УДК 681.5:004.4
DOI: 10.15827/0236-235X.128.744-749
Дата подачи статьи: 26.06.19
2019. Т. 32. № 4. С. 744–749

Моделирование и анализ программ многомерных интервально-логических регуляторов

А.Ф. Антипин [1], *к.т.н., доцент, andrejantipin@ya.ru*
Е.В. Антипина [1], *к.ф.-м.н., младший научный сотрудник, stepashinaev@ya.ru*

[1] *Стерлитамакский филиал Башкирского государственного университета, г. Стерлитамак, 453103, Россия*

В статье рассматривается специальное ПО для моделирования работы многомерных нечетких интервально-логических регуляторов и анализа их программ для контроллеров с программируемой логикой, которые могут использоваться на предприятиях химической, нефтяной и нефтеперерабатывающей промышленности при разработке АСУ технологическими процессами и объектами, не имеющими адекватных математических моделей. Актуальность разработки ПО обусловлена отсутствием прикладных программ для моделирования работы нечетких регуляторов по имеющимся экспериментальным данным.

Описанное в статье ПО позволяет рассчитать необходимое и достаточное количество продукционных правил и критически важных правил, составляющих систему продукционных правил. Кроме того, по имеющимся исходным данным, полученным в результате экспериментов, можно построить нечеткие модели работы многомерных нечетких интервально-логических регуляторов.

В статье приведены результаты вычислительного эксперимента по созданию нечеткой модели работы многомерных нечетких интервально-логических регуляторов и анализу их программ для контроллеров с программируемой логикой, в ходе которого рассчитаны основные параметры регуляторов данного типа: максимальное количество продукционных правил, составляющих систему продукционных правил; общее количество термов и количество критически важных термов для каждой из переменных; общее количество групп переменных и количество критически важных групп переменных; максимальное количество продукционных правил, количество критически важных продукционных правил для каждой из групп переменных и фактическое количество критически важных правил, входящих в систему продукционных правил.

По результатам расчетов сделаны выводы о сложности системы продукционных правил многомерных нечетких интервально-логических регуляторов и достижении требуемой точности вычислений.

Ключевые слова: *программное обеспечение, многомерный интервально-логический регулятор, моделирование, система продукционных правил, нечеткая логика.*

Регуляторы с четкими термами, или *многомерные интервально-логические регуляторы* (МИЛР), являются одной из разновидностей нечетких регуляторов с прямоугольной функцией принадлежности переменных [1–3]. В МИЛР ключевые для нечетких регуляторов понятия «фаззификация» и «дефаззификация» заменены на альтернативные термины «интервализация» и «деинтервализация» соответственно.

Под интервализацией в МИЛР понимается процесс определения принадлежности значений переменных конкретным интервалам (термам), входящим в их диапазон значений, а под деинтервализацией – обратный процесс выделения четких значений переменных МИЛР из интервалов (термов), найденных в конкретный момент времени [3].

Основными проблемами, возникающими в процессе разработки программ для *программируемого логического контроллера* (ПЛК-программ) с использованием МИЛР и нечетких регуляторов в целом, являются определение необходимого и достаточного количества правил, наиболее точно описывающих *объект управления* (ОУ) или подлежащий автоматизации процесс, а также наличие критически важных правил вне зависимости от режимов работы АСУ. Современные прикладные пакеты программ позволяют имитировать работу нечетких регуляторов только после полной настройки их параметров, что в некоторых случаях представляет собой достаточно длительный процесс, который к тому же необходимо многократно повторить [4–10].

Решить описанные проблемы может специальное ПО, разработанное в Стерлитамакском филиале Башкирского государственного университета и предназначенное для моделирования и анализа ПЛК-программ как самих много-

мерных интервально-логических регуляторов, так и систем автоматического регулирования на их основе, в кратчайшие сроки и по имеющимся экспертным данным, описывающим какой-либо технологический процесс или объект, подлежащий автоматизации [11, 12].

ПО для анализа ПЛК-программы МИЛР

ПО для анализа ПЛК-программы МИЛР (рис. 1) предназначено для автоматизации расчетов по предварительной оценке программ МИЛР для контроллеров с программируемой логикой, входящих в состав АСУ. В данном ПО реализован авторский метод расчета необходимого и достаточного числа продукционных правил, а также критически важных правил.

Логически программный продукт состоит из двух частей, реализованных на отдельных вкладках (рис. 1): ввод основных параметров и анализ переменных и *системы продукционных правил* (СПП).

На вкладке «Ввод основных параметров» (рис. 1а) задается общее количество переменных МИЛР с последующей настройкой таких свойств, как имя, тип (входная или выходная), рабочий диапазон значений и количество интервалов (или термов). Имеется возможность пометить отдельные переменные как критические. В соседних таблицах определяются взаимосвязи между переменными и граничные значения термов с пометкой «критический».

На вкладке «Анализ переменных и СПП» (рис. 1б) выводятся результаты автоматического анализа переменных и СПП МИЛР.

К ним относятся:

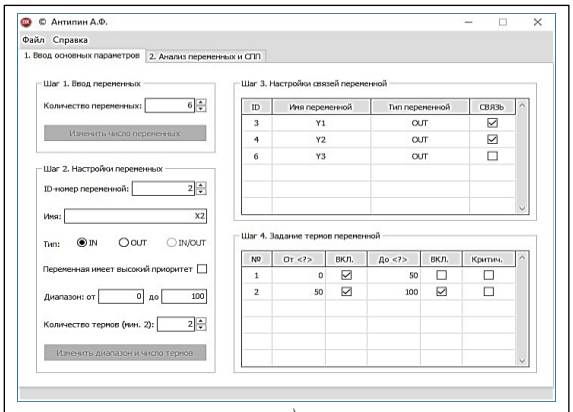
1) максимальное количество продукционных правил СПП: $R_{\max} = \prod_{i=1}^{n} k(x_i)$, где $k(x_i)$ – количество термов (или интервалов разбиения) i-й входной переменной x МИЛР;

2) общее количество термов $k(x)$, а также количество критически важных термов $k_{\text{кр.}}(x)$ для каждой из переменных;

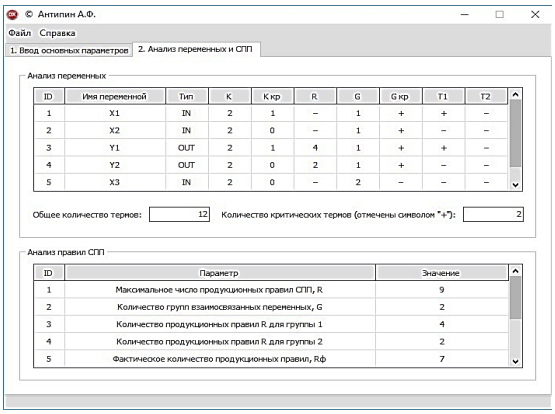
3) общее количество, а также количество критически важных групп переменных G и $G_{\text{кр.}}$ соответственно;

4) максимальное количество и количество критически важных правил $R_{\text{гр.}}$ и $R_{\text{гр.кр.}}$, соответственно, для каждой из групп переменных:

$R^{g}_{\text{гр.}} = \prod_{i=1}^{n_g} k(x_i)$, где n_g – количество входных переменных x МИЛР, входящих в группу под номером g;



а)



б)

Рис. 1. ПО для анализа ПЛК-программы:
а) ввод основных параметров,
б) анализ переменных и СПП

Fig. 1. Software for the analysis of programmable logic controller program:
a) input of main parameters,
б) variables and production system analysis

$R^{g}_{\text{гр.кр.}} = \prod_{i=1}^{n_g} k(x_i) - \prod_{i=1}^{n_g} \left(k(x_i) - k_{\text{кр.}}(x_i) \right)$, где $k_{\text{кр.}}(x)$ – количество критических (сигнальных или аварийных) термов, определенных для конкретных входных переменных x, входящих в группу под номером g;

5) фактическое количество критически важных правил СПП: $R_{\text{ф.кр.}} = \sum_{i=1}^{G} R^{i}_{\text{гр.кр.}} + 1$;

6) номер группы, в которую входит та или иная переменная, и пр.

ПО для моделирования работы МИЛР

Данный программный продукт, внешний вид которого показан на рисунках (см. http://www.swsys.ru/uploaded/image/2019-4/2019-4-dop/12.jpg), предназначен для создания нечеткой модели работы МИЛР.

На первом шаге работы с приложением по аналогии с предыдущим ПО задается общее количество переменных МИЛР, после чего выполняется их конфигурирование, связанное с указанием имен переменных, выбором их типа (входная, или IN; выходная, или OUT) и настройкой взаимных связей. При необходимости можно вручную выполнить интерпретацию переменных термами, то есть задать диапазон значений, указать количество термов и их интервалы и пр. Далее вводится, как минимум, один набор экспертных данных. Чем больше данных будет введено, тем точнее описание МИЛР и, как следствие, сама модель.

На рисунке (см. http://www.swsys.ru/uploaded/image/2019-4/2019-4-dop/12.jpg) показан фрагмент одного из продукционных правил СПП, генерируемой автоматически после ввода исходных данных.

Вычислительный эксперимент и тестирование ПО

Рассмотрим примеры оценки ПЛК-программы и моделирования работы МИЛР с использованием описанного ПО.

В таблице 1 приведены параметры МИЛР с тремя входными переменными x и двумя выходными переменными y. Схема взаимосвязей переменных МИЛР, представленная на рисунке 2, показывает, что входная переменная x_1 имеет взаимную связь с выходной переменной y_1, а входные переменные x_2 и x_3 – с выходной переменной y_2.

Таблица 1

Основные параметры регулятора

Table 1

Key controller parameters

Входные/ выходные переменные	**Количество термов k**	**Количество критических термов $k_{кр.}$**
x_1	3	1
x_2	3	1
x_3	3	1
y_1	5	2
y_2	5	2

По окончании ввода параметров МИЛР из таблицы 1 в приложение для анализа ПЛК-программы МИЛР получены следующие результаты:

1) общее количество термов:

$$K = \sum_{i=1}^{3} k(x_i) + \sum_{j=1}^{2} k(y_j) = 19;$$

2) количество критических термов $K_{кр.} = 7$;

3) максимальное количество продукционных правил, составляющих СПП:

$$R_{\max} = \prod_{i=1}^{3} k(x_i) + 1 = 28;$$

4) количество групп взаимосвязанных переменных: $G = 2$;

5) максимальное количество продукционных правил для группы 1:

$$R^1_{гр.} = k(x_1) = 3;$$

6) максимальное количество продукционных правил для группы 2:

$$R^2_{гр.} = k(x_2)k(x_3) = 9;$$

7) фактическое количество продукционных правил СПП:

$$R_{ф.} = \sum_{i=1}^{2} R^i_{гр.} + 1 = 13;$$

8) количество критически важных продукционных правил СПП для группы 1:

$$R^1_{гр.кр.} = k(x_1) - \left(k(x_1) - k_{кр.}(x_1)\right) = k_{кр.}(x_1) = 1;$$

9) количество критически важных продукционных правил СПП для группы 1:

$$R^2_{гр.кр.} = k(x_2)k(x_3) - \left(k(x_2) - k_{кр.}(x_2)\right) \times$$

$$\times \left(k(x_3) - k_{кр.}(x_3)\right) = 5;$$

10) фактическое количество критических продукционных правил:

$$R_{ф.кр.} = \sum_{i=1}^{2} R^i_{гр.кр.} + 1 = 7.$$

Внешний вид окна приложения для анализа ПЛК-программы МИЛР с результатами произведенных расчетов приведен на рисунке (см. http://www.swsys.ru/uploaded/image/2019-4/2019-4-dop/13.jpg).

Расчеты показали, что при максимально возможном числе продукционных правил, равном 28, при разработке СПП достаточно описать только 13 правил, 7 из которых являются критически важными для работы.

Для создания нечеткой модели МИЛР необходимы экспериментальные данные [13, 14],

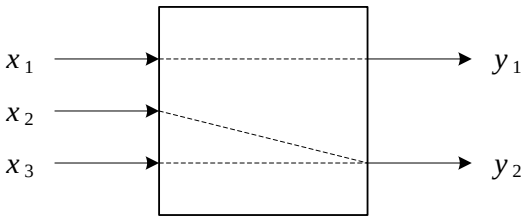


Рис. 2. Схема взаимосвязей переменных

Fig. 2. Variables relations diagram

приведенные в таблице 2, которые вместе с параметрами МИЛР вносят в соответствующие поля приложения для моделирования работы МИЛР.

Таблица 2

Экспериментальные данные

Table 2

Experimental data

Переменная	Эксперимент					
	1	**2**	**3**	**4**	**5**	**6**
x_1	0	3	6	9	12	15
x_2	0	0	3	3	6	6
x_3	3	9	3	9	3	9
y_1	0	27	108	243	432	675
y_2	-9	-27	6	-12	21	3

Из данных таблицы 2 следует, что входные переменные x_1 и x_2 имеют прямой вид зависимости с выходными переменными y_1 и y_2, а переменная x_3 – обратный. Зависимость входных и выходных переменных МИЛР выявляется при отключении всех контуров регулирования, кроме исследуемого.

В процессе моделирования МИЛР определяются диапазоны значений входных и выходных переменных, как в таблице 3.

Таблица 3

Диапазоны значений переменных

Table 3

Ranges of values for variables

x_1		x_2		x_3		y_1		y_2	
$x_н$	$x_н$	$x_к$	$x_н$	$x_к$	$x_к$	$y_н$	$y_к$	$y_н$	$y_к$
0	15	0	6	3	9	0	675	-27	21

Далее приложение осуществляет расчет согласно методике, описанной в [12].

В результате расчетов определяются коэффициенты a_y, которые в дальнейшем используются для определения четких значений выходных переменных y МИЛР:

$$y^{(t)} = y_н^{(t)} + a_y \left(y_к^{(t)} - y_н^{(t)} \right).$$

Таким образом,

$$y_1 = 675 \cdot a_1,$$

$$y_2 = -27 + 48 \cdot a_2,$$

$$a_1 = \mathrm{pos}^2(x_1^{(t)}),$$

$$a_2 = \frac{3 + 5 \cdot \mathrm{pos}(x_2^{(t)}) - 3 \cdot \mathrm{pos}(x_3^{(t)})}{8},$$

где $\mathrm{pos}(x_1^{(t)})$, $\mathrm{pos}(x_2^{(t)})$, $\mathrm{pos}(x_3^{(t)})$ – функции, определяющие позиции текущих значений входных переменных x МИЛР в границах их термов.

Протестируем работу нечеткой модели при следующих значениях входных переменных МИЛР: $x_1 = 10$, $x_2 = 5$, $x_3 = 4$.

В качестве проверочных функций используются выражения:

$$y_1^к = 3 \cdot x_1^2,$$

$$y_2^к = 5 \cdot x_2 - 3 \cdot x_3.$$

Так, для указанных ранее значений входных переменных МИЛР

$$\mathrm{pos}(x_1^{(t)}) = 0,67, \quad \mathrm{pos}(x_2^{(t)}) = 0,83,$$

$$\mathrm{pos}(x_3^{(t)}) = 0,17, \quad a_1 = 0,44, \; a_2 = 0,83,$$

$$y_1 = y_1^к = 300, \quad y_2 = y_2^к = 13,$$

что позволяет говорить о достижении требуемой точности вычислений.

Заключение

Описанное в статье специальное ПО позволяет выполнить анализ ПЛК-программ МИЛР, рассчитать необходимое и/или достаточное количество продукционных правил, составляющих СПП, и количество критически важных продукционных правил, а также построить нечеткую модель работы МИЛР. Последнее доступно только в тех случаях, когда возможно проведение ряда экспериментов, связанных с отключением отдельных контуров регулирования. Применение приведенных выражений имеет смысл, если не известен точный вид функций зависимости выходных переменных y от входных переменных x МИЛР, поскольку дальнейшее повышение точности вычислений сводится в итоге или к увеличению общей совокупности термов, которыми интерпретируются переменные МИЛР, или к изменению вида выражений для расчета a_y и/или $y^{(t)}$.

Данное ПО может применяться специалистами, связанными с разработкой АСУ сложными технологическими процессами или объектами, не имеющими адекватной математической модели, и при грамотном использовании существенно сократить ручной труд, обеспечить быструю разработку надежных и эффективных ПЛК-программ МИЛР, а также уменьшить сроки разработки и сопровождения систем управления на их основе.

Литература

1. Леоненков А.В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. СПб: БХВ-Петербург, 2003. 719 с.

2. Усков А.А., Сургучева И.В., Горбунов А.М. Анализ систем обработки информации и управления с помощью групповых нечетких чисел // Программные продукты и системы. 2009. № 3. С. 19–21.

3. Антипин А.Ф. Особенности настройки взаимосвязей и составления системы продукционных правил в интервально-логическом регуляторе // Информационные системы и технологии. 2015. № 1. С. 5–13.

4. Седова Н.А. Нечеткая продукционная модель первичной оценки опасности столкновения судов // Мир транспорта. 2015. Т. 13. № 2. С. 200–206.

5. Sugeno M. On stability of fuzzy systems expressed by fuzzy rules with singleton consequents. IEEE Transactions on Fuzzy Systems, 1999, vol. 7, pp. 201–224.

6. Yongming Li, Yali Li, Zhanyou M. Computation tree logic model checking based on possibility measures. Fuzzy Sets and Systems, 2015, vol. 262, pp. 44–59.

7. Степашина Е.В. Оптимизация финансовых показателей предприятия на основе нейросетевой модели // Информационные системы и технологии. 2014. № 5. С. 34–42.

8. Подколзин А.С. Компьютерное моделирование логических процессов. Архитектура и языки решателя задач. М.: Физматлит, 2008. 1024 с.

9. Седова Н.А., Седов В.А. Логико-лингвистическая модель оценки уровня аварийных ситуаций // Современная наука: актуальные проблемы теории и практики. 2016. № 2. С. 65–69.

10. Круглов В.В., Дли М.И. Интеллектуальные информационные системы: компьютерная поддержка систем нечеткой логики и нечеткого вывода. М.: Физматлит, 2002. 256 с.

11. Антипин А.Ф. Об оценке программ контроллеров с программируемой логикой // Автоматизация, телемеханизация и связь в нефтяной промышленности. 2018. № 11. С. 41–46. DOI: 10.30713/0132-2222-2018-11-41-46

12. Антипин А.Ф. Способ повышения точности дефаззификации в многомерных нечетких интервально-логических регуляторах // Автоматизация, телемеханизация и связь в нефтяной промышленности. 2017. № 4. С. 24–28.

13. Степашина Е.В., Байтимерова А.И., Мустафина С.А. О свойствах решений задач моделирования каталитических процессов с переменным реакционным объемом // Журнал Средневолжского математического общества. 2010. Т. 12. № 3. С. 122–128.

14. Пегат А. Нечеткое моделирование и управление; [пер. с англ. А.Г. Подвесовского, Ю.В. Тюменцева]. М.: БИНОМ. Лаборатория знаний, 2013. 798 с.

Software & Systems
DOI: 10.15827/0236-235X.128.744-749

Received 26.06.19
2019, vol. 32, no. 4, pp. 744–749

Modeling and analysis of programs for multidimensional interval-logic controllers

A.F. Antipin [1], *Ph.D. (Engineering), Associate Professor, andrejantipin@ya.ru*
E.V. Antipina [1], *Ph.D. (Physics and Mathematics), Junior Researcher, stepashinaev@ya.ru*

[1] *Sterlitamak Branch of Bashkir State University, Sterlitamak, 453103, Russian Federation*

Abstract. The paper examines special software designed for modeling the operation of multidimensional fuzzy interval-logic controllers and analysis of their programs for programmable logic controllers. They can be used at enterprises of chemical, oil and oil refining industries in the development of automatic control systems by technological processes and objects that do not have adequate mathematical models. The relevance of software development arises from the lack of application programs designed to simulate operation of fuzzy controllers according to available experimental data.

The software described in the paper allows calculating the necessary and sufficient number of production rules and the number of critically important rules that make up a production system. In addition, according to available initial data obtained as a result of experiments, it is possible to create fuzzy models of multidimensional fuzzy interval-logic controllers.

The paper presents the results of a computational experiment in creating a fuzzy model of multidimensional fuzzy interval-logic controllers and analyzing their programs for programmable logic controllers, during which the main parameters of this type of controllers were calculated: the maximum number of production rules that make up the production system; the total number of terms and the number of critically important terms for each

variable; the total number of variable groups and the number of critically important variable groups; the maximum number of production rules, the number of critical production rules for each of variable groups and the actual number of critical rules that make up the production system.

Based on calculation results, conclusions were drawn regarding the complexity of a production system for multidimensional fuzzy interval-logic controllers and about how to achieve the required calculation accuracy.

Keywords: software, multidimensional interval-logic controller, simulation, production system, fuzzy logic.

References

1. Leonenkov A.V. *Fuzzy Modeling in MATLAB and FuzzyTECH*. St. Petersburg, BHV-Petersburg Publ., 2003, 719 p. (in Russ.).
2. Uskov A.A., Surguchova I.V., Gorbunov A.M. Analysis of the systems of treatment of information and control by means group fuzzy numbers. *Programmnye Produkty i Sistemy* [*Software & Systems*]. 2009, no. 3, pp. 19–21 (in Russ.).
3. Antipin A.F. Features of the setup of interlinkages and the creation of the system of condition-action rules in interval-logic controller. *Information Systems and Technologies*. 2015, no. 1, pp. 5–13 (in Russ.).
4. Sedova N.A. Fuzzy production model for initial evaluating of the risk of collisions. *World of Transport and Transportation*. 2015, vol. 13, no. 2, pp. 200–206 (in Russ.).
5. Sugeno M. On stability of fuzzy systems expressed by fuzzy rules with singleton consequents. *IEEE Transactions on Fuzzy Systems*. 1999, vol. 7, pp. 201–224.
6. Yongming Li, Yali Li, Zhanyou M. Computation tree logic model checking based on possibility measures. *Fuzzy Sets and Systems.* 2015, vol. 262, pp. 44–59.
7. Stepashina E.V. Optimization the financial performance of enterprise on neural network model. *Information Systems and Technologies*. 2014, no. 5, pp. 34–42 (in Russ.).
8. Podkolzin A.S. *Computer Simulation of Logical Processes. Architecture and Problem Solver Languages.* Moscow, Fizmatlit Publ., 2008, 1024 p. (in Russ.).
9. Sedova N.A., Sedov V.A. The logical-linguistic model for assessing of the emergency level. *Modern Science: Actual Problems of Theory and Practice*. 2016, no. 2, pp. 65–69 (in Russ.).
10. Kruglov V.V. *Intelligent Information Systems: Computer Support for Fuzzy Logic and Fuzzy Inference Systems.* Moscow, Fizmatlit Publ., 2002, 256 p. (in Russ.).
11. Antipin A.F. Programs evaluation of controllers with programmable logics. *Automation, Telemechanization and Communication in Oil Industry*. 2018, no. 11, pp. 41–46 (in Russ.).
12. Antipin A.F. The method of de-fuzzification accuracy increase in multi-dimensional fuzzy interval-logic regulators. *Automation, Telemechanization and Communication in Oil Industry*. 2017, no. 4, pp. 24–28 (in Russ.). DOI: 10.30713/0132-2222-2018-11-41-46.
13. Stepashina E.V., Baytimerova A.I., Mustafina S.A. The problem of modelling catalytic processes with varying reaction volume and the properties of solutions. *Middle Volga Mathematical Society J*. 2010, vol. 12, no. 3, pp. 122–128 (in Russ.).
14. Piegat A. *Fuzzy Modeling and Control*. 2001. Rus. ed.: Moscow, BINOM. Laboratoriya znany Publ., 2013, 798 p.

Для цитирования

Антипин А.Ф., Антипина Е.В. Моделирование и анализ программ многомерных интервально-логических регуляторов // Программные продукты и системы. 2019. Т. 32. № 4. С. 744–749. DOI: 10.15827/0236-235X.128.744-749.

For citation

Antipin A.F., Antipina E.V. Modeling and analysis of programs for multidimensional interval-logic controllers. *Software & Systems*. 2019, vol. 32, no. 4, pp. 744–749 (in Russ.). DOI: 10.15827/0236-235X.128.744-749.

УДК 004.382.2
DOI: 10.15827/0236-235X.128.750-758
Дата подачи статьи: 09.09.19
2019. Т. 32. № 4. С. 750–758

Алгоритм формирования множества вариантов структуры суперкомпьютера в интересах решения военно-прикладных задач

Я.Н. Гусеница [1], *к.т.н., начальник испытательной лаборатории информатики и вычислительной техники, yaromir226@gmail.com*
Д.О. Петрич [2], *к.т.н., старший преподаватель кафедры программно-алгоритмического обеспечения автоматизированных систем управления, pdo_1985@mail.ru*

[1] *Военный инновационный технополис «ЭРА», г. Анапа, 353456, Россия*
[2] *Военно-космическая академия им. А.Ф. Можайского, г. Санкт-Петербург, 197198, Россия*

В работе обоснована необходимость создания и применения суперкомпьютеров для решения военно-прикладных задач. Дано обоснование разработки качественно нового научно-методического аппарата обоснования архитектур суперкомпьютеров, основанного на принципах унификации, комплексирования и интеграции и обеспечивающего максимальную производительность.

Описана архитектура суперкомпьютера с помощью модели коллектива вычислителей, включающей структуру и алгоритм работы коллектива вычислителей. Показано, что структура суперкомпьютера должна представлять собой программно-неделимый аппаратный ресурс при организации вычислительного процесса. Поэтому формирование структуры суперкомпьютера целесообразно производить по принципу модульной наращиваемости из однотипных базовых модулей (вычислительных кластеров), которые, с одной стороны, имеют конструктивные ограничения, а с другой, должны формироваться для конкретных военно-прикладных задач. Каждый вычислительный кластер включает некоторое сбалансированное количество элементарных процессоров и блоков памяти, объединенных с помощью полнодоступной системы быстрых каналов и способных реализовать базовые наборы различных задач.

Представлен алгоритм, который, в отличие от существующих, основан на использовании спектра графа при генерации множества структур связей между вычислительными кластерами, а также элементарными процессорами в составе вычислительных кластеров. Это позволяет значительно уменьшить вычислительную сложность формирования множества вариантов структуры суперкомпьютера.

Предлагаемый алгоритм может быть использован для обоснования архитектурных решений при создании новых суперкомпьютеров и выделения вычислительных ресурсов под решение военно-прикладных задач на существующих суперкомпьютерах.

Ключевые слова: *суперкомпьютер, варианты структуры, алгоритм, матрица смежности, спектр графа.*

По мнению военных экспертов, уровень развития информатики и вычислительной техники может стать одним из главных факторов достижения успеха в военных конфликтах будущего, определяя технический облик перспективного вооружения [1].

Данное обстоятельство обусловливает наличие большого количества отечественных и зарубежных научно-исследовательских организаций, высших учебных заведений, а также коммерческих предприятий, активно занимающихся теорией и практикой в области информатики и вычислительной техники.

Анализ научно-технической литературы показывает, что одним из важнейших направлений в области информатики и вычислительной техники является создание и применение суперкомпьютеров [2–5]. Это направление во всем мире относится к факторам стратегического потенциала оборонного, научно-технического и народнохозяйственного назначения [1].

Создание и применение суперкомпьютеров в военной сфере обусловлены объективной потребностью в огромных вычислительных ресурсах для решения таких военно-прикладных задач, как

– обеспечение проведения научной работы (обеспечение выполнения в установленном порядке научно-исследовательских и опытно-конструкторских работ);

– информационная поддержка органов военного управления (мониторинг и прогнозирование состояния войск (сил), военно-политической и общественно-политической обстановки);

– численное моделирование (моделирование поражающих факторов, сложных динамических объектов);

– имитационное моделирование (моделирование обеспечения войск (сил), боевых действий);

– виртуальное моделирование образцов вооружения, военной и специальной техники на основе технологий цифровых двойников и дополненной реальности;

– обработка неструктурированной и слабоструктурированной информации на основе технологий больших данных.

Важнейшей аспектом при создании и применении суперкомпьютеров в военном деле является разработка архитектурных решений, обеспечивающих максимальную производительность на широком классе военно-прикладных задач [1]. Постоянное изменение качественного состава военно-прикладных задач, технологий производства вычислительных систем, алгоритмов обработки информации обусловливает требование к непрерывному совершенствованию научно-методического аппарата проектирования вычислительных систем специального назначения. Поэтому разработка качественно нового научно-методического аппарата обоснования архитектур суперкомпьютеров, основанного на принципах унификации, комплексирования и интеграции и обеспечивающего максимальную производительность, является одной из важнейших научных задач в военном деле.

Представление архитектуры суперкомпьютера с помощью модели коллектива вычислителей

Архитектура суперкомпьютера как любой другой вычислительной системы – совокупность средств и правил, обеспечивающих взаимодействие устройств обработки информации и программ [6–8].

Разработка архитектуры суперкомпьютера подразумевает решение следующих вопросов:

– выбор типов применяемых элементов и определение технических характеристик каждого из них;

– выбор методов управления и синхронизации системы;

– выбор стратегии передачи информации между элементами суперкомпьютера;

– выбор типа каналов связи;

– деление на разделяемые и индивидуальные ресурсы;

– адаптация суперкомпьютера к алгоритмам.

Каноническую основу архитектуры суперкомпьютера может составить модель коллектива вычислителей, которая представляется парой:

$$S = <H, A>, \quad (1)$$

где H и A – конструкция и алгоритм работы коллектива вычислителей [9].

Конструкция коллектива вычислителей описывается в виде

$$H = <C, G>, \quad (2)$$

где $C = \{c_i\}$ – множество вычислителей $\{c_i\}$, $i = 0(1)N–1$; N – мощность множества C; G – описание структуры коллектива вычислителей (сети связей между вычислителями).

Конструкция коллектива вычислителей представляет отражение следующих основополагающих архитектурных принципов [9–11].

1. Параллелизм при обработке информации (параллельного вычисления операций на множестве C вычислителей, взаимодействующих посредством связей структуры G).

2. Программируемость структуры (изменение структуры G на основе программного управления).

3. Однородность конструкции H (однородности вычислителей $C = \{c_i\}$ и структуры G).

Алгоритм A работы коллектива S обеспечивает работу всех вычислителей $C = \{c_i\}$ и структуры G в процессе решения общей задачи. Данный алгоритм может быть представлен в виде суперпозиции

$$A(P(D)), \quad (3)$$

где D – исходный массив данных, подлежащих обработке в процессе решения задачи.

В свою очередь,

$$D = \bigcup_{i=0}^{N-1} D_i, \quad (4)$$

где D_i – индивидуальный массив данных для вычислителя $c_i \in C$.

Причем в общем случае

$$\bigcap_{i=0}^{N-1} D_i \neq \varnothing, \quad (5)$$

где P – параллельная программа решения общей задачи.

Параллельная программа может быть представлена в виде нескольких ветвей программы:

$$P = \bigcup_{i=0}^{N-1} P_i, \ \bigcap_{i=0}^{N-1} P_i = \varnothing, \quad (6)$$

где P_i – i-я ветвь программы P.

Особенности представления структуры суперкомпьютера

Структура суперкомпьютера должна представлять собой программно-неделимый аппаратный ресурс при организации вычислительного процесса. Поэтому формирование структуры суперкомпьютера целесообразно производить по принципу модульной наращиваемости из однотипных базовых модулей – *вычислительных кластеров* (ВК), *которые,* с одной стороны, имеют конструктивные ограничения, а с другой, должны формироваться для конкретных военно-прикладных задач.

Каждый ВК включает некоторое сбалансированное количество *элементарных процессоров* (ЭП) и блоков памяти, объединенных с помощью полнодоступной системы быстрых каналов и способных реализовать, желательно целиком, базовые наборы различных задач.

Схематически общая структурная схема ВК изображена на рисунке 1.

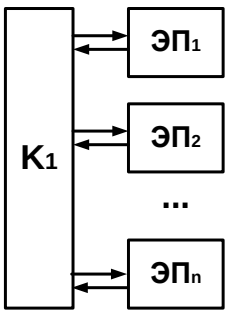


Рис. 1. Укрупненная структурная схема ВК

Fig. 1. An integrated structural diagram of a computing cluster

Структура суперкомпьютера в целом должна формироваться по принципу модульной наращиваемости путем объединения нескольких ВК с помощью коммутационной среды K_2, соблюдая принцип иерархичности. Данный принцип позволяет увеличить производительность суперкомпьютера при увеличении количества ВК. В состав суперкомпьютера могут входить разнотипные ВК, а также интерфейсы (I_1, ..., I_q) внешних вычислительных ресурсов, связанных с суперкомпьютером.

Схематически структурная схема суперкомпьютера изображена на рисунке 2.

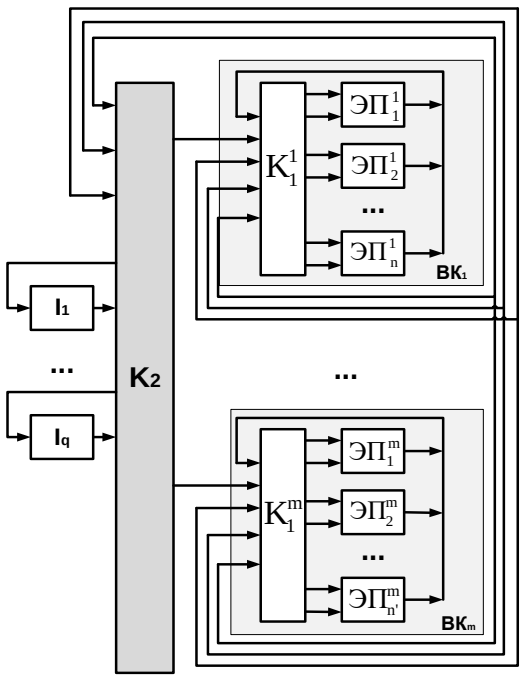


Рис. 2. Укрупненная структурная схема суперкомпьютера

Fig. 2. The enlarged supercomputer structural diagram

Каждый вариант структуры суперкомпьютера представляет собой совокупность аппаратных и программных ресурсов и математически описывается следующим образом.

В основе варианта лежит элемент множества **U**, который представляет собой вектор параметров:

$$U_i = <(G_k)_i, (G_p)_i>, \ i = 1(1)O(\mathbf{U}), \quad (7)$$

где G_k – совокупность аппаратных ресурсов уровня суперкомпьютера (ВК и связей между ними); G_p – совокупность аппаратных ресурсов уровня ВК (ЭП и связей между ними).

В свою очередь, $(G_k)_i$ представляет собой следующий вектор параметров:

$$(G_k)_i = <(N_k)_i, (S_k)_i, (M_{kcom})_i>, \ i = 1(1)O(\mathbf{U}), \quad (8)$$

где N_k – количество ВК, входящих в состав суперкомпьютера; M_{kcom} – тип устройства межкластерной коммуникации; $\mathbf{S}_k$ – матрица смежности графа, описывающего структуру связей между ВК.

$\mathbf{S}_k$ представляет собой симметричную квадратную матрицу размером $N_k \times N_k$ неориентированного графа, описывающего структуру связей ВК между собой. Данная матрица формируется следующим образом:

$$(S_k)_{ij} = \begin{cases} 1, & \text{если элементы } i \text{ и } j \text{ связаны}, \\ 0, & \text{если элементы } i \text{ и } j \text{ не связаны}, \end{cases} \quad (9)$$

где $i, j = 1(1)N_k$ – количество ВК, входящих в состав суперкомпьютера.

Каждый элемент $(G_p)_i$ представляет собой следующий вектор параметров:

$$(G_p)_i = <(M_u)_i, (N_u)_i, (M_s)_i, (N_s)_i, (M_{pcom})_i, (\mathbf{S}_p)_i>, \; i = 1(1)O(\mathbf{U}), \quad (10)$$

где M_u – тип универсального процессора, входящего в состав каждого ВК; N_u – количество универсальных процессоров, входящих в состав каждого ВК; M_s – тип специализированного процессора, входящего в состав каждого ВК; N_s – количество специализированных процессоров, входящих в состав каждого ВК; M_{pcom} – тип среды межпроцессорной коммуникации; $\mathbf{S}_p$ – матрица смежности графа, описывающего структуру связей между специализированными процессорами в составе каждого ВК.

$\mathbf{S}_p$ представляет собой симметричную квадратную матрицу размером $N_s \times N_s$ неориентированного графа, описывающего структуру связей между специализированными процессорами в составе каждого ВК. Данная матрица формируется следующим образом:

$$\left(\mathbf{S}_p\right)_{ij} = \begin{cases} 1, & \text{если элементы } i \text{ и } j \text{ связаны}, \\ 0, & \text{если элементы } i \text{ и } j \text{ не связаны}, \end{cases} \quad (11)$$

где $i, j = 1(1)N_s$.

Формирование множества вариантов структуры суперкомпьютера осуществляется в следующей последовательности.

1. Генерируется множество допустимых вариантов структур связей между ВК.

2. Генерируется множество допустимых вариантов структур связей между ЭП в составе ВК.

3. Формируется множество допустимых вариантов устройств межпроцессорной коммуникации, совместимых с ЭП.

4. Формируется множество комбинаций для описанных выше элементов.

Генерация множества допустимых вариантов структур связей между ВК, а также процессорами в составе ВК сводится к формированию матриц $\mathbf{S}_k$ и $\mathbf{S}_p$. Данная задача относится к классу комбинаторных задач. Поскольку $\mathbf{S}_k$ и $\mathbf{S}_p$ являются матрицами смежности неориентированных графов, в силу их симметричности относительно главной диагонали можно вычислить зависимость числа сгенерированных комбинаций от числа ВК по формуле

$$M = 2^{\sum_{i=1}^{N-1}(N-i)}, \quad (12)$$

где M – число сгенерированных вариантов; N – число комбинируемых ВК.

ВК должны быть связаны между собой какой-либо коммуникационной средой. Данное условие математически определяется свойством связности графа, описывающего структуру суперкомпьютера.

Проверка связности некоторого графа G с вершинами $v_1, v_2, \ldots, v_n$ и матрицей смежности $\mathbf{A}$ осуществляется на основе проверки условия

$$\hat{\mathbf{A}}^I_{ij} = 1, \; i, j = \overline{1,n}, \quad (13)$$

здесь $$\hat{\mathbf{A}}^I = \mathbf{I} \vee \mathbf{A} \vee \mathbf{A}^{\otimes 2} \vee \mathbf{A}^{\otimes 3} \vee \ldots \vee \mathbf{A}^{\otimes n}, \quad (14)$$

где $\mathbf{I}$ – единичная мультипликативная матрица; $\otimes$ – операция булева произведения матриц [12].

Для математического описания структуры вычислительной системы, как правило, используют матрицу смежности. Однако матрица смежности не может служить исчерпывающим описанием структуры суперкомпьютера, поскольку жестко привязывается к нумерации вершин. Это приводит к тому, что некоторые одинаковые по топологии варианты структуры суперкомпьютера, графы которых изоморфны, обладают отличными матрицами смежности. Данное обстоятельство обусловливает необходимость распознавания изоморфизма и изоморфного вложения сложных структур, заданных в форме матриц или графов.

С содержательной точки зрения изоморфизм структур означает тождественность их функционирования, что в некоторых случаях приводит к возможности замены одной структуры другой, изоморфной ей. Однако для распознавания изоморфизма применяется алгоритм полного перебора, что делает проблему изоморфизма практически нерешаемой уже при сравнительно небольшом числе элементов данной структуры. Легко видеть, что для распознавания изоморфизма графов, которые имеют n вершин, требуется в общем случае выполнить $n!$ попарных сравнений, то есть при относительно небольшом количестве элементов в графах (около 100) решение задачи об изоморфизме методом полного перебора невозможно.

Для решения задачи определения изоморфизма графов в качестве топологической характеристики графа, инвариантной к нумерации его вершин, предлагается использовать спектр графа.

Спектр отражает общие свойства всех тех графов, матрицы смежности которых преобразуются друг в друга посредством некоторой невырожденной матрицы [9].

Обыкновенным спектром графа G называют спектр его матрицы смежности $\mathbf{A}$.

Использование спектра графа обусловлено следующим. Каждому графу G однозначно соответствует класс $\mathbf{A} = A(G)$ матриц смежности. Две матрицы смежности $\mathbf{A}$ и $\mathbf{A}'$ определяют один и тот же граф тогда и только тогда, когда существует такая матрица перестановки $\mathbf{P}$, что $\mathbf{A} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}$. Следовательно, важным инвариантом класса $\mathbf{A}$ является

$$\mathrm{SP}(G) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}, \quad (15)$$

где λ_i – корни уравнения $P_G(\lambda) = 0$.

Таким образом, для определенного класса графов с ограничением на количество вершин и на имеющиеся сведения о неизоморфных коспектральных графах возможно решение задачи определения изоморфизма графов на основе использования одного из инвариантов графов – их спектров [9, 13, 14].

Вычисление спектра графа по сути своей является определением собственных значений его матрицы смежности. Существуют следующие методы вычисления.

1. Метод непосредственного развертывания, который используется для вычисления собственных значений матриц невысокого порядка ($n \leq 10$). Вычислительная сложность алгоритма, реализующего данный метод, описывается порядком $O(n^3)$ при решении задачи развертывания дискриминанта и $O(\log_2 n)$ при условии решения нелинейных уравнений методом дихотомии. Таким образом, сложность алгоритма составляет порядка $O(n^3)$.

2. Метод итераций, который применяется для решения частичной проблемы собственных значений матрицы (на его основе можно определить приближенно собственные значения матрицы). Вычислительная сложность алгоритма, реализующего данный метод, описывается порядком $O(2n(n + 1)L)$. Здесь L – количество итераций алгоритма, зависящее от установленной точности вычислений.

Очевидно, что даже такая высокая вычислительная сложность реализуемых алгоритмов является более предпочтительной, чем сложность $O(n!)$, соответствующая алгоритму полного перебора.

Алгоритм формирования множества вариантов структуры суперкомпьютера

Для формирования множества вариантов структуры суперкомпьютера разработан оригинальный алгоритм, основанный на использовании свойств спектра графа.

Исходные и выходные данные разработанного алгоритма представлены в таблице.

Исходные и выходные данные

Input and output data

Обозначение	Наименование параметра
Исходные данные	
$N_{\text{вк}}$	Количество военно-прикладных задач
N_u	Максимальное количество универсальных процессоров в ВК
N_s	Максимальное количество специализированных процессоров в ВК
$\mathbf{M}_u=\{(M_u)_i\}$, $i=1(1)O(\mathbf{M}_u)$	Множество типов универсальных процессоров
$O(\mathbf{M}_u)$	Количество типов универсальных процессоров
$\mathbf{M}_s=\{(M_s)_i\}$, $i=1(1)O(\mathbf{M}_s)$	Множество типов специализированных процессоров
$O(\mathbf{M}_s)$	Количество специализированных процессоров
$\mathbf{M}_{pcom}=\{(M_{pcom})_i\}$, $i=1(1)O(\mathbf{M}_{pcom})$	Множество типов устройств межпроцессорной коммуникации
$O(\mathbf{M}_{pcom})$	Количество типов устройств межпроцессорной коммуникации
$\mathbf{M}_{kcom}=\{(M_{kcom})_i\}$, $i=1(1)O(\mathbf{M}_{kcom})$	Множество типов устройств межкластерной коммуникации
$O(\mathbf{M}_{kcom})$	Количество типов устройств межкластерной коммуникации
$\mathbf{\Omega}_u$	Матрица совместимости универсальных процессоров и устройств межпроцессорной коммуникации
$\mathbf{\Omega}_s$	Матрица совместимости специализированных процессоров и устройств межпроцессорной коммуникации
$\mathbf{M}_{kcom}=\{(M_{ram})_i\}$, $i=1(1)O(\mathbf{M}_{ram})$	Множество типов модулей оперативной памяти
$O(\mathbf{M}_{ram})$	Количество типов модулей оперативной памяти
Выходные данные	
$\mathbf{U}=\{U_i\}$, $i=1(1)O(\mathbf{U})$	Множество, объединяющее параметры каждого варианта структуры суперкомпьютера
$O(\mathbf{U})$	Количество вариантов структуры суперкомпьютера

$\mathbf{\Omega}_u$ и $\mathbf{\Omega}_s$ представляют собой симметричные матрицы относительно главной диагонали размером $O(\mathbf{M}_u) \times O(\mathbf{M}_{pcom})$, $O(\mathbf{M}_s) \times O(\mathbf{M}_{pcom})$ соответственно. Они формируются следующим образом:

$$(\Omega_u)_{ij} = \begin{cases} 1, \text{если элементы } i \text{ и } j \text{ совместимы}, \\ 0, \text{если элементы } i \text{ и } j \text{ несовместимы}, \end{cases} \quad (16)$$

где $i = 1(1)O(\mathbf{M}_u)$, $j = 1(1)O(\mathbf{M}_{pcom})$.

$$(\Omega_s)_{ij} = \begin{cases} 1, \text{если элементы } i \text{ и } j \text{ совместимы}, \\ 0, \text{если элементы } i \text{ и } j \text{ несовместимы}, \end{cases} \quad (17)$$

где $i = 1(1)O(\mathbf{M}_s)$, $j = 1(1)O(\mathbf{M}_{pcom})$.

В качестве ограничений и допущений при разработке настоящего алгоритма рассматриваются следующие:

– все ВК осуществляют обработку информации параллельно и имеют непосредственную или косвенную связь друг с другом на основе общей магистрали (граф, описывающий структуру суперкомпьютера, является связным);

– каждый ВК должен представлять собой объединение однородных специализированных процессоров, находящихся под управлением одного или нескольких универсальных процессоров;

– все ЭП в составе ВК имеют непосредственную или косвенную связь друг с другом (граф, описывающий структуру ВК, является связным).

Алгоритм (см. http://www.swsys.ru/uploaded/image/2019-4/2019-4-dop/1.jpg) предполагает выполнение следующих шагов.

Шаг 1. Ввод исходных данных.

Шаг 2. Начало цикла по $i = 1, 2^{\sum_{j=1}^{N_{\text{вк}}-1} (N_{\text{вк}}-j)}$.

Шаг 3. Генерация матрицы смежности $(\mathbf{S}_k)_i$.

Шаг 4. Расчет матрицы $(\hat{\mathbf{S}}_k)_i$ для выявления свойства связности графа.

Шаг 5. Проверка свойства связности графа $((\hat{\mathbf{S}}_k)_i)_{lm} = 1,\ l, m = \overline{1, N_{\text{вк}}}$. Если данное условие выполняется, то осуществляется переход к шагу 6, в противном случае – к шагу 10.

Шаг 6. Расчет спектра $\text{SP}(\mathbf{S}_k)_i$ матрицы смежности $(\mathbf{S}_k)_i$ графа.

Шаг 7. Проверка факта совпадения рассчитанного спектра $\text{SP}(\mathbf{S}_k)_i$ со спектром одного из выбранных ранее графов. Если данное условие выполняется, осуществляется переход к шагу 8, в противном случае – к шагу 9.

Шаг 8. Проверка того, относится ли граф с матрицей смежности $(\mathbf{S}_k)_i$ к исключениям – семействам коспектральных неизоморфных графов или нет. Если данное условие выполняется, осуществляется переход к шагу 9, в противном случае – к шагу 10.

Шаг 9. Добавление $(\mathbf{S}_k)_i$ в множество допустимых вариантов $\mathbf{S}'_k$.

Шаг 10. Конец цикла по *i*.

Шаг 11. Начало цикла по $i = 1, 2^{\sum_{j=1}^{N^S-1} (N^S-j)}$.

Шаг 12. Генерация матрицы смежности $(\mathbf{S}_p)_i$ графа.

Шаг 13. Расчет матрицы $(\hat{\mathbf{S}}_p)_i$ для проверки связности графа, описывающего структуру ВК.

Шаг 14. Проверка свойства связности графа, описывающего структуру суперкомпьютера: $((\hat{\mathbf{S}}_p)_i)_{lm} = 1$; $l, m = 1(1)N_{\text{вк}}$. Если данное условие выполняется, осуществляется переход к шагу 15, в противном случае – к шагу 19.

Шаг 15. Расчет $\text{SP}((\mathbf{S}_p)_i)$ – спектра матрицы смежности $(\mathbf{S}_p)_i$ графа, описывающего структуру ВК.

Шаг 16. Проверка факта совпадения рассчитанного спектра $\text{SP}((\mathbf{S}_p)_i)$ со спектром одного из выбранных ранее графов. Если данное условие выполняется, осуществляется переход к шагу 17, в противном случае – к шагу 18.

Шаг 17. Проверка, относится ли граф с матрицей смежности $(\mathbf{S}_p)_i$ к исключениям – семействам коспектральных неизоморфных графов. Если данное условие выполняется, осуществляется переход к шагу 18, в противном случае – к шагу 19.

Шаг 18. Добавление $(\mathbf{S}_p)_i$ в множество допустимых вариантов структуры ВК $\mathbf{S}'_p$.

Шаг 19. Конец цикла по *i*.

Шаг 20. Выбор процессора общего назначения из множества $\mathbf{M}_u$.

Шаг 21. Начало цикла по $i = 1(1)O(\mathbf{M}_{pcom})$.

Шаг 22. Проверка совместимости выбранного процессора общего назначения с $(M_{pcom})_i$ устройством межпроцессорной коммуникации: $(\Omega_u)_{ki} = 1$, k – номер элемента из множества $\mathbf{M}_u$. Если данное условие выполняется, осуществляется переход к шагу 23, в противном случае – к шагу 24.

Шаг 23. Добавление $(M_{pcom})_i$ в множество допустимых вариантов структуры ВК $\mathbf{M}'_{pcom}$.

Шаг 24. Конец цикла по *i*.

Шаг 25. Проверка, является ли множество $\mathbf{M}'_{pcom}$ пустым ($\mathbf{M}'_{pcom} = \varnothing$). Если данное условие выполняется, осуществляется переход к шагу 20, в противном случае – к шагу 26.