СПОСОБ ОПТИМИЗАЦИИ СТРУКТУРЫ ПРОГРАММНО-ТЕХНИЧЕСКОГО КОМПЛЕКСА РАСПРЕДЕЛЕННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ ДЛЯ КРУПНЫХ ПРОМЫШЛЕННЫХ ПРЕДПРИЯТИЙ

Закирзянов Р.М.,
ООО «НЕКСТ инжиниринг», Казань, Россия
zr@nexteng.ru

Аннотация. В статье предлагается способ оптимизации структуры программно-технического комплекса автоматизированной системы управления непрерывными технологическими процессами для крупных промышленных предприятий. Даются общие сведения об актуальности проблемы выбора структуры системы, построенной на базе серийно выпускаемых компонентов, приводится формальное описание задачи оптимизации, выделяются критерий и ограничения. Описывается способ решения с применением метаэвристического алгоритма муравьиных колоний. Приводится численный пример решения, анализируются результаты работы алгоритма, определяются направления дальнейших исследований.

Ключевые слова: распределенная система управления, структурный синтез, структурная оптимизация, иерархическая структура, метаэвристический алгоритм, автоматизированная система управления технологическими процессами.

Введение

Современные системы управления технологическими процессами крупных промышленных предприятий строятся, как правило, на базе программно-технических комплексов [1], компоненты которых серийно выпускаются промышленностью. Свойства и функциональные возможности таких распределенных систем управления во многом определяются структурой комплекса технических средств, так как характеристики применяемых компонентов задаются заводами-изготовителями и не подлежат изменению. Таким образом, для достижения требуемых характеристик системы необходимо построить такую структуру из компонентов с известными характеристиками, которая была бы по возможности минимальной по стоимости и удовлетворяла всем требованиям и ограничениям, то есть была бы оптимальной.

Построение оптимальной структуры является важной задачей, которая в настоящее время в основном решается эмпирически на основании опыта проектировщика и рекомендаций заводов-изготовителей программно-технических средств. Особенно важной задача становится при проектировании систем управления крупными опасными территориально распределенными объектами с большим количеством параметров. В настоящей статье рассмотрена формализация задачи оптимизации структуры программно-технического комплекса автоматизированной системы управления технологическими процессами, предлагается способ ее решения, а также даются рекомендации по дальнейшему развитию исследований в данной области.

1. Особенности распределенных систем управления непрерывными технологическими процессами

Технологические процессы на промышленных предприятиях условно разделяют на непрерывные и дискретные [2]. Непрерывными технологическими процессами характеризуются нефтегазовая, химическая, нефтехимическая отрасли, а также частично металлургия и теплоэнергетика. Объекты управления крупных промышленных предприятий указанных отраслей имеют свои характерные особенности. Такие объекты, как правило, территориально распределены, являются опасными и технически сложными, имеют непрерывные процессы. На таких объектах часто происходят изменения: меняются параметры технологического режима, часть оборудования выводится в ремонт. Ввод в эксплуатацию таких объектов происходит частями (очередями) или этапами.

Серийно выпускаемые промышленностью программно-технические комплексы для построения на их базе АСУТП крупных промышленных предприятий с непрерывными технологическими

Данная версия является авторской версией статьи, опубликованной в сборнике материалов конференции MLSD’2025. Официальная версия доступна по ссылке: https://mlsd2025.ipu.ru/proceedings/1002.pdf

процессами получили в литературе [3] название DCS – Distributed Control System. В русскоязычной литературе чаще всего применяется термин «РСУ» – распределенная система управления [2]. Стоит отметить, что термин «РСУ», как в отечественной, так и в зарубежной литературе, перегружен значениями. В данной статье под «РСУ» понимается исключительно специализированный программно-технический комплекс для построения систем управления сложными непрерывными технологическими процессами.

Особенностью РСУ является наличие специальных средств и функций, направленных на решение двух задач: обеспечение безопасности процесса и снижение времени внедрения. Такими специальными мерами могут быть наличие возможности загрузки прикладного ПО без останова процесса, наличие частичной загрузки прикладного ПО, штатные функции резервирования, наличие единой базы тегов, бессерверная архитектура и др.

Программно-технический комплекс РСУ, как правило, представляет собой иерархическую структуру (рис. 1).

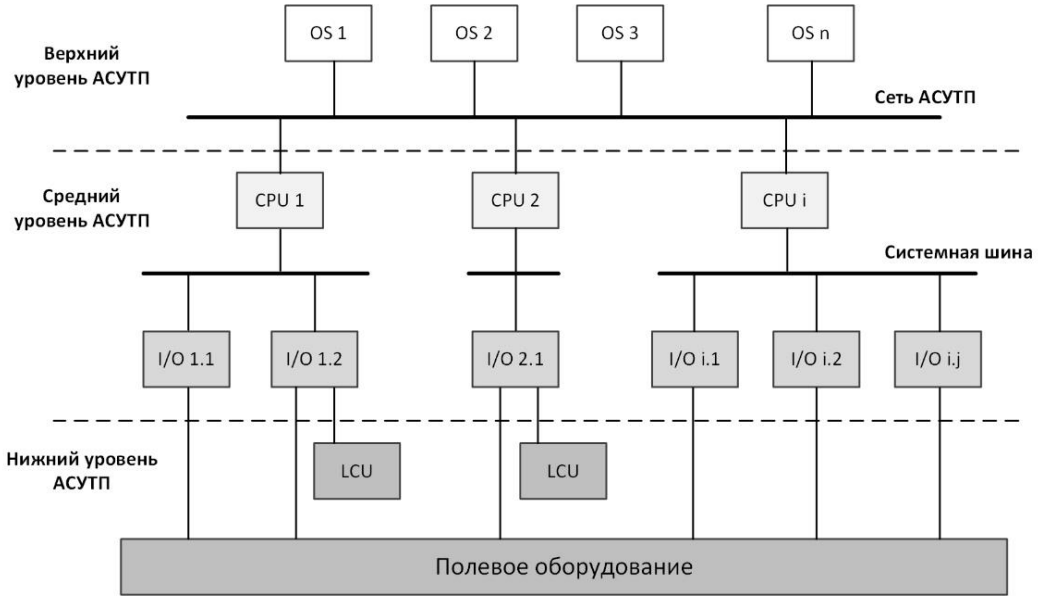

Рис. 1. Типовая структура РСУ

Здесь CPU – процессорные модули станций автоматизации (контроллеров), I/O – станции ввода-вывода, OS – автоматизированные рабочие места (АРМ) операторов технологического процесса, LCU – локальные системы управления.

Проектировщик АСУТП при создании системы решает комплексную задачу. Необходимо выбрать такие компоненты для системы и построить из них такую архитектуру, чтобы качественно решалась задача управления объектом, система удовлетворяла требованиям надежности и была минимальной по стоимости.

2. Описание структуры

Для постановки задачи поиска оптимальной структуры РСУ представим ее структуру в виде дерева (ациклического графа), пример которого показан на рис. 2. Здесь на уровне 3 расположены «полевые» устройства (датчики и исполнительные механизмы), находящиеся на «нижнем уровне» АСУТП (рис. 1). На уровне 2 расположены устройства связи с объектом (устройства ввода-вывода). На уровне 1 расположены контроллеры (станции автоматизации), отвечающие за реализацию алгоритмов управления объектом. «Верхний уровень» АСУТП в рамках данного дерева не рассматривается. Количество уровней в структуре задается проектировщиком и может варьироваться.

Методы оптимизации иерархических структур подробно рассмотрены в [4, 5, 6].

Пусть задана иерархическая структура распределенной системы управления в виде дерева $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, где $v \in \mathcal{V}$ – устройства (вершины графа), $\mathcal{E}$– ребра графа (каналы связи между устройствами).

Пусть также задано множество типов устройств $\mathcal{U} = \{u_1, \dots, u_U\}$, состоящее из U типов устройств.

Примем, что каждый узел структуры (устройство) любого типа выполняет однотипные действия, состоящие из трех фаз цикла работы внутренней программы устройства:

- сбор (чтение) информации от объекта управления, либо от узлов предыдущего уровня иерархии;
- обработка информации (реализация алгоритмов управления);
- выдача информации (запись) на нижестоящий уровень, либо воздействие на объект управления.

Не все устройства могут обрабатывать сигналы. Устройство, которое обладает функцией обработки информации, будем называть обработчиком (это могут быть контроллеры и серверы обработки информации). Устройство, которое способно только передавать информацию, будем называть ретранслятором (это могут быть коммутаторы и модули ввода-вывода).

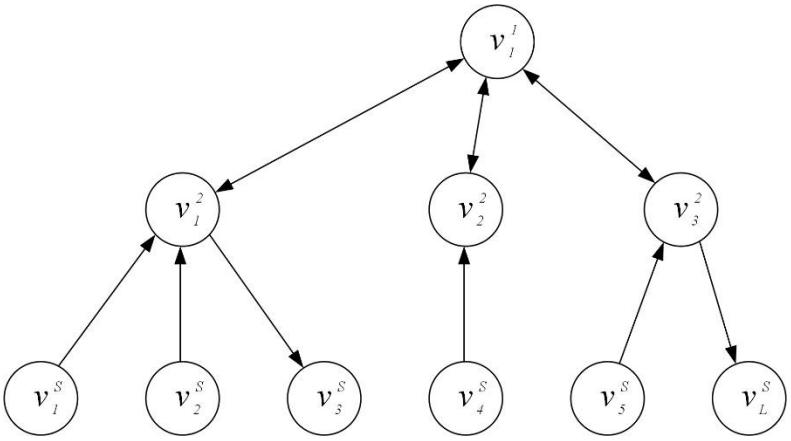

Рис. 2. Иерархическая структура РСУ

Каждый тип $u_i \in \mathcal{U}$ узла структуры характеризуется следующими параметрами:

- стоимость устройства $C_i \in \mathbb{R}^+$;
- количество подключаемых физических каналов $N_i \in \mathbb{N}$;
- максимальный объем памяти $R_i \in \mathbb{R}^+$;
- вероятность отказа устройства $P_i \in [0,1]$;
- производительность (время выполнения одной программной инструкции) $T_i \in \mathbb{R}^+$;
- режим работы $y_i \in \{0,1\}$ (1 – обработчик, 2 – ретранслятор);
- максимальное количество дочерних устройств $M_i \in \mathbb{N}$;
- задержка передачи для ретрансляторов $\tau_i \in \mathbb{R}^+$ (для обработчиков принимаем $\tau_i = 0$).

Каждый тип устройства представляет собой вектор (1).

$$u_i = (C_i, N_i, R_i, P_i, T_i, y_i, M_i, \tau_i), \quad i = \overline{1,U}. \tag{1}$$

Пусть также задано множество контуров управления $\mathcal{A} = \{a_1, \dots, a_A\}$, состоящее из A контуров $a_j \in \mathcal{A}$:

$$a_j = \left(n_j, r_j, w_j\right), \quad j = \overline{1,J}, \tag{2}$$

где $n_j \in \mathbb{N}$ – количество физических сигналов в контуре, $r_j \in \mathbb{R}^+$ – количество памяти, требуемое для хранения всех инструкций и переменных контура, $w_j \in \mathbb{N}$ – количество инструкций в программе обработки сигналов данного контура (характеризует сложность программы обработки контура).

Контуром управления в данном случае может быть, например, модуль (набор модулей) ввода-вывода, подключенный к полевому оборудованию. Контур в общем случае может содержать различные типы сигналов (аналоговые, дискретные, входные, выходные и т.д.). Для данной задачи принимаем, что все «полевые» сигналы заранее распределены по контурам, количество сигналов в каждом контуре известно, количество инструкций в программе обработки контура известно, количество памяти, необходимое для хранения всех переменных контура и всех инструкций

программы обработки этого контура, также заранее известно (оценено). Считаем, что для выполнения любой инструкции программы обработки контура требуется одинаковое время T_i.

Структура оптимальной системы управления представляет собой иерархическую конструкцию (дерево) с количеством уровней $s = S \in \mathbb{N}$, при этом $s = 1$ соответствует корню дерева, то есть вершине иерархии, $s = S$ соответствует уровню листьев дерева, то есть самому нижнему уровню иерархии. $L \in \mathbb{N}$ – количество листьев дерева (устройств, непосредственно подключенных к полевому оборудованию), причем множество листьев дерева обозначим через $\mathcal{L} \subseteq \mathcal{V}$. $V \in \mathbb{N}$ – общее количество устройств в структуре. K_s – количество устройств на уровне s. Количество листьев дерева $K_S = L$. Множество листьев $\mathcal{L} = \{v_1^S, \dots, v_L^S\}$.

Опираясь на вышесказанное, для каждого устройства $v_k^s \in \mathcal{V}$ введем следующие функции:

- $u(v_k^s) = u_k^s \in \mathcal{U}$ – тип устройства;
- $\pi(v_k^s) \in \mathcal{V} \cup \{\emptyset\}$ – родительское устройство (родительский узел);
- $\mathcal{D}(v_k^s) \subseteq \mathcal{V}$ – множество дочерних узлов;
- $\mathcal{D}'(v_k^s) \subseteq \mathcal{V}$ – поддерево узла v_k^s, такое что $\mathcal{D}(v_k^s) \subseteq \mathcal{D}'(v_k^s)$;
- $\mathcal{P}(v_k^s) \subseteq \mathcal{V}$ – путь от узла v_k^s к корню дерева.

Здесь $s \in \{1, \dots, S\}$ – уровень в иерархии, $k \in \mathbb{N}$ – порядковый номер узла на данном уровне. Таким образом, каждое устройство также может быть представлено в виде вектора (3):

$$v_k^s = \left(u_k^s, \pi(v_k^s), \mathcal{D}(v_k^s), \mathcal{D}'(v_k^s)\right), \quad s = \overline{1,S}, \quad k = \overline{1,K_s}. \tag{3}$$

Для решения задачи принимаем, что только устройства уровня S (листья) непосредственно соединены с полевым оборудованием технологической установки, то есть могут принимать информацию от объекта управления и выдавать управляющие воздействия для реализации надлежащего управления объектом. Также принимаем, что все устройства однотипны, различаются только величинами указанных выше параметров. Принимаем, что горизонтальные связи между устройствами отсутствуют. Каждый контур должен быть физически подключен к какому-либо листу дерева. Лист v может сам обработать подключенные к нему контуры, если он является обработчиком ($y = 1$), или передать контур на обработку вышестоящему обработчику, если лист является ретранслятором. Каждый контур должны быть назначен какому-либо единственному обработчику. Обработчик может обрабатывать только контуры, подключенные к листьям из его поддерева.

Будем считать, что задержку вносят только ретрансляторы. Обработчики циклически выполняют программы обработки всех назначенных на них контуров. Время работы обработчика равно суммарному времени обработки всех назначенных ему контуров.

Введем также дополнительные переменные (индексы s, k и j в дальнейшем опущены для упрощения записи).

- $x_{va} \in \{0,1\}$, где $x_{va} = 1$, если контур a подключен к листу v ($\forall v \in \mathcal{L}, \forall a \in \mathcal{A}$), в противном случае $x_{va} = 0$;
- $z_{va} \in \{0,1\}$, где $z_{va} = 1$, если контур a назначен обработчику v ($\forall v \in \mathcal{V}, \forall a \in \mathcal{A}$), в противном случае $z_{va} = 0$.

Далее будем использовать следующую терминологию. Если контур управления физически подключен к узлу уровня S, то будем говорить, что контур a подключен к листу v ($x_{va} = 1$). Если сигналы контура a обрабатываются в узле v, то будем говорить, что контур a назначен на узел v ($z_{va} = 1$).

3. Ограничения

Для древовидной структуры распределенной системы управления справедливы следующие приведенные ниже ограничения. У дерева существует единственный корень (4). При этом корень дерева может быть как обработчиком, так и ретранслятором.

$$\exists!\, v_1^1 \in \mathcal{V}: \pi(v_1^1) = \emptyset. \tag{4}$$

Узлы, расположенные на уровне S (листья дерева), не могут иметь дочерних элементов:

$$|\mathcal{D}(v)| = 0, \quad \forall v \in \mathcal{L}. \tag{5}$$

К каждому узлу можно подключить ограниченное количество дочерних устройств (6). Например, сетевой коммутатор или контроллер могут иметь ограниченное количество портов Ethernet.

$$|\mathcal{D}(v)| \le M_i, \quad \forall v \in \mathcal{V} \backslash \mathcal{L}, \tag{6}$$

где $u(v) = u_i$. К узлу v типа u_i может быть подключено не более M_i дочерних узлов. Выражение (6) справедливо для всех узлов, кроме листьев ($\mathcal{L}$).

Каждый контур может быть подключен только на один лист:

$$\sum_{x \in \mathcal{L}} x_{va} = 1, \ \forall a \in \mathcal{L}. \tag{7}$$

Запишем ограничения, характеризующие обработку контуров управления узлами-обработчиками. Только обработчики могут выполнять обработку сигналов контуров, то есть контур не может быть назначен на узел, не являющийся обработчиком:

$$z_{va} \le y_v, \quad \forall v \in \mathcal{V}, \quad \forall a \in \mathcal{A}. \tag{8}$$

Для каждого контура может существовать только один обработчик:

$$\sum_{v \in \mathcal{V}} z_{va} = 1, \quad \forall a \in \mathcal{A}. \tag{9}$$

Для каждого обработчика существует ограничение по памяти (10). Если суммарная память, требуемая для хранения информации всех назначенных на данный обработчик контуров, будет превышать размер памяти данного типа узла, узел не сможет обработать все контуры. В этом случае необходим дополнительный обработчик, который возьмет на себя часть нагрузки.

$$\sum_{a \in \mathcal{A}} z_{va} r_a \le R_v, \quad \forall v \in \mathcal{V}. \tag{10}$$

Обработчик может обрабатывать только контуры, подключенные к листьям поддерева этого обработчика (11). Таким образом, обработчик обслуживает только свое поддерево, контуры, подключенные к листьям других поддеревьев, не могут быть назначены на данный обработчик.

$$z_{va} \le \sum_{\substack{v' \in \mathcal{L} \\ v' \in \mathcal{D}'(v)}} x_{v'a}, \quad \forall v \in \mathcal{V}, \quad \forall a \in \mathcal{A}. \tag{11}$$

Суммарное количество сигналов всех подключенных к листу v контуров должно быть не больше количества каналов этого листа (12). Количество каналов листа N_v определяется типом узла. У обработчика может быть $N_v = 0$.

$$\sum_{a \in \mathcal{A}} x_{va} n_a \le N_v, \quad \forall v \in \mathcal{V}. \tag{12}$$

Поскольку проектируемая структура является структурой системы управления динамическим объектом, важно при синтезе учесть динамические свойства системы. На практике это является довольно сложной задачей [7]. В качестве показателя, характеризующего качество системы управления, примем время цикла работы программы (период квантования) для одного контура управления. При слишком большом времени цикла программы обработки контура функция обработки информации может быть передана на вышестоящий уровень иерархии для обработки более производительным узлом. Для любого контура, независимо от его подключения и назначения, время обработки T_{cont} не должно превышать заданное время $T_{max} \in \mathbb{R}^+$. Время обработки контура складывается из времени обработки всех инструкций всех назначенных на данный узел контуров и суммарного времени всех задержек, вносимых ретрансляторами на пути от листа к обработчику (13).

$$T_{cont} = \max_{a \in \mathcal{A}} \left[\sum_{a' \in \mathcal{A}} z_{v(a)a'} \cdot w_{a'} \cdot T_{v(a)} + \sum_{v' \in \mathcal{P}(v(a),a)} (1 - y_{v'}) \tau_{v'} \right] \le T_{max}, \tag{13}$$

где $v(a)$ – обработчик контура a, $\mathcal{P}(v(a), a)$ – путь от листа с контуром a до обработчика $v(a)$, $(1 - y_{v'})$ – индикатор ретранслятора (1 для ретрансляторов и 0 для обработчиков).

При проектировании системы необходимо учитывать ее надежность. В качестве показателя надежности системы используем вероятность отказа. Будем считать, что система приходит в неработоспособное состояние при отказе хотя бы одного из ее компонентов. Все отказы устройств в системе считаем независимыми. Вероятность отказа системы P_{sys} в этом случае будет равна произведению вероятностей отказа всех ее компонентов. Методика расчета надежности иерархических

автоматизированных систем управления технологическими процессами приведена в [8]. Вероятность отказа системы не должна превышать заданной величины $P_{max} \in [0,1]$:

$$P_{sys} = 1 - \prod_{v \in \mathcal{V}}(1 - P_v) \leq P_{max}. \quad (14)$$

где P_v – вероятность отказа v-го устройства. Считаем, что $P_v = const$. В реальных системах вероятность отказа является функцией времени: $P_v = f(\mathrm{t})$ [9].

Также запишем условия согласования обработчиков. У обработчика не может быть вышестоящего обработчика:

$$y_v + y_{\pi(v)} \leq 1, \quad \forall v \neq v_1^1. \quad (15)$$

У обработчика не может быть обработчиков в его поддереве:

$$y_v + \sum_{v' \in \mathcal{D}'(v)} y_{v'} = 1, \quad \forall v \in \mathcal{V} : \; y_v = 1. \quad (16)$$

На пути от любого листа к корню должен быть ровно один обработчик:

$$\sum_{v' \in \mathcal{P}(v)} y_{v'} = 1, \quad \forall v \in \mathcal{L}, \quad (17)$$

где $\mathcal{P}(v)$ – путь от листа v к корню (включая v и корень).

4. Целевая функция

Ранее [10, 11] были определены ключевые критерии оптимизации иерархической структуры распределенной системы управления.

Определяющим фактором при создании системы является ее стоимость. Стоимость системы складывается из капитальных и операционных затрат. Если принять, что затраты на создание одного узла системы являются постоянными, суммарные финансовые затраты на реализацию системы определяются как сумма затрат на каждый узел.

Оптимальная иерархическая структура $\mathcal{G}_O$ должна минимизировать стоимость системы при указанных в предыдущем параграфе ограничениях. Таким образом, оптимальная структура может быть определена в результате решения задачи оптимизации:

$$C_O = \min_{\mathcal{G}} \sum_{v \in \mathcal{V}} C_v, \quad (18)$$

где C_O – оптимальная суммарная величина затрат на создание и эксплуатацию системы.

Таким образом, задача структурной оптимизации формулируется следующим образом. Необходимо подобрать такую иерархическую структуру из узлов заранее заданных типов, чтобы доставить минимум целевой функции (18) при заданных ограничениях (4)…(17). Считаем, что можно использовать неограниченное количество устройств каждого типа. Принимаем также, что искомая структура представляет собой однородное дерево, и путь от любого листа к корню всегда содержит одно и то же количество узлов.

5. Выбор метода и программная реализация

Описанная в параграфах 3-4 задача, является задачей комбинаторной оптимизации при множественных ограничениях. Данная задача является NP-трудной так как может быть при некоторых допущениях сведена к NP-полной задаче о точном покрытии. Для решения задачи могут быть применены методы целочисленного программирования или эвристические алгоритмы.

Методы целочисленного программирования требовательны к ресурсам и сложно масштабируются, поэтому могут быть неэффективны при больших размерностях задачи. Для подобных задач большой размерности могут быть применены эвристические и метаэвристические алгоритмы, такие как генетический алгоритм и его модификации, алгоритм муравьиных колоний, роя частиц, имитации отжига и др. Для решения данной задачи выбран алгоритм муравьиной колонии, так как он достаточно хорошо кодирует задачу, представленную в виде графа, в отличие от генетических алгоритмов, ориентированных на системы с постоянной структурой. Алгоритм муравьиных колоний и варианты его практического применения подробно описаны в [12, 13], сравнение генетического алгоритма и алгоритма муравьиных колоний дано в [14].

Для реализации алгоритма была разработана программа на языке Python, способная решать численные примеры при достаточно больших размерностях. Исходные данные для численного примера приведены в таблице 1. В примере рассматривается случай, когда имеется набор из A одинаковых контуров с параметрами $n_1 = 1$, $r_1 = 1$, $w_1 = 5$. В качестве ограничений приняты максимальное время обработки контура в системе $T_{max} = 1\ c$ и максимальная вероятность отказа системы $P_{max} = 0{,}2$.

Таблица 1. Исходные данные для численного примера

Тип устройства u_i	***Стоимость C_i, усл. ед.***	***Кол-во каналов N_i, шт.***	***Объем памяти R_i, усл. ед.***	***Вероятность отказа P_i***	***Время обработки инструкции T_i, с***	***Признак обработчика y_i***	***Макс. кол-во дочерних узлов M_i, шт.***	***Временная задержка τ_i, с***
u_1	*1000*	*0*	*512*	*0,01*	*0,01*	*1*	*4*	*0*
u_2	*80*	*8*	*0*	*0,005*	*0*	*0*	*4*	*0,1*

Обработчиком u_1 может быть, например, программируемый логический контроллер, имеющий достаточно высокую стоимость, но не имеющий каналов ввода-вывода. Ретранслятором u_2 могут быть модуль ввода-вывода, к которому можно подключить до восьми физических сигналов, или сетевой коммутатор. Оба типа имеют на борту сетевые порты (4 шт.) для подключения нижестоящих устройств.

Результаты работы программы при различных значениях A и S сведены в таблицу 2.

Таблица 2. Результаты работы программы

Кол-во контуров управления, A,	***Кол-во уровней в иерархии S***	***Стоимость системы, усл. ед.***	***Мин. время обработки контура, с***	***Вероятность отказа системы***	***Кол-во обработчиков, шт.***	***Кол-во ретрансляторов, шт.***
2	*3*	*1160*	*0,12*	*0,0199*	*1*	*2*
10	*3*	*1240*	*0,2*	*0,0248*	*1*	*3*
30	*3*	*2400*	*0,25*	*0,0442*	*2*	*5*
50	*3*	*3640*	*0,28*	*0,0678*	*3*	*8*
100	*3*	*7360*	*0,31*	*0,1354*	*6*	*17*
150	*3*	-	-	-	-	-
150	*4*	*11000*	*0,38*	*0,1941*	*9*	*25*

Примечательно, что при $A = 150$ и $S = 3$ алгоритм не нашел допустимого решения вследствие имеющихся ограничений.

Графическое изображение оптимальных структур для некоторых значений A и S показано на рисунках 3, 4.

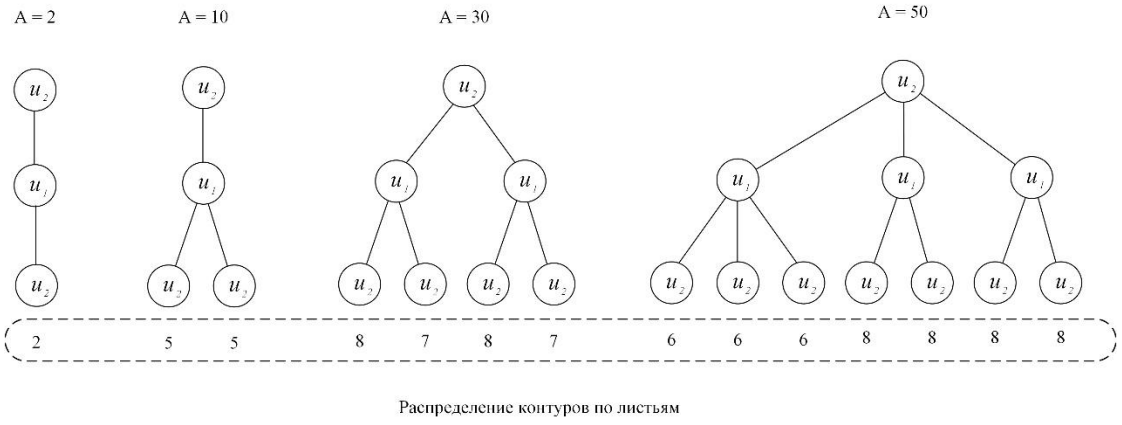

Рис. 3. Структура системы при $A = 2$, $A = 10$, $A = 30$ и $A = 50$ для $S = 3$

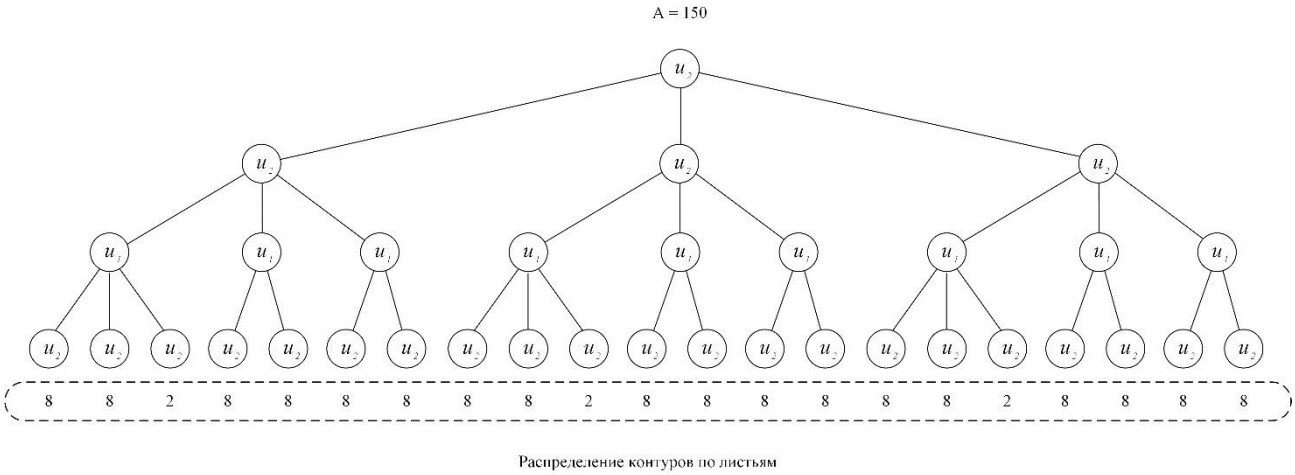

Рис. 4. Структура при $A = 150$ для $S = 4$

Алгоритм муравьиных колоний в чистом виде является приближенным алгоритмом и не дает точного решения задачи, однако на базе данного алгоритма можно сгенерировать опорное решение, которое можно улучшить, применив один из известных методов [6]. Для повышения качества результатов алгоритм муравьиных колоний может быть модифицирован путем добавления, например, локального поиска.

Решение численного примера показало, что примененный алгоритм имеет свойство застревать на локальных оптимумах, что особенно ярко выражается с ростом размерности задачи. Результаты эксперимента по определению оптимальной структуры при различных входных значениях количества подключаемых к системе контуров управления A показаны на рис. 5.

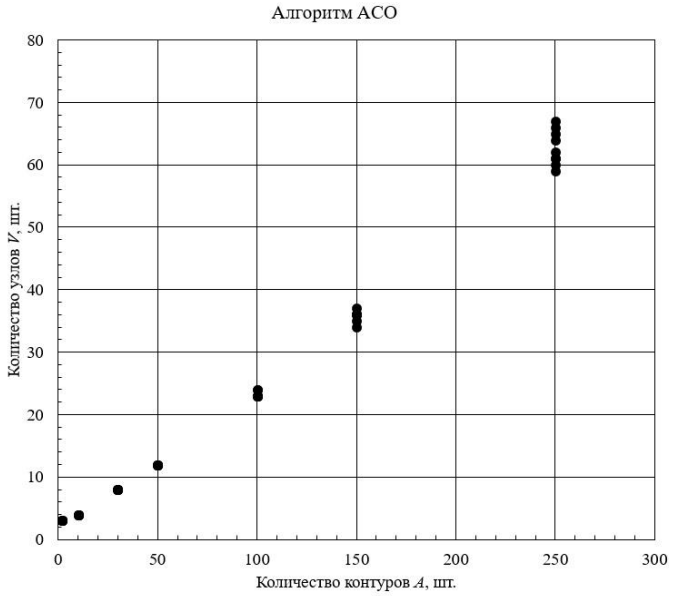

Рис. 5. Результаты работы алгоритма

Из рисунка видно, что алгоритм находит решение задачи в окрестности оптимального, однако, отклонение увеличивается с ростом количества элементов в структуре. Для минимизации эффекта застревания на локальных оптимумах необходимо подбирать параметры алгоритма, либо комбинировать его с каким-либо дополнительным методом [15].

6. Дальнейшие исследования

Для получения практически ценных результатов алгоритм решения задачи может быть доработан с учетом рекомендаций, данных в предыдущем параграфе. Следует иметь в виду, что решение задачи

приведено при определенных допущениях, таки как отсутствие горизонтальных связей между узлами иерархии, предопределенное разбиение массива входных и выходных сигналов на контуры, отсутствие задержки при передаче данных обработчиками, независимость отказов узлов системы и др. Учет указанных допущений в модели может являться предметом дополнительных исследований, развивающих и обобщающих результаты настоящей статьи.

Проверка качества полученной системы управления может быть выполнена путем имитационного моделирования системы с конкретным объектом управления, например, технологической установкой химического предприятия. Примечательно, что полученная система будет являться системой управления с телекоммуникационными каналами связи [16], анализ и синтез которой также являются предметами отдельного исследования.

Заключение

Задача поиска оптимальной структуры системы, построенной на базе серийно выпускаемых промышленностью компонентов, является чрезвычайно актуальной. В настоящее время синтез структуры таких систем выполняется в основном эмпирическим путем на основании опыта проектировщика и рекомендаций производителей оборудования и не всегда является оптимальным. Предложенная формализация задачи и способ решения позволяют сформировать алгоритм синтеза оптимальной структуры по стоимости при ограничениях на информационную мощность (количество каналов), память, быстродействие и надежность. Реализованный алгоритм может быть оформлен в виде программного продукта и использован при работе инженерами-проектировщиками АСУТП при проектировании крупномасштабных систем управления непрерывными технологическими процессами.

Литература

1. *Тверской, Ю. С.* О наукоемких этапах технологии создания территориально-распределенных многофункциональных АСУТП на базе ПТК сетевой иерархической структуры / Ю. С. Тверской, А. Н. Никоноров // Состояние и перспективы развития электротехнологии. XVII Бенардосовские чтения: Материалы Международной научно-технической конференции, Иваново, 29–31 мая 2013 года / Ивановский государственный энергетический университет имени В. И. Ленина. Том 2. – Иваново: Ивановский государственный энергетический университет им. В.И. Ленина, 2013. – С. 265-268.
2. *Федоров, Ю. Н.* Справочник инженера по АСУТП: Проектирование и разработка. Учебно-практическое пособие. - М.: Инфра-Инженерия, 2008. -928 стр., 12 ил.
3. *Ицкович, Э. Л.* Термины автоматизации и цифровизации предприятий технологических отраслей: Их пояснение, содержание и практическое значение / Э. Л. Ицкович // Автоматизация в промышленности. – 2020. – № 4. – С. 3-11.
4. *Воронин, А. А.* Оптимальные иерархические структуры / А. А. Воронин, С. П. Мишин. – Москва: Институт проблем управления им. В.А. Трапезникова РАН, 2003. – 214 с.
5. *Губко, М. В.* Модель выбора оптимальной древовидной иерархии / М. В. Губко, А. И. Даниленко, М. И. Сапико // Известия Орловского государственного технического университета. Серия: Информационные системы и технологии. – 2006. – № 1-1. – С. 46-50.
6. *Новиков, Д. А.* Методы оптимизации структуры иерархических систем / Д. А. Новиков, М. В. Губко // Управление развитием крупномасштабных систем: Современные проблемы / Под редакцией А.Д. Цвиркуна. Том Выпуск 2. – Москва: Общество с ограниченной ответственностью Издательская фирма «Физико-математическая литература», 2015. – С. 359-377.
7. *Цвиркун, А. Д.* Основы синтеза структуры сложных систем / А. Д. Цвиркун // М.: Наука, 1982. - 200 с.
8. *Акимова, Г. П.* Методология оценки надежности иерархических информационных систем / Г. П. Акимова, А. В. Соловьев // Труды Института системного анализа Российской академии наук. – 2006. – Т. 23. – С. 18-47.
9. *Ястребенецкий, М. А.* Надежность автоматизированных систем управления технологическими процессами: Учеб. Пособие для вузов – М.: Энергоатомиздат, 1989. – 264 с.: ил.
10. *Закирзянов, Р. М.* Критерии выбора оптимальной структуры распределенной системы управления технологическими процессами крупных промышленных предприятий / Р. М. Закирзянов // Математические методы в технологиях и технике. – 2024. – № 10. – С. 17-23.
11. *Риттер, А. В.* К вопросу выбора критериев оценки средств промышленной автоматизации для тепловой и атомной промышленности / А. В. Риттер, С. Б. Чебышов // Ядерная физика и инжиниринг. – 2013. – Т. 4, № 7. – С. 648.
12. *M. Dorigo, T. Stützle*, 2004. Ant Colony Optimization, MIT Press. ISBN 0-262-04219-3.

13. *Штовба, С. Д.* Муравьиные алгоритмы: теория и применение / С. Д. Штовба // Программирование. – 2005. – Т. 31, № 4. – С. 3-18.
14. *Семенкина, О. Е.* О сравнении эффективности муравьиного и генетического алгоритмов при решении задач комбинаторной оптимизации / О. Е. Семенкина, Е. С. Семенкин // Актуальные проблемы авиации и космонавтики. – 2011. – Т. 1, № 7. – С. 338-339.
15. *Емельянов, А. Ю.* Исследование и оптимизация муравьиного алгоритма / А. Ю. Емельянов, М. К. Семенов // Решетневские чтения: Материалы XXVII Международной научно-практической конференции, посвященной памяти генерального конструктора ракетно-космических систем академика М. Ф. Решетнева: в 2-х частях, Красноярск, 08–10 ноября 2023 года. – Красноярск: Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева», 2023. – С. 62-64.
16. *Zhang, W., Branicky, M. S., & Phillips, S. M.* (2001). Stability of networked control systems. IEEE Control Systems Magazine, 21(1), 84-97.