Ершов Владимир Иванович – e-mail: evi-monino@mail.ru; тел.: 89690301515; к.в.н.; профессор; ведущий научный сотрудник научно-исследовательского центра пожарной техники и пожарной автоматики.

Баранник Александр Юрьевич – Всероссийский научно-исследовательский институт по проблемам гражданской обороны и чрезвычайных ситуаций МЧС России, федеральный центр науки и высоких технологий; e-mail: auba@yandex.ru; г. Москва, Россия; тел.: 89166951214; к.т.н.; с.н.с.; ведущий научный сотрудник 6 научно-исследовательского центра «Развития технических средств и технологий».

Лагутина Анна Викторовна – e-mail: anya-lagutina@yandex.ru; тел.: 89057118834; старший научный сотрудник 6 научно-исследовательского центра «Развития технических средств и технологий».

Pavlov Evgeny Vladimirovich – Federal State Budgetary Institution "All-Russian Order" Badge of Honor "Research Institute of Fire Defense of the Ministry of Emergency Situations of Russia", e-mail: pavlov-vp@mail.ru; Balashikha, Russia; phone: +79167850002; senior researcher at the Research Center for Fire Engineering and Fire Automation.

Ershov Vladimir Ivanovich – e-mail: evi-monino@mail.ru; phone: +79690301515; cand. of mil. sc.; professor; leading researcher of the Research Center for Fire Engineering and Fire Automation.

Barannik Alexander Yuryevich – All-Russian Scientific Research Institute for Civil Defense and Emergency of the EMERCOM of Russia, Federal Science and High Technologies Center; e-mail: auba@yandex.ru; Moscow, Russia; phone: +79166951214; cand. of eng. sc.; senior researcher; leading researcher of the 6th Research Center Development of technical means and technologies.

Lagutina Anna Viktorovna – e-mail: anya-lagutina@yandex.ru; phone: +79057118834; senior researcher of the 6th Research Center Development of technical means and technologies.

УДК 65.012.122 DOI 10.18522/2311-3103-2023-2-53-66

А.М. Грузликов

ОРГАНИЗАЦИЯ ВЫЧИСЛЕНИЙ И ДИАГНОСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ БОРТОВЫХ ВЫЧИСЛИТЕЛЕЙ РОБОТОТЕХНИЧЕСКИХ КОМПЛЕКСОВ*

Целью исследования является повышение эффективности высокоуровнего проектирования робототехнических комплексов в части организации вычислений и диагностирования программного обеспечения бортовых вычислителей. Рассматриваются три проблемы: назначения, планирования и диагностирования. Первая проблема, задача назначения, определение необходимых ресурсов и построения назначения программных модулей по процессорам бортовых вычислителей согласно заданному критерию. В работе дана математическая постановка задачи, приведены алгоритмы, показано наличие областей эффективного доминирования алгоритмов в зависимости от выбранного критерия. Вторая проблема, задача планирования, определение последовательности выполнения заданий в многоканальных системах согласно заданному критерию. Дана математическая постановка задачи, приведены алгоритмы и результаты их исследований. Особенностью рассматриваемых алгоритмов планирования является использование единого подхода, а именно понятия отношения доминирования между процессорами и построение разрешимых классов систем. Третья проблема, диагностирование программного обеспечения. Сложность проблемы диагностирования вычислительных систем определяется не только их высокой размерностью, но и множественностью причин возникновения нарушений. Источником нарушений вычислительного процесса

* Работа проводилась при поддержке гранта РНФ № 22-29-00339.

могут быть как отказы аппаратуры, так и ошибки в организации вычислений допущенные разработчиками. В работе используется иерархический подход, в этом случае компоненты системы, связанные отношением включения, размещаются по уровням сложности так, что модель компоненты более высокого уровня представляются композицией моделей более низкого уровня. Для каждого уровня синтезируются свои средства диагностирования, ориентированные на отказы информационных связей между компонентами предыдущего уровня. В работе предлагается подход тестового диагностирования с использованием сетединамической модели, которая предполагает введение избыточности с целью упрощения диагностического эксперимента и снижение трудоёмкости его подготовки. Данный подход позволяет автоматически синтезировать средства диагностирования и обнаруживать нарушения в адресации информационных обменов при работе программного обеспечения робототехнических комплексов по прямому назначению. Апробация рассматриваемых алгоритмов проводилась с использованием разработанного автором программного обеспечения на этапе проектирования бортовых комплексов в АО «Концерн «ЦНИИ «Электроприбор».

Высокоуровневое проектирование; организация вычислений; задача назначения; задача планирования; тестовое диагностирование.

A.M. Gruzlikov

CONTROL OF COMPUTING AND DIAGNOSTICS OF SOFTWARE FOR ON-BOARD COMPUTERS OF ROBOTIC COMPLEXES

The aim of the study is to improve the efficiency of high-level design of robotic systems in terms of management of computing and diagnostics of on-board computer software. Three problems are considered: assignment, scheduling and diagnostics. The first problem, the assignment task, is to determine the necessary resources and build the distribution of program modules among the on-board computer processors in accordance with a given criterion. The article presents a mathematical formulation of the problem, algorithms are given, and the presence of areas of effective dominance of algorithms depending on the selected criterion is shown. The second problem, the scheduling problem, is to determine the sequence of tasks in multi-channel systems in accordance with a given criterion. A mathematical formulation of the problem is given, algorithms and the results of their study are given. A feature of the scheduling algorithms under consideration is the use of a unified approach, namely the concept of the dominance relation between processors and the construction of solvable classes of systems. The third problem is software diagnostics. The complexity of the task of diagnosing computer systems is determined not only by their high dimensionality, but also by the multiplicity of causes of violations. The source of violations of the computing process can be both hardware failures and errors in the organization of calculations made by developers. The article uses a hierarchical approach, in this case, the components of the system, connected by an inclusion relation, are arranged in accordance with the level of complexity in such a way that the model of higher-level components is represented by a composition of lower-level models. For each level, own diagnostic tools are synthesized, focused on failures of information links between the components of the previous level. The article proposes an approach to test diagnostics using a network dynamic model, which involves the introduction of redundancy in order to simplify the diagnostic experiment and reduce the complexity of its preparation. This approach allows you to automatically synthesize diagnostic tools and detect violations in the addressing of information exchanges when the software of robotic systems works as intended. Approbation of the algorithms under consideration was carried out using software developed by the author at the stage of designing on-board systems at JSC «Concern «Central Research Institute «Elektropribor».

High-level design; management of computing; assignment problem; scheduling problem; test diagnostics.

Введение. Современные бортовые робототехнические комплексы обработки информации и управления среди прочих систем занимают важное место, определяемое ответственностью решаемых ими задач. Основу этих комплексов в настоящее время составляют распределенные вычислительные системы, построенные с использование многопроцессорных вычислительных модулей. В последние

годы, впечатляющие достижения микроэлектроники привели к тому, что в области проектирования бортовых вычислительных систем сложилась устойчивая тенденция, в результате которой центр тяжести основных проблем сместился в сторону разработки программного обеспечения. При этом существенно возросла значимость высокоуровневого проектирования, т.е. проектирования на уровне базовых архитектурных принципов функционирования системы [1, 2].

Создание систем охватывают широкий круг взаимосвязанных вопросов, затрагивающих практически все важнейшие направления разработки, начиная с организации вычислений в части решения вопросов назначения задач на процессоры системы и планирования вычислений, далее выбор и реализация некоторой модели управления вычислениями и, наконец, высокоуровневое диагностирование вычислительных систем. Кроме того, следует отметить, что на фоне жестких требований по точности, достоверности и надежности бортовые робототехнические комплексы должны обеспечивать решение задач большой вычислительной сложности [3–5]. Необходимость решения всех задач в реальном времени накладывает дополнительные ограничения на применяемые подходы и, в частности, ограничения, связанные с временной привязкой задач.

Ниже излагаются основные результаты исследований, проведенных в АО Концерн «ЦНИИ «Электроприбор» в этой области. В фокусе исследований находились проблемы высокоуровневого проектирования, связанные с назначением задач на процессоры бортовых вычислителей, с планированием последовательности их выполнения, синтезом диагностических моделей, высокоуровневым тестированием и автоматизацией всех этих процессов для разрабатываемых робототехнических комплексов.

Назначение заданий по процессорам вычислительных модулей [5–7]. Будем предполагать, что рассматриваемое множество задач представлено сетевой моделью, заданной ориентированным графом. В общем случае граф может содержать циклы, и более одного пути между двумя выделенными вершинами, может состоять из нескольких компонент связности, которые далее будем называть заданиями. Каждое задание состоит из набора задач, для которых известны длительности их выполнения на используемом вычислительном модуле (процессоре). Считаем, что все процессоры имеют одинаковую производительность. Кроме того, задано отношение предшествование между задачами, и известно время для передачи информации между задачами с использованием выбранных каналов связи.

Формальная постановка проблемы назначения. Допустим, что множество задач представлено графом $G(V,E) = \langle V,E\rangle, V \neq \emptyset, E \subseteq V \times V$, где V – вершины (задачи), E – ребра графа (задано отношение предшествования между задачами). Пусть граф $G(V,E)$ не являться связным и содержит n компонент связности (задания), т.е. назначению на процессоры подлежат n независимых заданий.

Каждое задание состоит из множества задач $\tau^{(j)} = \left\{\tau_k^{(j)} | k = \overline{1,n^{(j)}}, j = \overline{1,n}\right\}$, где $\tau_k^{(j)}$ – k-я задача j задания, $n^{(j)}$ – число задач в j задании. Известна длительность выполнения каждой задачи $p_k^{(j)}$, причём длительность нормирована относительно производительности процессора. Считаем, что любая задача может быть решена на одном процессоре, т.е. организация распараллеливания последовательного кода в работе не рассматривается. Известна длительность $s_{a,b}^{(j)}$ обмена передачи информации из задачи a в задачу b, где задачи входят в состав j задания (вес ребра графа $G(V,E)$).

Введём дополнительные обозначения. Пусть:

- m – общее число задач в вычислительной системе (если каждую задачу разместить на своём вычислительном модуле, m будет максимальным числом необходимых вычислительных модулей);

♦ $x_{i,k}^{(j)} \in \{0,1\}$ – признак размещения задачи, при этом 1 – задача k из j задания выполняется на i вычислительном модуле, 0 - иначе;

♦ $y_i \in \{0,1\}$ – признак использования вычислительного модуля, при этом 1 – i вычислительный модуль используется, 0 – иначе;

♦ $z_{\alpha,\beta} \in \{0,1\}$ – признак наличия канала обмена, при этом 1 – существование канала обмена между вычислительными задачами α и β, 0 – иначе.

Зададим ограничение по допустимому размещению задач на одном вычислительном модуле:

$$\sum_{j=1}^{n}\sum_{k=1}^{n^{(j)}} p_k^{(j)} x_{i,k}^{(j)} \le y_i\,, i = \overline{1,m}.$$

Зададим ограничения по допустимому объёму (по длительности) передачи данных между задачами размещенных на различных вычислительных модулях с учётом пропускной способности канала обмена:

$$\sum_{j=1}^{n}\sum_{a=1}^{n^{(j)}-1}\sum_{b=a+1}^{n^{(j)}} \left(s_{a,b}^{(j)} + s_{b,a}^{(j)}\right) x_{\alpha,a}^{(j)} x_{\beta,b}^{(j)} \le z_{\alpha,\beta}\,, \alpha = \overline{1,m-1}, \beta = \overline{\alpha+1,m}.$$

Требование определяет размещение всех задач из состава комплекса:

$$\sum_{i=1}^{m} x_{i,k}^{(j)} = 1\,, k = \overline{1,n^{(j)}}, j = \overline{1,n}$$

Определим критерий назначения, который представим взвешенной суммой числа используемых процессоров и числа каналом обменов:

$$f = \min_{f_1,f_2}(A \cdot f_1 + B \cdot f_2),$$

где f_1 является числом используемых вычислительных модулей:

$$f_1 = \sum_{i=1}^{m} y_i$$

f_2 – число используемых каналов обмена:

$$f_2 = \sum_{\alpha=1}^{m-1}\sum_{\beta=\alpha+1}^{m} z_{\alpha,\beta}$$

A и B, соответственно коэффициенты учитывающий вес вклада числа вычислительных модулей и каналов обмена. Данные коэффициенты задаются разработчиком комплекса исходя из практических соображений.

Описание подхода по решению проблемы назначения. Проблема назначения сводиться к квадратичной задаче о рюкзаке [6–10], которая формально является *NP*-трудной, и более того является *APX*-трудной [9–10]. Нахождение точного решения, даже с использованием алгоритмов основанных на вычислении верхней границы является вычислительно сложными задачами. Для решения проблемы предлагается использовать следующие известные эвристические алгоритмы: остовный и кластерный алгоритм [11].

Остовный алгоритм, который базируется на построении максимального остовного дерева, т.е. дерева включающего все вершины графа и характеризующегося максимальным значением суммарного веса всех ребер. Учитывая, что в графе $G(V,E)$ задано n компонент связности, все сформированные деревья сортируются по убыванию их веса. Затем осуществляется последовательный перебор деревьев,

и назначение задач на процессоры в последовательности, определяемой некоторым специальным образом выбранным обходом вершин дерева, при котором вероятным становится локализация поддеревьев остова в рамках некоторых процессоров и, как следствие, сокращение межпроцессорных обменов.

Кластерный алгоритм, который предполагает, что на один процессор назначаются в первую очередь пары задач, информационный обмен между которыми наиболее интенсивен.

В обоих алгоритмах в процессе назначения проверяются ограничения на производительность процессора (1) и пропускную способность канала обмена (2). При их превышении на некотором шаге число процессоров или каналов обмена увеличивается на единицу. Недостатки обоих эвристических алгоритмов очевидны. По эффективности они, безусловно, будут проигрывать оптимальным. Для улучшения их характеристик предлагается использовать двухэтапные процедуры, где на первом этапе используется остовный или кластерный алгоритм, а на втором (когда размерность задачи существенно уменьшилась) использовать более эффективный ресурсоёмкий алгоритм.

Исследование эффективности предложенных алгоритмов назначения осуществлялось путем генерации тестовых примеров заданий. Учитывая многообразие возможных вариантов структуры графа, формировался вариант графа без базовых циклов (дерево), и вариант с пятью базовыми циклами. Для каждого примера проводилась оценка оптимального значения согласно критерию (3), и выполнялась процедура назначения по остовному и кластерному алгоритму.

Было установлено, что остовный и кластерный алгоритмы имеют неперекрывающиеся области эффективного применения (доминирования), определяемые соотношением затрат процессор/канал обмена (B/A). Пример считался позитивным, если полученный результат «близок» к оценке оптимального значения. На рис. 1 представлена доля положительных примеров в зависимости от затрат весовых коэффициентов B и A.

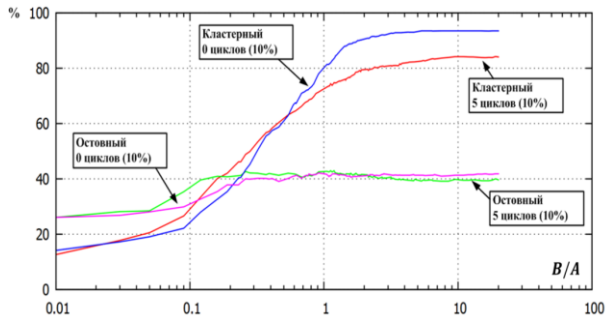

Рис. 1. Результаты исследования эффективности алгоритмов назначения

Планирование заданий [12–15]. В области планирования решение фокусируется на определение последовательности выполнения заданий в бортовой многоканальной системе обработки информации, когда система обрабатывает информацию, поступающую от одного или многих датчиков, но с формированием различных потоков обработки. Примером такой системы является бортовая система освещения обстановки робототехнического комплекса, когда информация поступившая с антенной решётки обрабатывается различными наборами задач – обнаружение и оценка параметров сигналов, задачи обеспечения безопасности маневрирования аппарата и т.д. При этом каждый поток информации обрабатывается по своему алгоритму с использованием распределенных вычислений, а все алгоритмы исполняются на общих вычислительных ресурсах. В результате, проблема планирова-

ния состоит в поиске упорядоченности заданий, наилучшей с точки зрения некоторого критерия. Данная проблема известна в теории расписаний под названием задачи поточного планирования (анг. – permutation flow shop scheduling) (PFS) [12].

Предполагается, что при назначении задач на процессоры было выполнено условие, гарантирующее независимость обработки разных порций входной информации. В этом случае исключается образование растущих очередей на процессоры из-за незавершенности обработки предыдущей порции входной информации. Следующая позиция постановки проблемы требует, чтобы передача и прием данных между задачами одного задания, выполняемыми на смежных процессорах, осуществлялась без дополнительных задержек по их готовности. Если же реально такие задержки существенны, то будем считать, что они учтены в используемых значениях длительностей решения задач.

Формальная постановка проблемы планирования [14, 16]. Будем предполагать, что по результату выполнения назначения, определено распределение задач по процессорам бортового вычислительного комплекса. При этом, рассматриваемое множество задач, задано информационным графом $G(V, E)$, содержащим n компонент связности (задания). Каждое задание $\tau^{(j)}, j = \overline{1,n}$ состоит из множества задач, которые изоморфны некоторому вычислительному графу (считаем, что задания выполняются на множестве процессоров с сохранением последовательности, т.е. порядок выполнения заданий на процессоре 1 совпадает с порядком выполнения заданий на процессоре 2 и т.д.). Известна длительность выполнения задач на процессорах вычислительных модулей: $p_i^{(j)}, i = \overline{1,m}, j = \overline{1,n}$, где m число используемых процессоров. Тогда для случая конвейерной обработки данных, условия времени выполнения задач будет:

$$
\begin{aligned}
&C(\pi_1, 1) = p_1^{(\pi_1)},\\
&C(\pi_j, 1) = C(\pi_{j-1}, 1) + p_1^{(\pi_j)}, \qquad j = \overline{2,n},\\
&C(\pi_1, i) = C(\pi_1, i-1) + p_i^{(\pi_1)}, \qquad i = \overline{2,m},\\
&C(\pi_j, i) = \max\{C(\pi_{j-1}, i), C(\pi_j, i-1)\} + p_i^{(\pi_j)}, j = \overline{2,n}, i = \overline{2,m}.\\
&C_{max}(\pi) = C(\pi_n, m),
\end{aligned}
$$

где $\pi = \{\pi_1, \pi_2, \dots, \pi_n\}$ – перестановка, которая определяет порядок выполнения заданий на процессорах, $C(\pi_j, i)$ – время завершения выполнения задания $\tau^{(j)}$ на процессоре i, $C_{max}(\pi)$ – время завершения выполнения последнего задания. Очевидно, что данное условие по времени выполнения задач на процессорах можно перенести и на произвольные ациклические графы.

По типу целевой функции, проблема планирования заданий делиться на задачи с минимаксными и с суммарными критериями оптимизации.

Сформулируем задачу с минимаксным критерием оптимизации: поиск такой перестановки π^* из множества П всех возможных перестановок заданий $\tau^{(j)}$, при которой $C_{max}(\pi^*) \le C_{max}(\pi)\ \forall \pi \in \Pi$, т.е. требуется определить такую последовательность выполнения заданий, при которой время выполнения последнего задания будет минимальным [16].

Для формулирования задачи с суммарным критерием оптимизации введём следующие понятия: джиттер задачи, временная привязка [17].

Пусть длительность выполнения задачи задано в интервале $\tilde{p}_i^{(j)} = \left[\underline{p}_i^{(j)}, \overline{p}_i^{(j)}\right]$, при этом величину $\Delta\left(\tau_i^{(j)}\right) = \overline{p}_i^{(j)} - \underline{p}_i^{(j)}$ будем называть джиттером задачи $\tau_i^{(j)}$.

Говорят, что -я задача j-го задания привязана к моменту времени $t_i^{(j)}$ с точностью $\delta_i^{(j)}$, если время начала решения этой задачи лежит в интервале $\left[t_i^{(j)} - \delta_i^{(j)}, t_i^{(j)} + \delta_i^{(j)}\right]$. Это понятие поясняется на рис. 2, где представлена временная

диаграмма исполнения трех задач $\tau_1^{(j)}, \tau_2^{(j)}, \tau_3^{(j)}$. Задача $\tau_1^{(j)}$, стоящая первой в цикле обработки, имеет точную временную привязку ($\delta_1^{(j)} = 0$). Однако уже привязка второй задачи $\tau_2^{(j)}$ характеризуется погрешностью $\delta_2^{(j)}$, равной неопределенности длительности выполнения первой задачи $\delta_2^{(j)} = \Delta\left(\tau_1^{(j)}\right) \neq 0$, привязка третьей задачи характеризуется погрешностью $\delta_3^{(j)}$, равной сумме неопределенностей длительностей выполнения первой и второй задач $\delta_3^{(j)} = \Delta\left(\tau_1^{(j)}\right) + \Delta\left(\tau_2^{(j)}\right) \neq 0$, и т.д.

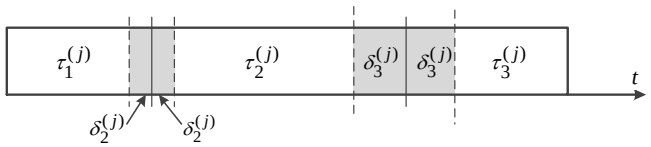

Рис. 2. Иллюстрация неточности временной привязки задач

Сформулируем задачу с суммарным критерием оптимизации: поиск такой перестановки π^* из множества П всех возможных перестановок заданий $\tau^{(j)}, j = \overline{1,n}$, при которой:

$$\sum_{j=1}^{n} \delta(\pi_j^*, m) \leq \sum_{j=1}^{n} \delta(\pi_j, m), \forall \pi \in \Pi.$$

т.е. требуется определить такую последовательность выполнения заданий, при которой суммарная неточность временной привязки будет минимальной.

Описание подхода по решению проблемы планирования. Проблема планирования является *NP*-трудной, поэтому для её решения воспользуемся следующим эвристическим алгоритмом на основе формализма разрешимых классов распределенных систем реального времени (РКС-формализм) для которых известен алгоритм нахождения оптимального плана [16].

Алгоритм планирования по критерию минимизации времени завершения последнего задания. Предлагаемый алгоритм основывается на понятии разрешимого класса распределенной системы, который вводится ниже. Для этого предварительно определим на множестве процессоров отношение доминирования «$\rhd$».

Определение 1. Процессор P_q доминирует над процессором P_r ($P_q \geq P_r$), если $\min_j p_q^{(j)} \geq \max_j p_r^{(j)}, j = \overline{1,n}$.

Общее свойство рассматриваемых далее разрешимых классов распределенных систем состоит в следующем: для любого задания, реализуемого в системе, критический путь единственен и проходит по одним и тем же процессорам. Номера процессоров вдоль критического пути обозначим как $P_{[1]}, P_{[2]}, \ldots, P_{[m^*]}$, где m^* будет длиной критического пути.

Приведем определяющие свойства для каждого из разрешимых классов.

Определение 2 (класс 1). Множество процессоров критического пути представляет собой последовательность $P_{[1]} \geq P_{[2]} \geq \cdots \geq P_{[m^*]}$, убывающую по отношению доминирования.

Определение 3 (класс 2). Множество процессоров критического пути представляет собой последовательность $P_{[1]} \leq P_{[2]} \leq \cdots \leq P_{[m^*]}$, возрастающую по отношению доминирования.

Определение 4 (класс 3). Множество процессоров критического пути представляет собой пару соединенных последовательность

$$P_{[1]} \leq P_{[2]} \leq \cdots \leq P_{[h]} \geq \cdots \geq P_{[m^*]}, 1 \leq h \leq m^*,$$

первая из которых возрастает, а вторая убывает по отношению доминирования.

Определение 5 (класс 4). Множество процессоров критического пути представляет собой пару соединенных последовательность

$$P_{[1]} \geq P_{[2]} \geq \cdots \geq P_{[h]} \leq \cdots \leq P_{[m^*]}, 1 \leq h \leq m^*,$$

первая из которых убывает, а вторая возрастает по отношению доминирования.

Очевидно, что на практике для распределенной системы общего вида, условия ее принадлежности к тому или иному разрешимому классу чаще всего не выполняются. В частности, могут не совпадать критические пути разных заданий, может нарушаться отношение доминирования между процессорами, а если оно и выполняется, то может не быть характерных для разрешимых классов упорядоченных по этому отношению цепочек процессоров. В результате исчезают гарантии оптимальности упомянутых выше алгоритмов, и теряет силу справедливый для разрешимых классов факт независимости качества плана от варианта упорядоченности некрайних заданий. В связи с этим используется пусть приближенный, но справедливый для любой из рассматриваемых систем рекурсивный алгоритм планирования, выполняемый за число шагов, не большее, чем исходное число заданий. На каждом шаге рекурсии определяется некоторый аналог критического пути, называемый псевдокритическим. Далее используется алгоритм планирования, соответствующий тому разрешимому классу, к которому наиболее близка рассматриваемая на данном шаге система. При этом выбранные задания занимают обе крайние позиции из интервала свободных позиций формируемого плана или одну из них. После размещения эти задания исключаются из исходного множества, и осуществляется переход к следующему шагу рекурсии, реализуемому уже для оставшегося множества заданий на множестве свободных позиций плана. В результате алгоритм последовательно размещает в плане все рассматриваемые задания в направлении от крайних заданий плана к центру.

Алгоритм планирования по критерию минимизации суммарной неточности временной привязки. По отношению к рассматриваемому ранее алгоритму планирования, новый алгоритм заключается не только в применении нового критерия, но и в оперировании интервальными длительностями задач. Как следствие, требуется небольшая корректировка определения отношения доминирования, но не определений разрешимых классов.

Определение 6. Процессор P_q доминирует над процессором P_r ($P_q \geq P_r$), если $\min_j \underline{p}_q^{(j)} \geq \max_j \overline{p}_r^{(j)}, j = \overline{1,n}$.

Далее будем различать входной и выходной джиттеры задания. В первом случае имеется в виду неточность привязки начала задания, а во втором случае – неточность привязки его конца.

Обозначим $\Delta_k(\pi)$ как выходной джиттер для плана π для k-го разрешимого класса системы.

Приведём следующие утверждения выбора последовательности заданий.

Утверждение 1. Минимальное значение среднего по заданиям выходного джиттера $\Delta_1(\pi)$ для системы из класса 1 достигается в плане π, в котором задания упорядочены по неубыванию джиттера их первых задач критического пути, т.е.

$$\Delta\left(\tau_{[1]}^{(1)}\right) \leq \Delta\left(\tau_{[1]}^{(2)}\right) \leq \cdots \leq \Delta\left(\tau_{[1]}^{(n)}\right).$$

Утверждение 2. Минимальное значение среднего по заданиям выходного джиттера $\Delta_2(\pi)$ для системы из класса 2 достигается в плане π, для которого выполняется:

1) задания упорядочены по неубыванию джиттера последних задач критического пути, т.е.

$$\Delta\left(\tau_{[m^*]}^{(1)}\right) \leq \Delta\left(\tau_{[m^*]}^{(2)}\right) \leq \cdots \leq \Delta\left(\tau_{[m^*]}^{(n)}\right).$$

2) первое задания плана π удовлетворяет условию

$$j^* = \arg\min_j \sum_{i=1}^{m^*-1} \Delta\left(\tau_{[i]}^{(j)}\right).$$

Утверждение 3. Минимальное значение среднего по заданиям выходного джиттера $\Delta_3(\pi)$ для системы из класса 3 достигается в плане π, для которого выполняется:

задания упорядочены по неубыванию джиттера задач стыковки критического пути, т.е.

$$\Delta\left(\tau_{[h]}^{(1)}\right) \le \Delta\left(\tau_{[h]}^{(2)}\right) \le \cdots \le \Delta\left(\tau_{[h]}^{(n)}\right).$$

1) первое задания плана π удовлетворяет условию

$$j^* = \arg\min_j \sum_{i=1}^{h-1} \Delta\left(\tau_{[i]}^{(j)}\right).$$

Утверждение 4. Минимальное значение среднего по заданиям выходного джиттера $\Delta_4(\pi)$ для системы из класса 4 достигается в плане π, для которого выполняется:

2) задания упорядочены по неубыванию суммарного джиттера первых и последних задач критического пути, т.е.

$$\left(\Delta\left(\tau_{[1]}^{(1)}\right) + \Delta\left(\tau_{[m^*]}^{(1)}\right)\right) \le \left(\Delta\left(\tau_{[1]}^{(2)}\right) + \Delta\left(\tau_{[m^*]}^{(2)}\right)\right) \le \cdots \le \left(\Delta\left(\tau_{[1]}^{(n)}\right) + \Delta\left(\tau_{[m^*]}^{(n)}\right)\right).$$

3) первое задания плана π удовлетворяет условию

$$j^* = \arg\min_j \sum_{i=1}^{m^*-1} \Delta\left(\tau_{[i]}^{(j)}\right).$$

Очевидно, что на практике, как и в предыдущем случае, для распределенной системы общего вида, описанной в постановке задачи, условия ее принадлежности к тому или иному разрешимому классу чаще всего не выполняются. В связи с этим предлагается аналогичный приближенный, но справедливый для любой из рассматриваемых систем итерационный алгоритм планирования, выполняемый за число шагов, не большее, чем число заданий.

Для исследования эффективности алгоритмов планирования был осуществлён модельный эксперимент, основанный на случайной генерации тестовых примеров. Проводилось исследование для различных ациклических графов, формировалась оценка нижней границы оптимального планирования заданий путём расширения подхода Тейлорда, предложенное им для конвейерной обработки информации [18].

Исследования показало, что средний проигрыш оценке нижней границы оптимального планирования заданий составил 8%.

Стоит отметить, что неоднозначность соотнесения системы к одному из разрешимых классов можно использовать в качестве генерации начального размещения агентов с дальнейшим использованием алгоритмов мультиагентной оптимизации, например для использования в методе роя частиц [19].

Диагностирование программного обеспечения бортовых вычислителей [20–24]. Рассмотрим метод тестового диагностирования с параллельной моделью для распределенных бортовых вычислительных систем реального времени [22]. Объектом рассмотрения являются комплексы, на вход которых поступает периодический поток данных, примером такого комплекса, является гидроакустическая аппаратура АНПА, где с заданным периодом осуществляется съём информации с

приёмного элемента антенной решётки. Данный метод предполагает введением в систему избыточности с целью упрощения диагностического эксперимента и снижения трудоемкости его подготовки.

Проиллюстрируем решение на следующем примере. Пусть для определенности граф информационных связей системы имеет вид, представленный на рис. 3. Каждая из систем на основе входных данных (u_1 – для Σ_1, u_2 – для Σ_2, и y_1 и y_2 – для Σ_3) формирует выходные (y_1, y_2 и y_3 соответственно).

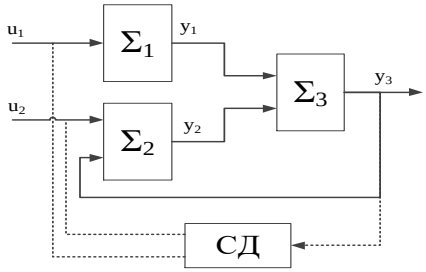

Рис. 3. Граф информационных связей системы

Все эти данные имеют вид массивов информационных слов. Пусть за период решения между системами происходит N сеансов обмена информацией. В данном случае их три. В первом сеансе система Σ_1 передает информацию системе Σ_3, во втором система Σ_2 передает информацию системе Σ_3, а в третьем система Σ_3 передает информацию системе Σ_2 и внешним потребителям.

Известно, что решение задачи диагностирования всегда связано с использованием в системе аппаратурной, информационной или временной избыточности. Сами системы диагностики (СД) являются примером такой избыточности. Кроме того, перед синтезом СД в систему может вводиться избыточность с целью упрощения процесса диагностирования. В частности, это могут быть шаги по увеличению числа входов или выходов системы, по введению в систему специальных средств анализа и т.п. Вводимая избыточность формально может быть отнесена как к системе, так и к СД, что несущественно. Именно этому направлению, предполагающему введение избыточности для упрощения диагностирования, и принадлежит рассматриваемый далее подход, названный тестовым диагностированием с параллельной моделью. Сразу оговоримся, что этот подход справедлив не только для распределенных систем, но и для одной локальной системы, рассматриваемой в виде композиции реализуемых в ней программных модулей (ПМ). Допустим и третий вариант, когда распределенная система, представлена композицией ПМ. В дальнейшем в обозначениях будем следовать последнему варианту.

На рис. 4 рассматриваемый пример представлен уже в новых обозначениях совместно с СД. В каждый ПМ системы уже введена избыточность в виде алгоритмов π_{d1}, π_{d2} и π_{d3}. СД состоят из генератора тестов (ГТ), генератора эталонных реакций (ГЭР) и компаратора (К).

СД формируют для системы тестовые данные, дополняя ими входные данные для $ПМ_1$ и $ПМ_2$, и анализирует выходную реакцию системы Σ ($ПМ_3$). В каждый ПМ – $ПМ_1$, $ПМ_2$ и $ПМ_3$ – по каналам обмена поступает информация, которая обрабатывается штатными алгоритмами. Параллельно с этим тестовые информационные слова обрабатываются специальными алгоритмами π_{d1}, π_{d2} и π_{d3}, реагирующими на события приема/выдачи информации, а результаты их обработки выдаются в составе выходных данных.

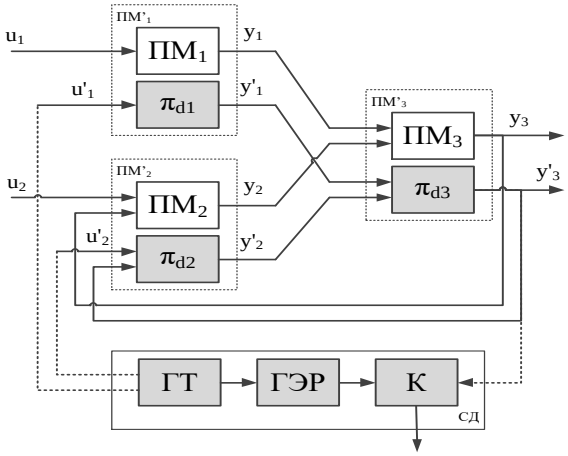

Рис. 4. Структура избыточной системы со средствами тестового диагностирования

Основу подхода к тестированию систем с параллельной моделью составляет алгоритм синтеза диагностической модели с иерархической структурой, состоящий из двух этапов. На первом этапе построения модели на основе известных алгоритмов [24] формируется множество вычислительных путей (трасс), составляющих покрытие дуг графа межмодульных связей. При этом под вычислительным путем понимаем последовательность срабатывающих ПМ, соединяющую некоторый вход системы с выходом. Затем с каждым из полученных путей сопоставляется цепь из такого числа динамических звеньев, через сколько ПМ проходит данный путь. После описанных построений модель системы представляется совокупностью функционально независимых цепей, а задача диагностирования может быть сведена к диагностированию отдельных цепей. На втором этапе учитывается, что искомая динамическая модель системы далее используется для построения тестов и что процедура построения тестов упрощается, если модель системы, во-первых, линейна, а во-вторых, управляема и наблюдаема. Отсюда можно сформулировать требование к звеньям цепей модели. Они должны быть линейны, т.е.

$$x_i(t+1) = f_i x_i(t) + g_i u_i(t),\ \ y_i(t) = h_i x_i(t),\ \ i = \overline{1,v},$$

где $x_i(t)$, $u_i(t)$, $y_i(t)$ – векторы состояния, входа и выхода, f_i, g_i, h_i – матрицы динамики, входа и выхода соответственно для i-го звена цепи, v – число звеньев в модели. Кроме того, звенья должны быть таковы, чтобы модель системы также была бы управляема и наблюдаема.

Описание цепи, получается, по следующим правилам. Предположим, что в каждый момент времени в системе происходит лишь один обмен. Тогда формируется вектор состояния $x(t)$, составленный из векторов состояния звеньев $x_i(t)\ i = \overline{1,v}$, входящих в эту модель, а с помощью матриц $F(t), G(t), H(t)$ описывается перенос информации между ПМ или ПМ и СД в каждом i-м информационном обмене. Если для удобства описания связать с каждой последовательностью матриц на интервале, равном периоду, свою последовательность индексов, множество которых обозначим через $\Gamma = \{\gamma_s | s = \overline{1,N}\}$, описание модели для системы Σ принимает вид:

$$x(t+1) = F\big(\gamma_s(j)\big)x(t) + G\big(\gamma_s(j)\big)u(t),\ \ y(t) = H(\gamma_s(j)x(t),\ \ j = \overline{1,N}.$$

Матрицы в этих уравнениях зависят от номера такта (номера обмена), т.е. модель нестационарна. Более того, она периодически нестационарна, т.к. из-за периодичности входного потока процессы обработки данных в системе периодические.

Центральной проблемой второго этапа синтеза диагностической сетединамической модели является обеспечение ее наблюдаемости и управляемости путем соответствующего выбора ее звеньев. Эта проблема решается в результате анализа соответствующей стационарной модели системы. Последняя получается путем сведения к ней исходной периодически нестационарной системы. Проблема сведения или, другими словами, замены периодически нестационарной модели стационарной известна, причем известно также, что она не всегда имеет решение. Однако в важном для нас случае такое сведение возможно [23].

Опишем процедуру построения теста. Тест предназначен для обнаружения всех неэквивалентных искажений матриц системы, т.е. таких искажений, при которых неисправная система неэквивалентна исправной. Тест состоит из N фрагментов:

$$U_T = U_{\gamma_1} U_{\gamma_2} \dots U_{\gamma_N},$$

каждый из которых включает две характерные части:

$$U_{1\gamma_r} = u^*_{1\gamma_r} 0^{nN} u^*_{2\gamma_r} 0^{nN} \dots u^*_{n\gamma_r} 0^{nN},$$
$$U_{2\gamma_r} = u_{1\gamma_r} 0^{nN} u_{2\gamma_r} 0^{nN} \dots u_{m\gamma_r} 0^{nN}.$$

В первой части $U_{1\gamma_r}$ всех фрагментов система при последовательности матриц γ_r проходит в пространстве состояний через состояния некоторого выбранного базиса $X = \{x_s | s = \overline{1,n}\}$. Для каждого состояния x_s в фрагменте предусмотрены установочная последовательность u^*_{s,γ_r} длиной, кратной N, и интервал свободного движения при последовательности матриц γ_r, когда на входе системы есть nN нулей, обозначенные как 0^{nN}. Благодаря тому, что длины установочных последовательностей кратны N, все интервалы свободного движения система проходит при последовательности матриц γ_r. Во второй части $U_{2\gamma_r}$, всех фрагментов на вход системы последовательно подаются векторы $u_{q,\gamma_r}, q = \overline{1,m}$, принадлежащие некоторому базису пространства входных векторов. После каждого вектора система находится в свободном движении на nN тактах при последовательности матриц γ_r.

Заключение. Рассмотренные в работе алгоритмы реализованы в инструментальной среде проектирования программного обеспечения бортовых вычислителей робототехнических комплексов. На входе среды задаётся описание информационного графа и параметры задач. В результате выполнения, формируется проект, реализующий управления вычислительным процессом бортовой вычислительной системы с решением рассмотренных в работе проблем.

Практическая значимость работы определяется тем, что предложенные алгоритмы позволяют повысить эфективность применяемых на практике средств высокоуровневого проектирования. Предложенные решения были применены при разработке бортовых комплексов в АО «Концерн «ЦНИИ «Электроприбор».

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Платунов А.Е.* Теоретические и методологические основы высокоуровневого проектирования встраиваемых вычислительных систем. – СПб.: Университет ИТМО, 2010. – 477 с.
2. *Harris D., Harris S.* Digital Design and Computer Architecture. – 2nd ed. – Morgan Kaufman, 2012.
3. *Liu J.W.S.* Real-Time Systems. Prentice Hall. – NJ, 2000. – 590 p.
4. *Топорков В.В.* Модели распределенных вычислений. – М.: Физматлит, 2004. – 316 с.
5. *Edward A. Lee and Sanjit A. Seshia.* Introduction to Embedded Systems, A Cyber-Physical Systems Approach. – Second ed. – MIT Press, 2017.

6. *Kellerer H., Pferschy U., Pisinger U.* Knapsack Problems. – Springer, Berlin, Germany, 2004. – 546 p.
7. *Burkard R.E., Dell'Amico M. and Mortello S.* Assignment problems. – SIAM, Philadelphia, 2009.
8. *Gonzalez T.F.* Handbook of Approximation Algorithms and Metaheuristics. – Second ed. Vol. 2: Contemporary and Emerging Applications. Ed. Chapman. – 2018. – https://doi.org/10.1201/9781351235426.
9. *Jansen K.* Parameterized Approximation Scheme for the Multiple Knapsack Problem // SIAM Journal on Computing. – 2010. – Vol. 39, No. 4. – P. 1392-1412. – https://doi.org/10.1137/080731207.
10. *Cacchiani V., Iori M., Locatelli A. et. al.* Knapsack problems - An overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems // Computers & Operations Research. – 2022. – Vol. 143. – P. 105693. – https://doi.org/10.1016/j.cor.2021.105693.
11. *Грузликов А.М., Колесов Н.В., Скородумов Ю.М., Толмачева М.В.* Графовый подход к назначению заданий в распределенных системах реального времени // Известия РАН. Теория и системы управления. – 2014. – № 5. – С. 84-94.
12. *Cottet F., Kaiser J., Mammeri Z.* Scheduling in Real-Time Systems. – John Wiley & Sons Ltd, 2002.
13. *Brucker P., Knust S.* Complex scheduling. – Springer-Verlag, Berlin, Heidelberg, 2006.
14. *Лазарев А.А.* Теория расписаний. Методы и алгоритмы. – М.: ИПУ РАН, 2019. – 408 с.
15. *Pinedo M.L.* Scheduling. Theory: Algorithms and systems. – Springer. Science, 2016. – 670 p.
16. *Грузликов А.М., Колесов Н.В., Скородумов Ю.М., Толмачева М.В.* Планирование заданий в распределенных системах реального времени // Известия РАН. Теория и системы управления. – 2017. – № 2. – С. 67-76.
17. *Грузликов А.М., Колесов Н.В.* Использование разрешимых классов систем реального времени для планирования с минимизацией джиттера // Известия РАН. Теория и системы управления. – 2021. – № 6. – С. 43-51.
18. *Taillard E.* Benchmarks for Basic Scheduling Problems // Europ. J. Operational Research. – 1993. – Vol. 64, No. 2. – P. 278-285.
19. *Zhanq X., Guo P., Zhanq H., Yao, J.* Hybrid Particle Swarm Optimization Algorithm for Process Planning // *Mathematics*. – 2020. – Vol. 8. – P. 1745. – https://doi.org/10.3390/math8101745.
20. *Isermann R.* Fault Diagnosis Application. – Heidelberg: Springer, 2011. – 354 p.
21. *Zaytoon J., Lafortune S.* Overview of Fault Diagnosis Methods for Discrete Event Systems // Annual Reviews in Control. – 2013. – Vol. 37. – P. 308-320.
22. *Грузликов А.М., Колесов Н.В., Лукоянов Е.В.* Тестовое диагностирование нарушений адресации информационных обменов в вычислительных системах с использованием параллельной модели // Известия РАН. Теория и системы управления. – 2018. – № 3. – С. 76-89.
23. *Gruzlikov A., Kolesov N., Lukoyanov E., and Tolmacheva M.* Test-based diagnosis of distributed computer system using a time-varying model // 10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes, SAFEPROCESS – 2018.
24. *Gruzlikov A.M., Kolesov N.V.* Hierarchical model for testing a distributed computer system // J. Korbicz et al. (eds.), Advances in Diagnostics of Processes and Systems, Studies in Systems, Decision and Control 313, 14th International Conference on Diagnostics of Processes and Systems (DPS), 2021.

REFERENCES

1. *Platunov A.E.* Teoreticheskie i metodologicheskie osnovy vysokourovnevogo proektirovaniya vstraivaemykh vychislitel'nykh sistem [Theoretical and methodological bases of high-level design of embedded computing systems]. St. Petersburg: Universitet ITMO, 2010, 477 p.
2. *Harris D., Harris S.* Digital Design and Computer Architecture. 2nd ed. Morgan Kaufman, 2012.
3. *Liu J.W.S.* Real-Time Systems. Prentice Hall. NJ, 2000, 590 p.
4. *Toporkov V.V.* Modeli raspredelennykh vychisleniy [Distributed computing models]. Moscow: Fizmatlit, 2004, 316 p.
5. *Edward A. Lee and Sanjit A. Seshia.* Introduction to Embedded Systems, A Cyber-Physical Systems Approach. Second ed. MIT Press, 2017.
6. *Kellerer H., Pferschy U., Pisinger U.* Knapsack Problems. Springer, Berlin, Germany, 2004, 546 p.

7. *Burkard R.E., Dell'Amico M. and Mortello S.* Assignment problems. SIAM, Philadelphia, 2009.
8. *Gonzalez T.F.* Handbook of Approximation Algorithms and Metaheuristics. Second ed. Vol. 2: Contemporary and Emerging Applications. Ed. Chapman, 2018. Available at: https://doi.org/10.1201/9781351235426.
9. *Jansen K.* Parameterized Approximation Scheme for the Multiple Knapsack Problem, *SIAM Journal on Computing*, 2010, Vol. 39, No. 4, pp. 1392-1412. Available at: https://doi.org/10.1137/080731207.
10. *Cacchiani V., Iori M., Locatelli A. et. al.* Knapsack problems - An overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems, *Computers & Operations Research*, 2022, Vol. 143, pp. 105693. Available at: https://doi.org/10.1016/j.cor.2021.105693.
11. *Gruzlikov A.M., Kolesov N.V., Skorodumov Yu.M., Tolmacheva M.V.* Grafovyy podkhod k naznacheniyu zadaniy v raspredelennykh sistemakh real'nogo vremeni [Graph approach to assignment of tasks in distributed real-time systems], *Izvestiya RAN. Teoriya i sistemy upravleniya* [Izvestiya of the RAS. Theory and control systems], 2014, No. 5, pp. 84-94.
12. *Cottet F., Kaiser J., Mammeri Z.* Scheduling in Real-Time Systems. John Wiley & Sons Ltd, 2002.
13. *Brucker P., Knust S.* Complex scheduling. Springer-Verlag, Berlin, Heidelberg, 2006.
14. *Lazarev A.A.* Teoriya raspisaniy. Metody i algoritmy [Schedule theory. Methods and algorithms]. Moscow: IPU RAN, 2019, 408 p.
15. *Pinedo M.L.* Scheduling. Theory: Algorithms and systems. Springer. Science, 2016, 670 p.
16. *Gruzlikov A.M., Kolesov N.V., Skorodumov Yu.M., Tolmacheva M.V.* Planirovanie zadaniy v raspredelennykh sistemakh real'nogo vremeni [Task scheduling in distributed real-time systems], *Izvestiya RAN. Teoriya i sistemy upravleniya* [Izvestiya of the RAS. Theory and control systems], 2017, No. 2, pp. 67-76.
17. *Gruzlikov A.M., Kolesov N.V.* Ispol'zovanie razreshimykh klassov sistem real'nogo vremeni dlya planirovaniya s minimizatsiey dzhittera [Using solvable classes of real-time systems for planning with jitter minimization], *Izvestiya RAN. Teoriya i sistemy upravleniya* [Izvestiya of the RAS. Theory and control systems], 2021, No. 6, pp. 43-51.
18. *Taillard E.* Benchmarks for Basic Scheduling Problems, *Europ. J. Operational Research*, 1993, Vol. 64, No. 2, pp. 278-285.
19. *Zhang X., Guo P., Zhang H., Yao, J.* Hybrid Particle Swarm Optimization Algorithm for Process Planning, *Mathematics*, 2020, Vol. 8, pp. 1745. Available at: https://doi.org/10.3390/math8101745.
20. *Isermann R.* Fault Diagnosis Application. Heidelberg: Springer, 2011, 354 p.
21. *Zaytoon J., Lafortune S.* Overview of Fault Diagnosis Methods for Discrete Event Systems, *Annual Reviews in Control*, 2013, Vol. 37, pp. 308-320.
22. *Gruzlikov A.M., Kolesov N.V., Lukoyanov E.V.* Testovoe diagnostirovanie narusheniy adresatsii informatsionnykh obmenov v vychislitel'nykh sistemakh s ispol'zovaniem parallel'noy modeli [Test diagnostics of addressing violations of information exchanges in computing systems using a parallel model], *Izvestiya RAN. Teoriya i sistemy upravleniya* [Izvestiya of the RAS. Theory and control systems], 2018, No. 3, pp. 76-89.
23. *Gruzlikov A., Kolesov N., Lukoyanov E., and Tolmacheva M.* Test-based diagnosis of distributed computer system using a time-varying model, *10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes, SAFEPROCESS – 2018*.
24. *Gruzlikov A.M., Kolesov N.V.* Hierarchical model for testing a distributed computer system, *J. Korbicz et al. (eds.), Advances in Diagnostics of Processes and Systems, Studies in Systems, Decision and Control 313, 14th International Conference on Diagnostics of Processes and Systems (DPS), 2021.*

Статью рекомендовал к опубликованию д.т.н. Н.В. Колесов.

Грузликов Александр Михайлович – Государственный научный центр Российской Федерации АО «Концерн «ЦНИИ «Электроприбор»; e-mail: agruzlikov@yandex.ru; г. Санкт-Петербург, Россия; тел.: 89312664852; к.т.н.; начальник отдела.

Gruzlikov Alexander Mikhailovich – State Research Center of the Russian Federation Concern CSRI Elektropribor; e-mail: agruzlikov@yandex.ru; Saint Petersburg, Russia; phone: +79312664852; cand. of eng. sc.; head of the department.