A Method for Synthesis of Specialized Processors to Support On-board Scheduling of Periodic Tasks

Vyacheslav Mikhailovich Khachumov and Mikhail Vyacheslavovich Khachumov

Department of Information Technologies
Peoples’ Friendship University of Russia (RUDN University)
6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation
{khachumov_mv, khachumov_vm}@rudn.university

Abstract — We consider the actual problem of synthesizing specialized computing devices (SCDs) for periodic task scheduling to support autonomous functioning of a small UAV having limited computing resources. A specialized processor implements a given system of concurrent local algorithms. The problems of minimizing the number of processing units (PUs) in SCDs and minimizing the number of inter-processor connections are solved. We consider the issues of designing a control unit for switching PUs to optimize periodic computational process. The solution to the problem of partitioning a connection graph of PUs into weakly coupled fragments is proposed. To determine the expediency of partitioning optimization we propose an efficiency function.

Keywords — UAV; specialized processor; periodic scheduling; local algorithms; connection graph.

I. INTRODUCTION

Periodic scheduling is often required for real-time data processing that takes place in navigation and control systems of UAVs, when an onboard computing device repeatedly executes given algorithms as a response to receiving external signals.

A significant contribution to the study of scheduling periodic real-time jobs was made in early papers [1-3]; however, this problem has not been fully investigated and is still of great importance. In papers [4, 5] the authors proposed an algorithm for planning admissible periodic schedules in a real-time multiprocessor system that receives requests to perform jobs in specified periods. A new kind of constraints that could be easily added to the linear programming problem for constructing a completely cyclic schedule was introduced in [6]. The authors considered a loop on a special graph as a scheduling model. Periodic operation scheduling for a system consisting of a set of one-type processors is considered in paper [7]. The authors of paper [8] consider a pipeline as a computational model for scheduling applications, such as signal/image processing. Real-time requirements for such applications include job timing, and latency. A simple but effective method for periodic scheduling based on acyclic graphs is proposed.

This paper extends research on periodic scheduling and synthesis on this basis of specialized computing devices (SCDs) to process data on board the UAV [9, 10]. We propose that SCDs should serve as a computing platform to solve complex tasks in real time, including analysis and processing of navigation data; synthesizing controls based on airborne sensor data, topographic data and geographical landmarks; environmental monitoring and mapping; target and trajectory tracking. Due to improvement and optimization of software and hardware computing facilities, synthesized SCDs are aimed to provide real-time processing with the required reliability in the conditions of a constant increase in the number of tasks and limited onboard computing resources.

The purpose of this study is to develop methods for the transition from a priori known systems of local algorithms (LAs) of periodic data processing, considered in [10] to the structures of specialized computing devices implementing these algorithms and to develop a software structure for automation design of schedules and SCDs structures that combine processing cycles.

II. BASIC CONCEPTS AND DEFINITIONS

To describe computational processes, we introduce an identity operator $e$ with the following properties: for any operations $a_i, b_j$ of local algorithms $A = a_1, a_2, \ldots, a_m$, and $B = b_1, b_2, \ldots, b_n$, $1 \leq m, n \leq M$ takes place $a_i \equiv eb_i \equiv a_i$, $b_i \equiv eb_j \equiv b_j$. It is assumed that any number of processing units (PUs) can implement operator $e$. An operator of the form $(cd)$, that implies simultaneous execution of operations $c$ and $d$, we call a compound operator. We define the following systems for local algorithms $A$ and $B$:

$$c_{A_1} = (ae) \ldots (ae)(a_{i+1}b_i) \ldots (a_{m+1}b_m) \ldots (eb_{m-k}) \ldots (eb_k),$$
where $m - n + 1 \leq k \leq m - 1$;

$$c_{A_2} = (ae) \ldots (ae)(a_{i+1}b_i) \ldots (a_{m+1}b_m)(a_{m-k}e) \ldots (a_ne),$$
where $0 \leq k \leq m - n$;

$$c_{A_3} = (eb) \ldots (eb) (a_{i+1}b_i) (a_{m-k}b_m)(a_{m-k}e) \ldots (a_ne),$$
where $1 - n \leq k \leq -1$.

LAs system $A_1$ is easily expanded when the number of states increases. When $k$ is fixed, LAs system is equivalent to a diagram of a single shift of algorithms $A$ and $B$, and composite operators - to state vectors, which determine a set of combined operations. The meaning of operator $e$ is to delay a process for one cycle. The delay can lead to a change in the phase shifts between monotone operations and, therefore, affect the quality of the schedule. Schedules for a fixed LAs system are equivalent to the corresponding schedules of local algorithms included in the system. The cyclic schedule of a system is equivalent to the cyclic schedules of local algorithms $A$ and $B$ of the same period. The value $k$ is called a shift of the system of local algorithms $A, B$ and is equal to the number of steps by...
which the beginning of the cycle of algorithm \( A \) outpaces (if \( k > 0 \)) or lags (if \( k < 0 \)) the beginning of the cycle of algorithm \( B \).

Let us consider the concept of fast scheduling. An average execution time of LAs system \( t(cA^*) \) is regarded as quality criteria. Let the model of a computing device be represented by a set of \( Q \) processing units \( \{q_1,q_2,...,q_l\} \) of \( l \) types which are functionally specialized in accordance with a set of local operations. Here \( q_j \) is a set of \( k_j \) identical PUs,

\[
\sum \limits_{j=1}^{l} k_j = Q, \quad 1 \leq k_j \leq p_j.
\]

Let \( t_i \) be the time required for the processing unit from \( q_j \) to complete the operation.

The transition from the structure of the algorithm to the structure of SCD is to establish a correspondence between the local operations and the set of PUs. Each operation of the structure of SCD is to establish a correspondence between the type of the \( i \)-th type is associated with a single PU from the set of the same type. If \( k_j = p_j, \forall j \in \{1,...,l\} \) is met, then there is a single way to assign PUs to LAs operations and each of them is used one time. Such a structure is called a structure with permanent links. Usually \( k_j \leq p_j \) is met and there are many ways to assign PU, characterized by multiple use of elements. Such structures are called structures with dynamic connections and use a commutator to switch the inputs of PUs. To solve the problem we optimally adjust the structure for a given algorithm and the generalized model of PUs with dynamic constraints, shown in Fig. 1, actually use a tunable structure.

![Fig. 1. The architecture of a CSD with dynamic links](image)

Here \( (x_i, y_i, z_i) \) is a set of signals of \( i \)-th PU, \( i=1,...,Q \). When solving the problem of transition from the system of algorithms to the structure of SCD, one can use the most general quality indicators determining technical and economic characteristics of the device. A commutator provides simultaneous access to the necessary operands.

III. FORMULATION OF THE PROBLEM OF SCD SYNTHESIS

The choice of quality criteria and optimal transition to the structures of specialized computing devices that implement a given LAs system can be characterized by a vector \( X = (x_1,x_2,...,x_n) \) reflecting the level of compliance of an SCD with its functional purpose. Currently, such indicators as performance, reliability, throughput, number of PUs, performance time, cost, energy consumption, number of PUs connections, as well as their various combinations, are used to assess the quality of an SCD. Such a relatively large number of indicators is needed due to the variety of requirements for specialized processors depending on tasks specifics. There are two types of problems: scalar, when one indicator should be minimized, and multi-criteria, when optimization is performed with several indicators. Often multi-criteria problem is very complex and one should reduce it to a scalar optimization problem. The most widely used method for reducing vector criteria to scalar is the additive convolution \( F(x) = \sum \limits_{i=1}^{n} \lambda_i x_i, \quad 0 \leq \lambda_i \leq 1, \quad \sum \limits_{i=1}^{n} \lambda_i = 1 \), called the generalized quality criterion \([11]\), where \( \lambda_i \) are the weights determined by expert assessments.

A local algorithm \( A^* \) where PUs are assigned to operations we call the marked local algorithm (MLA). Accordingly, we denote MLA system as \( cA^* \). As the complexity measure of LAs system execution we mean the total number of PUs \( Q(cA^*) \) used to organize the required computational process. As computational speed we assume an average time of a single execution of LAs system \( t(cA^*) \) or the time of its \( k \)-multiple execution in a specialized computing device. The reliability index \( P(cA^*) \) is the probability of failure-free operation of the specialized computing device at \( k \)-multiple execution of LAs.

Let us formulate the problem of transition from the structure of local algorithms to the structure of SCD in accordance with the selected quality criteria. The proposed approach is based on the consideration of a series of interrelated tasks of practical interest. Optimization consists of several stages. At the first stage, the problem \( Q(cA^*) \rightarrow \min \) is solved with restrictions imposed on the parameters \( t(cA^*), P(cA^*) \). When solving the problem one can obtain several variants, in which the parameter being optimized takes the same values, and all other parameters are within certain limits.

Thereby, at the second stage, we propose to optimize the number of PUs links \( L(cA^*) \), since it reflects such CSD characteristics as commutator complexity and operation complexity. We solve the problem \( L(cA^*) \rightarrow \min \) with the fixed values of parameters \( Q(cA^*), t(cA^*), P(cA^*) \).

Finally, at the third stage, we solve the problem of optimal designing of PUs switching control device for a periodic scheduling that combine processing cycles.

IV. ALGORITHMS FOR MINIMIZING THE NUMBER OF PROCESSING UNITS

We consider the problem of minimizing the number of PUs included in CSD to implement a given system of local algorithms focused on concurrency of objects.

**Definition 1.** We say that LAs system meets the requirements for concurrency if the following conditions are satisfied:

- LAs of the system are identical, i.e. contains the same operations performed in the same order;
- LA are performed cyclically with the same period;
- There is no informational relationship between the operations of different LAs.
MLAs system $cA^*$ meets the requirements for concurrency, if in addition to conditions listed above, the following ones are also met:

- Each MLA uses a set of PUs that do not intersect with the sets of PUs of other MLAs;
- An order of assignment PUs to the phases of LAs is the same for all marked LAs.

Let a specialized device in order to serve $n$ objects implements a system of identical LAs containing $m$ operations. Let we compose $Q$ processor elements, i.e. values $k_i$, $i=1,...,l$ are known. The task is to design a cyclic schedule minimizing the value $t(cA^*)$ that fulfill the above requirements and use all PUs. To solve this problem one need to distribute PUs optimally through processing channels the number $d$ of which is not known a priori. A system of local algorithms divides into $d$ subsystems in accordance with the number of selected channels and it reasonably to find a solution for one of these subsystems.

Let the boundary values of the average execution time of LAs system $t_d(cA^*)$ and the reliability of processing channel $t_0(cA^*)$ be given. The reliability of a processing channel is determined by $P(cA^*) = e^{-\lambda t}$, where $\lambda$ is the PU failure rate, $\mu$ is the total number of PUs for one channel, $t$ is the time of $k$-multiple execution of LA within the channel, if $k \rightarrow \infty, t \rightarrow kt(A*, q)$. We formulate the problem of the first stage of transition from LA system to the structure of specialized device as follows.

**Problem 1**

$$Q(cA^*) = dq \rightarrow \min$$

under restrictions

$$t(A^*, q) \left[ \frac{n}{d} \right] \leq t_0(cA^*), \quad (3)$$

$$t(A^*, q) \leq \frac{-\ln P(cA^*)}{\lambda q}, \quad (4)$$

$$1 \leq d \leq n, \quad (5)$$

$$1 \leq q \leq m, \quad (6)$$

$$\left[ \frac{mn}{t_0(cA^*)} \right] \leq Q(cA^*) \leq mn. \quad (7)$$

The following parameters are given a priori: the number $n$ of local algorithms in a system; the total number $m$ of LA operations; the number $l$ of types of LA operations; the multiplicity $k$ of the system execution. Variables to be obtained are: the number $d$ of processing channels in the device; the number $q$ of PUs in the channel; the average time $t(A^*, q)$ to get a solution for a channel containing $q$ PUs.

To solve problem (2)-(7) we propose the following algorithm.

**Algorithm 1**

1. Select a value $Q(cA^*)$ in accordance with the lower boundary given in (7), go to step 3.
2. Assign a value $Q(cA^*) +1$ to $Q(cA^*)$.

3. In accordance with constraints (5)-(6) form lists $d$, $\varphi(d) = \left[ \frac{n}{d} \right]$,

$$q = \left[ \frac{Q(cA^*)}{d} \right]$$. If the lists are empty, go to step 2.

4. For each $<d, \varphi(d), q>$ form a set of tables containing PUs assignment variants (a set of MLA).

5. For each table solve the problem of planning an optimal schedule. If we find MLA with values $t(A^*, q)$ that satisfy constraints (3)-(4), then go to step 6, otherwise to step 2.

6. Determine $Q(cA^*)$ for each MLA. Select MLA with the minimal $Q(cA^*)$.

7. End.

As a result, we obtain the minimal required number of PUs, optimal MLA and a schedule for its cyclic execution. Next, we consider the problem to parallelize adjacent operations.

**Definition 2.** A system of three local algorithms $A_1$, $A_2$, $A_3$, meets the requirements for concurrency of adjacent operations if the following conditions are fulfilled:

1) inputs $A_1$, $A_2$ are independent, and the outputs serve as $A_3$ inputs;

2) periodicity of all LAs is the same.

The corresponding MLAs system meets the requirement for concurrency of adjacent operations if MLA $A_3$ and $A_2$ use non-intersecting sets of PUs. That allows combining in time $A_1$ execution with $A_2$ execution.

For a cyclic implementation of LAs system realizing concurrency of adjacent operations, various methods of assigning PUs to LAs are used. The problem of minimizing the number of PUs for LAs system realizing concurrency of adjacent operations is formulated as follows:

**Problem 2**

$$Q(cA^*) = \sum_{i=1}^{n} q_i \rightarrow \min$$

under restrictions

$$\left[ \sum_{i=1}^{n} \frac{m_i}{t_0(cA^*)} \right] \leq Q(cA^*) \leq \sum_{i=1}^{n} m_i, \quad (9)$$

$$t(cA^*) \leq t_0(cA^*), \quad (10)$$

$$t(cA^*)Q(cA^*) \leq \frac{-\ln P(cA^*)}{\lambda k}, \quad (11)$$

$$l_i \leq q_i \leq m_i, \quad i = 1,...,n. \quad (12)$$

Here $n$ is the number of LAs in a system; $m_i$ is the number of operations for $i$-th LA, $l_i$ is the number of types of operations for $i$-th LA.

The solution of the problem (8)-(12) uses algorithms for constructing optimal schedules that are specific to certain LAs system.

**Algorithm 2**

1. Select $Q(cA^*)$ with account for the lower boundary condition (8), go to step 3.
2. Assign a value \(Q(cA^*) + 1\) to \(Q(cA^*)\).
3. Find PUs assignment variants with account for condition (12).
4. Solve the problem of optimal assignment of PUs. If one obtain MLAs that satisfy conditions (9)-(10) for \(i(cA^*)\), then go to step 5, otherwise to step 2.
5. The end.

As an optimization result, we can obtain more than one variant that satisfy all the imposed constraints. In this case, it is necessary to minimize the number of inter-processor links of the specialized device.

V. MINIMIZING THE NUMBER OF PUS LINKS TO SYNTHESIZE A SPECIALIZED DEVICE

We use the matrix form of the linking scheme. The existence of information transferring from the \(i\)-th to \(j\)-th operation is described by \(a_{ij}\), where \(a_{ij}\) is an element of the matrix of links \(A\).

**Definition 3.** Communication links of a specialized computing device that map a set of PUs outputs to PUs inputs, we call static inter-processor connections.

Static inter-processor links can be conveniently represented by a directed graph \(V\), the vertices of which correspond to PUs, and arcs to information links. Such a model allows us to formalize counting the number of static links that determine complexity of commutator structure and select best solution for the first stage.

Consider the issues of designing a control unit (CU) for switching PUs to optimize periodic computational process.

**Definition 4.** Links that map a set of PUs outputs to PUs inputs at discrete time points corresponding to the cycles of periodic scheduling, we call dynamic inter-processor links.

Dynamic inter-processor links obtained at each step we represent in the form of a bipartite graph \(U_i\). The vertices of the first and second layers of \(U_i\) are respectively PUs information inputs and outputs. The vertices are oriented from the second layer to the first one and set in accordance with inter-processor structure obtained on the \(i\)-th step. By combining the graphs, we find a generalized bipartite graph.

An optimal control system (CS) design is essential for CSDs synthesis. Parallel control buses carries \(n\)-bit binary words to exchange information between PUs using special CS commands. At that \(y_i(t)=1\) at time points that coincide with steps (cycles) \(i(0+T),...,i(kT)\) of the cyclic process with \(T\) processing period, and \(y_i(t)=0\) at all other time points. The problem is reduced to transferring one-bit information between PUs registers. We describe the control system operation in terms of logic functions. We assign each term \(J_r\) from \(J_r\) to a vertex of graph \(E\). We use two kinds of edges, one to connect variables with same name and the other to connect terms belonging to the same function. The problem of designing optimal CS using standard logic elements can be reduced to the problem of partitioning the graph into subgraphs while minimizing links between subgraphs taking into account restriction imposed on the number of vertices in a subgraph. By partitioning, one can obtain independent blocks of algorithms and programs for their parallel execution.

Let us study the problem of splitting CS connection graph into weakly coupled fragments. To solve this problem, we consider the method of constructing a tuple of vertices with a minimum total length of edges [12] and the method of sequential transformation of the adjacency matrix, presented in [13]. Let the original algorithm be represented as a graph \(Q=(V, U)\), where \(V=\{v_1,...,v_n\}\) is a set of vertices; \(U=\{u_{i1},...,u_{in}\}\) is a set of edges. Vertices correspond to operations and edges to control links. Consider the problem of splitting a graph into \(k\) \((k\leq n)\) parts. We define the length of an edge as a non-negative subtraction of vertices index numbers incident to this edge. For a complete graph, the total length of its edges is \(L_n = \sum_{i=1}^{n} (n - 1)\), where \(N=n(n-1)/2\) is the number of edges. In the general case \(|U|<N\), the total length of edges depends on vertices order and can serve as a criterion for constructing an algorithm for partitioning a graph into subgraphs.

We give a description of the graph in the form of an adjacency matrix (Table 1).

<table>
<thead>
<tr>
<th>vertices</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

We construct a tuple of a graph with an initial order of vertices, as shown in Fig. 2.

![Fig 2](image)

The total length of edges here is equal to \(L_2=26\). Partitioning of such a graph into two parts with an approximately equal number of vertices \(|V_1|=4, \ |V_2|=3\) (dotted line in the figure) corresponds to a quality indicator \(|U_p|\)=5 (the number of connections between subgraphs).

Along with this one, we consider another tuple (Fig. 3) with a total length of edges equal to \(L_2=10\), where \(|V_1|=4, \ |V_2|=3, \ |U_p|=1\)

![Fig 3](image)

Two considered tuples correspond to the initial matrix (Table 1) and the transformed one (Table 2).
In both cases, splitting into two subgraphs corresponds to two square submatrices \( A_1 \) and \( A_2 \) (highlighted by shading). Elements that do not belong to matrices \( A_1 \) and \( A_2 \) correspond to edges connecting subgraphs.

### TABLE 2. TRANSFORMED ADJACENCY MATRIX

<table>
<thead>
<tr>
<th>vertices</th>
<th>4</th>
<th>1</th>
<th>7</th>
<th>3</th>
<th>6</th>
<th>2</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Partitioning \( V_1=\{4,1,7,3\}, \quad V_2=\{6,2,5\} \) is optimal with \( |U_1|=1 \) and can be obtained by transposition of vertices in a tuple. The algorithm of such a transposition considered in [13].

### VI. ESTIMATION OF THE EFFICIENCY OF PARTITIONING A GRAPH INTO SUBGRAPHS

For a family of graphs with \( n \) vertices and \( m \) edges, \((n-1)<m<N\), the maximum total length of edges is estimated as

\[
L_{\text{max}} = \sum_{i=1}^{i_i} (n-i) + x_1 [n-(i+1)], \quad (13)
\]

where \( i_i' = \lfloor i_i' \rfloor \) is the integer part,

\[
i_i' = \frac{1}{2} \left( 1 + \sqrt{8m+1} \right), \quad x_1 = m - (1+i_i) \frac{i_i'}{2}.
\]

The minimum total length of edges is estimated as:

\[
L_{\text{min}} = \sum_{i=1}^{i_2} (n-i) + x_2 (i_2+1), \quad (14)
\]

where

\[
i_2' = \lfloor i_2' \rfloor, \quad i_2' = \frac{2n-1}{2} - \frac{1}{2} \sqrt{(2n-1)^2-8m} + 1,
\]

\[
x_2 = m - [(n-1) + (n-i_2)] \frac{i_2}{2}.
\]

For an arbitrary graph, the condition \( L_{\text{min}} \leq L \leq L_{\text{max}} \) is satisfied. We estimate the total number of links between subgraphs after dividing a graph into \( k \) parts with approximately the same number of vertices, \( 1<k<n \). The infimum \( b_{\text{max}}(k) \) of the number of links between subgraphs is:

\[
b_{\text{max}}(k) = \sum_{i=1}^{i_i} i + \sum_{i=i_j}^{i_2} [i-k(i-j_i+1)] + [x_1 - k(i_j + j_i + 2)],
\]

where \( j_i \leq i_i \),

\[
b_{\text{max}}(k) = \sum_{i=1}^{i_i} i + (x_1 - k), \quad \text{where} \quad j_i = i_i + 1,
\]

\[
b_{\text{max}}(k) = \sum_{i=i_j}^{i_2} i + x_1, \quad \text{where} \quad j_i > i_i + 1,
\]

\[
j_i = \frac{n(k-1)}{k} + 1.
\]

If the following conditions are met:

\[
x_1 - k(i_i - j_i + 2) < 0, \quad (x_1 - k) < 0,
\]

we assume

\[
x_1 - k(i_i - j_i + 2) = 0, \quad (x_1 - k) = 0.
\]

The supremum \( b_{\text{min}}(k) \) of the number of links:

\[
b_{\text{min}}(k) = \sum_{i=1}^{i_i} [(n-i) - k(n-i)] + \sum_{i=i_j}^{i_2} (n-i) + x_2 =
\]

\[
= (k-1) \sum_{i=1}^{i_i} i + \sum_{i=j_i}^{i_2} (n-i) + x_2,
\]

where \( j_2 < i_2 \),

\[
b_{\text{min}}(k) = (k-1) \sum_{i=1}^{i_2} i + x_2, \quad \text{where} \quad j_2 = i_2,
\]

\[
b_{\text{min}}(k) = (k-1) \sum_{i=1}^{i_2} i + [x_2 - k(n-k)/(i_2+1)],
\]

where \( j_2 > i_2, j_2 = \frac{n}{k} - 1. \)

If the following conditions are met:

\[
x_1 - k(n-(i_2 + 1)) < 0, \quad [x_2 - k(n-(i_2 + 1))] < 0,
\]

we assume

\[
x_1 - k(n-(i_2 + 1)) = 0, \quad [x_2 - k(n-(i_2 + 1))] = 0.
\]

Partitioning efficiency is estimated as follows:

\[
f(k) = \frac{b_{\text{max}}(k) - b_{\text{min}}(k)}{m}, \quad 0 \leq f(k) \leq 1
\]

Examples for \( n = 12; \quad m = 26, 40, 53 \) are presented in Fig.4. Decision on the expediency of partitioning optimization in each specific case is made according to the efficiency curve.

### VII. CONCLUSION

The proposed method of designing onboard computers containing processing units with a set of operations in conjunction with software for planning periodic schedules
allows one to efficiently solve various problems emerging in UAV control and navigation systems.

We propose to use the developed models, algorithms and methods of structural optimization to solve real-time problems related to multiple (cyclic) execution of a set of tasks on a specialized computing device consisting of programmable processing units.

The publication has been prepared with the support of the “RUDN University Program 5-100” (recipient Khachumov M.V., basic concepts, formulation of the problem and mathematical model development). The reported study was funded by RFBR, project number 17-29-07003 (recipient Khachumov V.M., optimization problems solving to synthesize a specialized device). The reported study was funded by RFBR, project number 18-07-00025 (recipient Khachumov M.V., solution evaluation and numerical analysis).

REFERENCES


