, the cost of return to base will be zero (since there is nothing to radiate anymore). Let us consider the case where the number of megalopolises is 48, i. e., $N = 48$ (in [11] a solution is given for the case of a significantly smaller dimension: it was assumed there $N = 30$ and $N = 31$). The megalopolises are contained inside circles on the plane. Thus, let the megalopolises, which imitate the entry and exit points to the spaces with radiation sources, be obtained by discretizing the circles (the boundaries of the near zones): on every circle, there are 30 equally spaced (in view of the angular distance). To every megalopolis, we assign a point object that imitates the radiation source in the space the megalopolis describes. The set of admissible starting points X^0 consists of 10 elements. In our example, the set $\mathbf{K}$ contains 45 address pairs that define the precedence constraints. Let this set have the following address pairs:
(38,45) (42,24) (22,7) (23,26) (32,17) (46,31) (34,8) (4,24) (17,8) (3,45) (0,26) (31,7) (3,20) (2,28) (18,47) (5,40) (36,25) (20,9) (7,6) (47,32) (46,40) (28,8) (33,5) (26,5) (0,34) (43,35) (9,27) (1,2) (1,37) (0,31) (7,23) (23,28) (39,31) (24,29) (17,45) (44,6) (29,11) (32,25) (2,14) (2,20) (15,36) (37,46) (21,10) (35,45) (12,37),

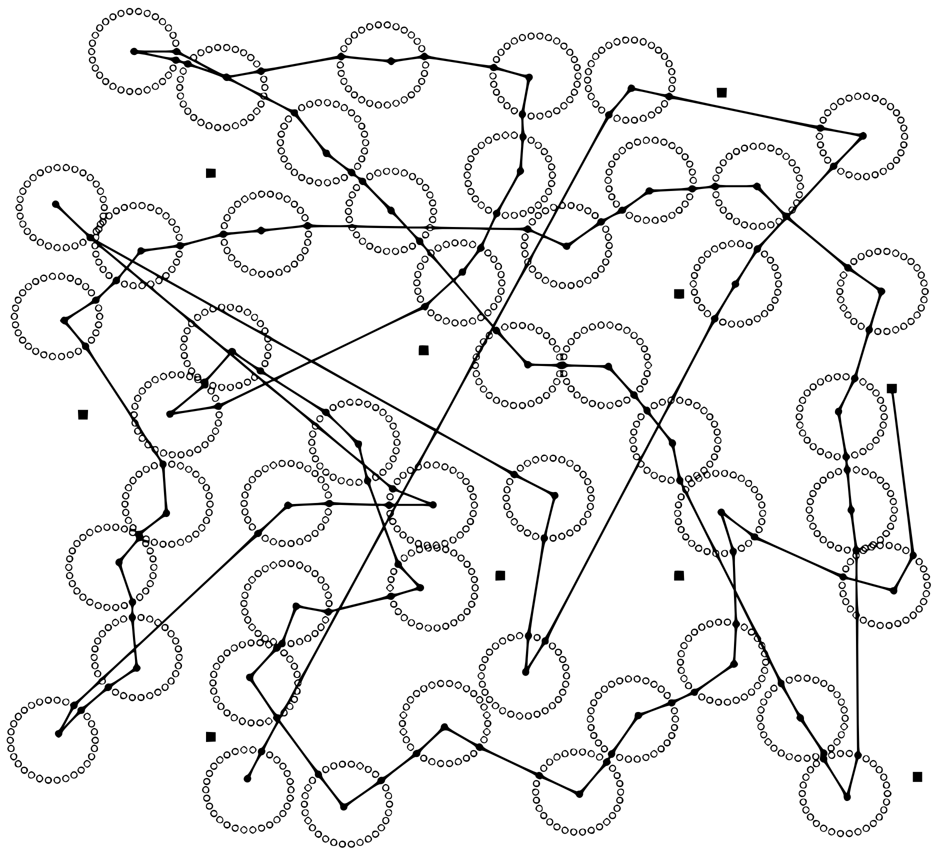
Figure 1. Route and track for visiting 48 megalopolises.

where the first argument specifies the sender and the second argument specifies the receiver. To verify the theoretical construction in practice, we implemented it in C++ for the Uran supercomputer. The program works under a 64-bit Linux operating system. The computational experiment was conducted on the nodes of the Uran cluster with the following characteristics:

two six-core Intel Xeon X5675 (3.07GHz) processors

192 GB RAM

2 × 12 MB Level 2 cache

8 Tesla M2090 GPUs (6 GB Global Memory)

400 GB local hard disk drive

The experiment used 20 cluster nodes, each of which had 12 cores. Thus, our practical implementation used 240 computational cores. The computations resulted in the starting point, route and track, see Fig. 1. The following results were obtained: $\mathbb{V} = 1.417074$ (extremum of the problem); the computation time was 15.772 seconds; the maximum RAM usage for a single computational node was 26.246 MB.

In a separate computational experiment, we considered the same problem with only 20 points per megalopolis (recall that those are viewed as exit/entry points into the facility associated with the megalopolis), equally spaced (with respect to angular distance). The following results were obtained: $\mathbb{V} = 1.4208160$ (extremum of the problem); the computation time was 13.256 seconds; the maximum RAM usage for a single computational node was 24.523 MB.

One could note that as the number of cities per megalopolis decreases, the extremum of the problem increases somewhat; this may be connected with the fact that in the second case there are fewer possible tracks, which, in its turn, makes the result worse.

In Fig. 1, the squares denote the admissible starting points. Transparent circles denote the cities in the megalopolises. Filled circles denote the entry and exit points of the megalopolises and the radiation sources inside them.

6. Computation with application of greedy algorithm

In this section we consider the solution of our basic problem by greedy algorithm similar to [33, Section 6] (in connection with construction of optimal algorithm on the base of DP, we note [7, 16, 19]). Now, we note only brief scheme of the clear greedy algorithm.

Namely, we fix $x^0 \in X^0$, suppose $\mathbf{z}^{(0)} \triangleq (x^0, x^0)$, and consider the problem

$$\mathbf{c}(x^0, \mathrm{pr}_1(z), \overline{1,N}) + c_j(z, \overline{1,N}) \longrightarrow \min, j \in \mathbf{I}(\overline{1,N}), z \in \mathbb{M}_j. \tag{6.1}$$

Now, we choose $\mathbf{j}_1 \in \mathbf{I}(\overline{1,N})$ and $\mathbf{z}^{(1)} \in \mathbb{M}_{\mathbf{j}_1}$ for which

$$\begin{gathered}\mathbf{c}(x^0, \mathrm{pr}_1(\mathbf{z}^{(1)}), \overline{1,N}) + c_{\mathbf{j}_1}(\mathbf{z}^{(1)}, \overline{1,N}) = \\ = \min_{j \in \mathbf{I}(\overline{1,N})} \min_{z \in \mathbb{M}_j} \big[\mathbf{c}\big(x^0, \mathrm{pr}_1(z), \overline{1,N}\big) + c_j(z, \overline{1,N})\big].\end{gathered} \tag{6.2}$$

Then, we obtain that

$$(\mathrm{pr}_2(\mathbf{z}^{(1)}), \overline{1,N} \setminus \{\mathbf{j}_1\}) \in D_{N-1}$$

Now, we have the above-mentioned position. Consider the problem

$$\mathbf{c}(\mathrm{pr}_2(\mathbf{z}^{(1)}), \mathrm{pr}_1(z), \overline{1,N} \setminus \{\mathbf{j}_1\}) + c_j(z, \overline{1,N} \setminus \{\mathbf{j}_1\}) \longrightarrow \min, j \in \mathbf{I}(\overline{1,N} \setminus \{\mathbf{j}_1\}), z \in \mathbb{M}_j.$$

We choose $\mathbf{j}_2 \in \mathbf{I}(\overline{1,N} \setminus \{\mathbf{j}_1\})$ and $\mathbf{z}^{(2)} \in \mathbb{M}_{\mathbf{j}_2}$ for which

$$\begin{gathered}\mathbf{c}(\mathrm{pr}_2(\mathbf{z}^{(1)}), \mathrm{pr}_1(\mathbf{z}^{(2)}), \overline{1,N} \setminus \{\mathbf{j}_1\}) + c_{\mathbf{j}_2}(\mathbf{z}^{(2)}, \overline{1,N} \setminus \{\mathbf{j}_1\}) = \\ = \min_{j \in \mathbf{I}(\overline{1,N} \setminus \{\mathbf{j}_1\})} \min_{z \in \mathbb{M}_j} \big[\mathbf{c}\big(\mathrm{pr}_2(\mathbf{z}^{(1)}), \mathrm{pr}_1(z), \overline{1,N} \setminus \{\mathbf{j}_1\}\big) + c_j(z, \overline{1,N} \setminus \{\mathbf{j}_1\})\big].\end{gathered} \tag{6.3}$$

Then, we obtain the next inclusion

$$(\mathrm{pr}_2(\mathbf{z}^{(2)}), \overline{1,N} \setminus \{\mathbf{j}_1; \mathbf{j}_2\}) \in D_{N-2}$$

The further construction are realized similar to (6.2) and (6.3) up to exhaustion of all list $\overline{1,N}$. We obtain two next finite processions

$$(\mathbf{j}_k)_{k \in \overline{1,N}} : \overline{1,N} \longrightarrow \overline{1,N},$$

$$(\mathbf{z}^{(k)})_{k \in \overline{0,N}} : \overline{0,N} \longrightarrow \mathbb{X} \times \mathbb{X}.$$

In addition, $\mathbf{i}[x^0] \triangleq (\mathbf{j}_k)_{k \in \overline{1,N}} \in \mathbf{A}$ and $(\mathbf{z}^{(k)})_{k \in \overline{0,N}} \in \mathcal{Z}_{\mathbf{i}[x^0]}[x^0]$. Of course, the value

$$\mathfrak{C}_{\mathbf{i}[x^0]}[(\mathbf{z}^{(k)})_{k \in \overline{0,N}}] \in \mathbb{R}_+ \tag{6.4}$$

corresponds to our initial state $x^0 \in X^0$. Therefore, we introduce designation

$$w[x^0] \triangleq \mathfrak{C}_{\mathbf{i}[x^0]}[(\mathbf{z}^{(k)})_{k \in \overline{0,N}}].$$

The analogous constructions are realized for all $x \in X^0$. As a result, we obtain values

$$w[x], x \in X^0.$$

We choose $x_0 \in X^0$ by the rule

$$w[x_0] = \min_{x \in X^0} w[x] \tag{6.5}$$

We consider (6.5) as upper estimate for $\mathbb{V}$ and use $\mathbf{i}[x_0]$ as the solution corresponding to greedy algorithm.

Consider a variant of computation. We preserve parameters of Section 6: $N = 48$, $|M_j| = 30$, $|\mathbf{K}| = 45$ (concrete address pairs are indicated in Section 5), $|X^0| = 10$. Under computation with employment of greedy algorithm, the result value

$$\min_{x \in X^0} w[x] = 1.884528 \tag{6.6}$$

was obtained. The minimizing point x^0 (see (6.5)) coincides with (103.12; 5.06). In this connection, we recall that, for optimal solution (see Section 5), we have $x^0 = (100.00; 53.26)$, for best initial state. In addition, for extremes realized by optimal and greedy algorithms, we obtain the following ratio: global extremum achievable by the DP procedure improves the value (6.6) about 25%. Of course, time of computing under employment of greedy algorithm about 173 seq. (recall that analogous time for optimal algorithm is 15772 seq.). We note that our greedy algorithm can be used for solving of problems having big detention. This algorithm was used in problem with 254 megalopolises and $|\mathbf{K}| = 45$. In this case, the value (6.5) and point x^0 were obtained during 1687 seq (most of this time was spent on calculating the cost function).

7. Conclusion

In this paper, we consider the issues related to the solving a routing problem with precedence constraints and complicated travel cost functions aimed at applications connected with conducting a sequence of actions in a high-radiation area. However, similar problem statements are also present in other applications. For example, in particular, a "more complex" general statement can be used to solve a problem connected with CNC plate cutting machines; see, in particular, [22–28]. A comparison with the latter is natural: both statements are very much oriented towards the practice and conduct routing with "interior" tasks. Travel cost functions' dependence on the set of pending tasks can be connected with the need to account for various constraints of dynamic character (see, [16]), specifically, a system of penalties. In this problem, the starting point is normally known in advance — if we consider the engineering problems connected with nesting; however, thinking in perspective, it may be worthwhile to consider statement (2.10) as a way of tackling the problem of choosing the initial state of the tool. This may be of importance in view of the characteristic constraints (rigidity of the whole plate and each item). In connection with TSP and TSP-like problems, let us note [29] and [30] concerned with two versions of dynamic programming and [31], which deals with the branch-and-bound method. In connection with construction of production-oriented heuristics, note [32]. However, it appears that real-life problems connected with routing have many specific issues and peculiarities, and must thus be treated with special methods (first and foremost, special heuristics); in this paper, we have endeavored to construct some.

REFERENCES

1. Melamed I. I., Sergeev S. I., Sigal I. The traveling salesman problem. Issues in theory. *Autom. Remote Control*, 1989. Vol. 50, No. 9. P. 1147–1173.
2. Melamed I. I., Sergeev S. I., Sigal I. The traveling salesman problem. Exact methods. *Autom. Remote Control*, 1989. Vol. 50, No. 10. P. 1303–1324.
3. Melamed I. I., Sergeev S. I., Sigal I. The traveling salesman problem. Approximate algorithms. *Autom. Remote Control*, 1989. Vol. 50, No. 11. P. 1459–1479.

4. Gutin G., Punnen A. P. *The Traveling Salesman Problem and Its Variations.* New York: Springer, 2002. DOI: 10.1007/b101971
5. Cook W. J. *In Pursuit of the Traveling Salesman. Mathematics at the Limits of Computation.* New Jersey: Princeton University Press, 2012. p. 248.
6. Gimadi E. Kh., Khachai M. Yu. *Ekstremalnye zadachi na mnozhestvax perestanovok* [Extremal Problems on Sets of Permutations], Yekaterinburg: UMC UPI, 2016. p. 220 (in Russian)
7. Chentsov A. G., Chentsov A. A. Route problem with constraints depending on a list of tasks. *Doklady Mathematics*, 2015. Vol. 92, No. 3. P. 685–688. DOI: 10.1134/S1064562415060083
8. Chentsov A. G., Chentsov A. A. A discrete-continuous routing problem with precedence conditions. *Proc. Steklov Inst. Math.*, 2018. Vol. 300, No. 1. P. 56–71. DOI: 10.1134/S0081543818020074
9. Chentsov A. G., Grigoryev A. M. Dynamic Programming Method in a Routing Problem: a Scheme of Independent Computations. *Mekhatronika, Avtomatizatsiya, Upravlenie*, 2016. Vol. 17, No. 12. P. 834–846. DOI: 10.17587/mau.17.834-846 (in Russian)
10. Chentsov A. G., Grigoryev A. M. A scheme of independent calculations in a precedence constrained routing problem. *Lecture Notes in Computer Science*, Vol. 9869: Intern. Conf. on Discrete Optimization and Operations Research (DOOR–2016), 2016. P. 121–135. DOI: 10.1007/978-3-319-44914-2_10
11. Chentsov A. G., Grigoryev A. M., Chentsov A. A. Decommissioning of nuclear facilities: minimum accumulated radiation dose routing problem. *CEUR-WS Proc.*, Vol. 1987: 8th Intern. Conf. on Optimization and Applications (OPTIMA–2017), 2017. P. 123–130. http://ceur-ws.org/Vol-1987/paper19.pdf
12. Chentsov A. G., Khachai M. Y., Khachai D. M. An exact algorithm with linear complexity for a problem of visiting megalopolises. *Proc. Steklov Inst. Math.*, 2016. Vol. 295, supp. 1. P. 38–46. DOI: 10.1134/S0081543816090054
13. Korobkin V. V., Sesekin A. N., Tashlykov O. L., and Chentsov A. G. *Metody marshrutizacii i ih prilozheniya v zadachah povysheniya ehffektivnosti i bezopasnosti ehkspluatacii atomnyh stancij* [Methods of Routing with Application to the Problems of Safety Enhancement and Operational Effectiveness of Nuclear Power Plants], Ed. I.A. Kalyaev. Moscow: Novye Tekhnologii, 2012. (in Russian)
14. Kuratowski K., Mostowski A. *Set Theory.* Amsterdam: North-Holland Publishing Company, 1968. p. 417.
15. Dieudonné J. *Foundations of Modern Analysis.* New York: Academic Press, 1969. 407 p.
16. Chentsov A. G., Chentsov P. A. Routing under constraints: Problem of visit to megalopolises. *Autom. Remote Control*, 2016. Vol. 77, No. 11. P. 1957–1974. DOI: 10.1134/S0005117916110060
17. Chentsov A. G. On a parallel procedure for constructing the Bellman function in the generalized problem of courier with internal jobs. *Autom. Remote Control*, 2012. Vol. 73, No. 3. P. 532–546. DOI: 10.1134/S0005117912030113
18. Chentsov A. G. A parallel procedure of constructing Bellman function in the generalized courier problem with interior works. *Vestnik YuUrGU. Ser. Mat. Model. Progr.*, 2012. No. 18. P. 53–76. (in Russian)
19. Chentsov A. A., Chentsov A. G., and Chentsov P. A. Elements of Dynamic Programming in the Extremal Problems of Routing. *Autom. Remote Control*, 2014. Vol. 75, No. 3. P. 537–550 DOI: 10.1134/S0005117914030102
20. Chentsov A. G., Chentsov A. A. A model variant of the problem about radiation sources utilization (iterations based on optimization insertions). *Izv. Inst. Mat. Inform. Udmurt. Gos. Univ.*, 2017. Vol. 50. P. 83–109. DOI: 10.20537/2226-3594-2017-50-08 (in Russian)
21. Chentsov A. G., Chentsov A. A., Grigoryev A. M. On one routing problem modeling movement in radiation fields. *Vestn. Udmurtsk. Univ. Mat. Mekh. Komp. Nauki*, 2017. Vol. 27, No. 4. P. 540–557. DOI: 10.20537/vm170405 (in Russian)
22. Petunin A.A. About some strategy of formation of a route of the cutting tool by development of the controlling programs for the thermal sheet cutting machines. *The UGATU Bulletin. Series: Control, ADP equipment and informatics*, 2009. Vol. 13, No. 2 (35). P. 280–286.
23. Frolovskii V.D. Computer-aided design of the control programs for thermal metal cutting on NPC machines. *Informacionnye tekhnologii v proektirovanii i proizvodstve*, 2005. No. 4. P. 63–66.
24. Wang G.G. and Xie S.Q. Optimal process planning for a combined punch-and-laser cutting machine using ant colony optimization. *Int. J. Product. Res.*, 2005. Vol. 43, No. 11. P. 2195–2216. DOI: 10.1080/00207540500070376

25. Lee M.-K. and Kwon K.-B. Cutting path optimization in NC cutting processes using a two-step genetic algorithm. *Int. J. Product. Res.*, 2006. Vol. 44, No. 24. P. 5307–5326. DOI: 10.1080/00207540600579615
26. Jing Y. and Zhige C. An optimized algorithm of numerical cutting-path control in garment manufacturing. *Adv. Mater. Res.*, 2013. Vol. 796. P. 454–457. DOI: 10.4028/www.scientific.net/AMR.796.454
27. Ganelina N.D. and Frolovskii V.D. On constructing the shortest circuits on a set ofline segments. *Sib. Zh. Vychisl. Mat.* [Siberian J. of Numer. Mathematics], 2006. Vol. 9, No. 3. P. 241–252.
28. Verkhoturov M.A. and Tarasenko P.Yu. Software for the problems of optmization of the cutting tool path for planar figure cutting on the basis of chain cutting. *Vestn. UGATU*, 2008. Vol. 10, No. 2 (27). P. 123–130. http://journal.ugatu.ac.ru/index.php/Vestnik/article/view/1274/1103
29. Bellman R. Dynamic programming treatment of the travelling salesman problem. *J. ACM.* 1962. Vol. 9, No. 1. P. 61–63. DOI: 10.1145/321105.321111
30. Held M. and Karp R.M. A dynamic programming approach to sequencing problems. *J. Soc. Ind. Appl. Math.*, 1962. No. 10 (1). P. 196–210. DOI: 10.1137/0110015
31. Little J. D. C., Murti K. G., Sweeney D. W., and Karel C. Algorithm for the traveling salesman problem. *Econ. Mat. Metod.*, 1965. Vol. 1, No. 1. P. 94–107.
32. Escudero L. An inexact algorithm for the sequential ordering problem. *Eur. J. Oper. Res.*, 1988. Vol. 37, No. 2. P. 236–249.
33. Chentsov A.G., Chentsov A.A., Chentsov P.A. Extremal routing problem with internal losses. *Proc. Steklov Inst. Math.* 2009. Vol. 264, suppl. 1. P. 87–106. DOI: 10.1134/S0081543809050071