УДК 004.414.2

СИНТЕЗ АЛГОРИТМОВ КОНТРОЛЯ БОРТОВЫХ ИНФОРМАЦИОННЫХ И УПРАВЛЯЮЩИХ СИСТЕМ ЛЕТАТЕЛЬНЫХ АППАРАТОВ ПО КРИТЕРИЮ МИНИМУМА ОБРАЩЕНИЙ К ЛОКАЛЬНОЙ ПАМЯТИ

Н.С. АКИНШИН, Е.А.СТАРОЖУК

Предложена методика формирования структуры локальной памяти бортовых информационных и управляющих систем. Формализованы задачи выбора набора программ для реализации множества алгоритмов и оптимизации структуры локальной памяти, которые относятся к классу задач дискретного программирования с псевдобулевыми переменными. На основе применения теории графов разработаны алгоритмы повышения эффективности использования сверхоперативной памяти бортовых информационных и управляющих систем.

Ключевые слова: бортовые информационные и управляющие системы, дискретная оптимизация, алгоритмы укладки графов

Средства обработки информации, принятия решений и формирования команд управления занимают центральное место и определяют специфику работы управляющей системы сложной технической системы как единого целого [1]. Эти средства представляют собой, как правило, совокупность вычислительных устройств бортовой информационной и управляющей системы (БИУС) различного функционального назначения, не связанных, слабо связанных, либо сильно связанных между собой и представляющих единую вычислительную систему. В связи с постоянным усложнением задач, которые необходимо решать комплексу управления воздушным движением (УВД), в качестве средств обработки информации все более полно будут использоваться управляющие вычислительные машины и системы машин, связанные между собой и составляющие, например, многомашинные комплексы, мультипроцессорные и, как наиболее перспективные, распределенные вычислительные системы. Применение на борту летательных аппаратов (ЛА) вычислительных систем, органически связанных между собой, должно обеспечить высокую эффективность функционирования средств УВД.

Эффективность функционирования БИУС во многом зависит от качества принятых конструкторских решений, в том числе в области проектирования систем памяти.

Методика выбора структуры и параметров локальной памяти БИУС заключается в определении такой структуры внутренней памяти и такого набора программ решения задач, при которых удовлетворяются все ограничения на параметры микропроцессорной системы (МПС), а выбранный критерий оптимальности достигает своего экстремального значения [2]. Суть предлагаемой методики состоит в следующем.

1. Выбирается критерий оптимальности - минимум суммарной емкости ОЗУ и ПЗУ, требующейся для хранения программ и данных. Для этого введятся булевы переменные x_{ij}:

$$x_{ij} = \begin{cases} 1, \text{ если алгоритм } L_i \in \tilde{L}, \text{ обработки сигнала } c_i \in C \text{ реализуется } j\text{-й программой;} \\ 0, \text{ в противном случае} \end{cases}$$

2. Формулируется задача выбора набора программ для реализации множества алгоритмов $\tilde{L}$ для МПС, работающих в составе вычислительных средств ИУС, где время работы строго ограничено и каждой задаче $c_i \in C$ соответствует свой алгоритм решения $L_i \in \tilde{L}$, *p-n*, путем минимизации целевой функции

$$\Phi(x) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} \left(a_{ij} + h_{ij}\right) x_{ij} \qquad (1)$$

при временных ограничениях на реализацию алгоритмов

$$\sum_{i \in C_1} \sum_{j=1}^{m_j} t_{ij} g_i x_{ij} \le T_1; \quad \sum_{i \in C_1} \sum_{j=1}^{m_j} t_{ij} g_i x_{ij} + \sum_{i \in C_2} \sum_{j=1}^{m_i} t_{ij} g_i x_{ij} \le T_2; \quad \sum_{l=1}^{u} \sum_{i \in C_l} \sum_{j=1}^{m_i} t_{ij} g_i x_{ij} \le T_l; \qquad (2)$$

при ограничениях на число ячеек СОЗУ и ОЗУ

$$\max_{L_i \in \tilde{L}} \left\{ \sum_{j=1}^{m_i} b_{ij} x_{ij} \right\} \le B; \ \sum_{i=1}^{n} \sum_{j=1}^{m_i} h_{ij} x_{ij} \le H; \tag{3}$$

при ограничениях на аддитивные параметры

$$\sum_{i=1}^{n} \sum_{j=1}^{m_i} r_{ijk} x_{ij} \le R_k, k = \overline{1, q}; \tag{4}$$

при логических условиях существования единственной программы реализации каждого алгоритма

$$\sum_{j=1}^{m_i} x_{ij} = 1, i = \overline{1, n}. \tag{5}$$

Здесь t_{ij} – время решения алгоритма L_i j – м методом; g_i – относительная частота появления алгоритма в общей программе; R_k, B и H – ограничения на k – ый ресурс, емкость сверхоперативной памяти (СОЗУ) и ОЗУ соответственно; T_l- максимально допустимое время реализации задач, имеющих l – ый приоритет. Задача (1)–(5) является задачей дискретного программирования с псевдобулевыми переменными [3,4].

3. Логические возможности q-ый программы определяются в виде множества упорядоченных пар $П^{(q)}=\{(i^{(q)}, j^{(q)})\}$, $i = \overline{1, p}$, $j = \overline{1, m_i}$, где $(i^{(q)}, j^{(q)})$ тогда q-ая программа способна реализовать алгоритм $L_i \in \tilde{L}$ j-ым способом.

4. Вводятся переменные y_q таким образом, чтобы

$$y_q = \begin{cases} 1, \text{если } q-\text{ая комплексная программа} \\ \text{входит в состав системы программ;} \\ 0 \text{ в противном случае.} \end{cases}$$

5. Для $П=\{П^{(1)}, ..., П^{(Q)}\}$- множества комплексных программ, задачу оптимизации структуры локальной памяти можно записать в виде минимизации функционала

$$\Phi(x, y) = \sum_{\substack{i=1 \\ (i,j) \notin П}}^{p} \sum^{m_i} \left(a_{ij} + h_{ij}\right) x_{ij} + \sum_{q=1}^{Q} \left(a_q + h_q\right) y_q \tag{6}$$

при ограничениях на число ячеек СОЗУ и ОЗУ

$$\max_{q \& (i,j) \notin П} \left\{ \sum_{j=1}^{m_i} b_{ij} x_{ij}; b_q y_q \right\} \le B; \ \sum_{\substack{i=1 \\ (i,j) \notin П}}^{p} \sum_{j=1}^{m_i} h_{ij} x_{ij} + \sum_{q=1}^{Q} h_q y_q \le H; \tag{7}$$

на аддитивные параметры

$$\sum_{\substack{i=1 \\ (i,j) \notin П}}^{p} \sum_{j=1}^{m_i} r_{ijk} x_{ij} + \sum_{q=1}^{Q} r_{qk} y_q \le R_k; \tag{8}$$

на единственность решения задачи

$$\sum_{(i,j) \notin П^{(q)}} x_{ij} - M_q y_q \le 0, q = \overline{1, Q}; \ \sum_{j=1}^{m_i} x_{ij} + \sum_{q=1}^{Q} y_q \ge 1, y_q + z_q = 1; q = \overline{1, Q}. \tag{9}$$

на время реализации алгоритмов

$$\sum_{l=1}^{p} \left[\sum_{\substack{i \in C_l \\ (i,j) \notin П}} \sum_{j=1}^{m_i} t_{ij} g_i x_{ij} + \sum_{q \in C_l} t_q g_q y_q \right] \le T_l. \tag{10}$$

Ограничения (9) в задаче (6)-(10) показывают, что $x_{ij}=0$ в тех случаях, когда имеется хотя бы одна пара $(i,j)\in П^{(q)}$, $j=\overline{1,m_i}$ при $y_q=1$; z_q являются псевдопеременными и позволяют сохранить задачу оптимизации ($z_q=1$, когда $y_q=0$); M_q – произвольное целое число, большее мощности множества $П^{(q)}$.

Здесь Q_q – общее число ячеек ОЗУ или ПЗУ, требующихся для записи q-й комплексной программы; b_q и h_q – соответственно число ячеек СОЗУ и ОЗУ, необходимых для хранения промежуточных результатов, а также исходных, некоторых промежуточных и вспомогательных данных при реализации q-й комплексной программы; r_{qk} – количество ресурса типа k, требующееся для реализации q-й программы.

Можно оптимизировать структуру локальной памяти по различным критериям с учетом ограничений, накладываемых на другие параметры системы.

В общем случае МПС должна реализовать множество задач контроля C за заданное время, при этом каждый микропроцессорный модуль MM_l предназначен для решения в пределах заданного времени подмножества задач $C_l \subset C$. Для каждой пары "процессор – секция локальной памяти" множество возможных программ задается функциональным графом G [5,6], на котором определено отношение предшествования $\angle$, причем запись $A_i \angle A_j$ означает, что оператор A_j использует результирующую величину оператора A_i. Это отношение порождает орграф $G=(X,U)$, где $U=\{(A_i,A_j) \mid (A_i \angle A_j)\}$. Далее предполагается, что в графе G имеется единственная входная вершина (что не влияет на общность результатов) и отсутствуют контуры и параллельные дуги.

Укладка графа $G=(X,U)$ – такая последовательность $L(G)=\left(A_{i_1}, A_{i_2},..., A_{i_n}\right)$ всех вершин графа, что для каждой дуги $\left(A_{i_k}, A_{i_p}\right)\in U$ имеет место $k < p$. Будем говорить, что вершины A_{i_k} и $A_{i_{k+1}}$ находятся в отношении соседней связности в укладке L, если $\left(A_{i_k}, A_{i_{k+1}}\right)\in U$.

Число соседней связности укладки L – величина

$$\chi(\mathrm{L})=\sum_{k=1}^{n-1}\delta\left(A_{i_k}, A_{i_{k+1}}\right), \text{ где } \delta\left(A_{i_k}, A_{i_{k+1}}\right)=\begin{cases}1, & \text{если } \left(A_{i_k}, A_{i_{k+1}}\right)\in U\\ 0 & \text{в противном случае.}\end{cases}$$

Компонента укладки – последовательность вершин этой укладки $C=\left(A_{j_1},..., A_{j_t}\right)$, удовлетворяющая условиям $\delta\left(A_{j_1-1}, A_{j_1}\right)=0;\quad \delta\left(A_{j_t}, A_{j_t+1}\right)=0;\ \delta\left(A_{j_p}, A_{j_p+1}\right)=1,\quad p=\overline{1,t-1}.$

Число соседней связности графа G-$\chi(\mathrm{Q})=\max\limits_{L\in R(G)}\{\chi(\mathrm{L})\}$, где $R(G)$ – множество всех укладок графа G.

Задача заключается в определении оптимальной укладки $L_*\in R(G)$, для которой $\chi(L_*)=\chi(G)$. Дугу $(A_i,A_j)\in U$ будем называть избыточной, если в графе G существует, хотя бы один путь μ из $A_i\in X$ в A_j, длины $l(\mu)\geq 2$. Здесь и далее под длиной $l(\mu)$ пути μ понимается количество дуг в этом пути. Назовем A – преобразованием графа G граф $G'=A(G)$, полученный из G удалением всех избыточных дуг. Путь $\mu=\left(A_{i_1},..., A_{i_p}\right)$ графа G проникающим, если

а) $d^-(A_{i1})=0$, где $d^-(A_i)$ – число дуг, заходящих в вершину A_i;

б) в графе G отсутствуют дуги вида $\left(A_y, A_{i_q}\right)$, где $q=\overline{1,p},\ A_y\in X\setminus\bigcup\limits_{q=1}^{p}A_{i_q}$.

Предложен алгоритм укладки графа, основанный на отыскании максимальных проникающих путей.

Алгоритм 1.

Начать с графа $G_1=A(G)$, где A – символ A-преобразования.

1. На k-м шаге в графе G_k определяется множество M_k проникающих путей и среди

них выбирается путь $\mu^*_k \in M_k$, имеющий максимальную длину.

2. Из графа G_k удаляется путь μ^*_k и образуется новый граф $G_{k+1}=G_k \backslash \mu^*_k$.

Алгоритм заканчивается, когда на некотором шаге $G_m=\varnothing$. Полученные на каждом шаге пути μ^*_k отождествляются с компонентами k-го шага $\boldsymbol{C}_k$ искомой укладки. Тогда последовательность $\Delta=(\mu^*_1, ..., \mu^*_{m-1})$ представляет собой укладку графа G, определенную с помощью алгоритма 1. При определении максимального проникающего пути μ^*_k удобно использовать алгоритм 2, основанный на пометке вершин графа G_k.

Алгоритм 2.

Первоначально все вершины графа $G_k=(X_k, U_k)$ считаются непомеченными и непросмотренными.

1. В графе G_k входные вершины A_x, у которых $d^-(A_x)=0$, получают метку $\lambda(A_x)=1$. После этого вершины A_x считаются помеченными и непросмотренными.

2. Пусть A_x – некоторая помеченная и непросмотренная вершина в G_k. Рассмотрим множество вершин $Y=\{A_y \mid (A_x,A_y)\in U_k\}$. Если $d^-(A_y)=1$, то вершина $A_y \in Y$ получает пометку $\lambda(A_y)=\lambda(A_x)+1$ и считается помеченной и непросмотренной. Если $d^-(A_y)>1$, то вершина не помечается. После анализа всех $A_y \in Y$ вершина A_x считается просмотренной. Процесс пометки заканчивается, когда все помеченные вершины просмотрены.

3. В графе G_k выбирается вершина A_{i_p}, для которой пометка $\lambda\left(A_{i_p}\right)$ максимальна. Тогда путь $\mu^*_k=\left(A_{i_1},..., A_{i_p}\right)$ с последовательно возрастающими пометками вершин является искомым максимальным проникающим путем.

Алгоритм 2 решает задачу с оценкой в $O(n^2)$ действий. Описанный ниже алгоритм 3 является усложнением алгоритма 1 и позволяет во многих случаях повысить точность получаемых решений за счет введения операции прогноза на один шаг.

Алгоритм 3.

Начать с графа $G_1=A(G)$.

1. В графе G_k определяется множество проникающих путей M_k.

2. Для каждого пути $\mu \in M_k$ образуется граф $G_k(\mu)=G_k \backslash \mu$ и в этом графе отыскивается максимальный проникающий путь $\pi^*_k(\mu)$.

3. Среди всех путей $\mu \in M_k$ выбирается путь μ^*_k, для которого максимальна величина $l\left(\mu^*_k\right)+l\left(\pi^*_k\left(\mu^*_k\right)\right)$, где $l(\mu^*_k)$ и $l(\pi^*_k(\mu^*_k))$ – длины соответствующих путей.

После этого образуется новый граф $G_{k+1}=G_k \backslash \mu^*_k$. Описанная процедура заканчивается на шаге m, для которого $G_m=\varnothing$. Аналогичным образом могут быть построены эвристические алгоритмы укладки графов с прогнозом на большее число шагов, однако при этом резко возрастает их трудоемкость. Исследования показывают, что алгоритм 1 дает точное решение для укладки программ, являющихся прадеревьями с одним корнем.

Определение верхней границы числа $\chi(G)$ сводится к определению минимального цепного разложения графа G.

Для определения нижней границы числа $\chi(G)$ достаточно найти число висячих вершин в прадереве G_T. В соответствии с алгоритмом 4 определяются оценки γ числа $r\left(\overline{G}_T\right)$ сверху: $\gamma \geq r\left(\overline{G}_T\right)$.

Алгоритм 4.

Начать с графа G_1, полученного из графа G путем введения фиктивной вершины z_1 и дуги (z_1, A_z), где A_z – входная вершина графа G.

1. Среди выходных вершин графа $G_k=(X_k, U_k)$ выбирается произвольная вершина A_{x_*} и для нее определяется величина $\varepsilon_k\left(A_{x_*}\right)=l\left(\mu_*\left(z_k, A_{x_*}\right)\right)$, где $\mu_*(z_k, A_{x_*})$ – кратчайший путь из z_k

в A_{x_*}.

2. В графе G_k находится множество $I_k(A_{x_*})$ вершин, из которых достижима A_{x_*}, и строится граф G'_k путем удаления из G_k подграфа, образованного множеством вершин $I_k(A_{x_*}) \cup \{A_{x_*}\}$.

3. Из графа G'_k конструируется новый граф $G_{k+1}=(X_{k+1},U_{k+1})$ по следующим правилам: вводится фиктивная вершина z_{k+1}, вводятся дуги вида (z_{k+1},A_y), если существуют дуги (A_w, A_y), где $A_w \notin G'_k$, $A_y \in G'_k$.

Описанная процедура повторяется до тех пор, пока на некотором шаге не выполнится условие $G_m=\varnothing$. Тогда величина $\gamma = n - \sum_{A_{x_*}} \varepsilon_k(A_{x_*})$ является искомой верхней границей числа $r(\overline{G}_T)$. Здесь суммирование производится по вершинам A_{x_*}, выбранным на всех шагах алгоритма. Следует отметить, что в зависимости от очередности выбора вершины A_{x_*} на k-м шаге алгоритма 4 верхняя оценка может меняться. На практике хорошие результаты получаются при выборе такой вершины A_{x_*}, для которой значение $\varepsilon_k(A_{x_*})$ максимально.

Наряду с разработкой алгоритмов и методов укладки графов по критерию максимальной соседней связности большой практический интерес представляют верхние и нижние оценки числа соседней связности. Их можно применять как для предварительного анализа программ с точки зрения эффективности использования СОЗУ, так и для оценки точности решений, получаемых с помощью эвристических алгоритмов укладки.

Таким образом, рассмотренная модель проектирования памяти БИУС позволяет определять минимально необходимую емкость СОЗУ, обеспечивающую реализацию алгоритма без обращения к ОЗУ за операндами и дополнительными данными.

ЛИТЕРАТУРА

1. Зубков Б.В., Прозоров С.Е. Безопасность полётов: учебник для вузов / под ред. Б.В. Зубкова. М.: МГТУ ГА, 2011.

2. Лебедев В.А., Терсков В.А. Моделировние и оптимизация многопроцессорных систем оперативного управления. М.: МАКС Пресс, 2002. - 330 с.

3. Антамошкин А.Н. Регулярная оптимизация псевдобулевых функций. -Красноярск: Изд-во КГУ, 1979. 160 с.

4. Донской В.И. Задачи псевдобулевой оптимизация с дизъюнктивным ограничением // Журнал вычислительной математики и мат.физики. - 1994. - Т.34, №3. - С.389-398.

5. Оре О. Теория графов. – М.: Наука, 1980.

6. Харари Ф. Теория графов, пер. с англ., М., 1973.

Synthesis of control algorithms on-board information and control systems of aircraft at the minimum criterion of accesses to local memory

Akinshin N.S., Staroguk E.A.

The technique of formation of the structure of the local memory on-Board information and control systems. Formalized the problem of choosing a set of programs to implement many of the algorithms and optimizing the structure of the local memory that belong to the class of problems of discrete programming with pseudoboolean variables. Based on the application of graph theory algorithms improve the efficient use of cache memory on-Board information and control systems.

Keywords: on-Board information and control systems, discrete optimization, algorithms stacking graphs

REFERENCES

1. Zubkov B.V., Prozorov S.E. Bezopasnost' polyotov: uchebnik dlya vuzov / pod red. B.V. Zubkova. M.: MGTU GA, 2011.

2. Lebedev V.A., Terskov V.A. Modelirovnie i optimizatsiya mnogoprotsessornyh sistem operativnogo upravleniya. M.: MAKS Press, 2002. - 330 s.

3. Antamoshkin A.N. Regulyarnaya optimizatsiya psevdobulevyh funktsiy. -Krasnoyarsk: Izd-vo KGU, 1979. 160 s.

4. Donskoy V.I. Zadachi psevdobulevoy optimizatsiya s diz"yunktivnym ogranicheniem // ZHurnal vychislitel'noy matematiki i mat.fiziki. - 1994. - T.34, №3. - S.389-398.

5. Ore O. Teoriya grafov. – M.: Nauka, 1980.

6. Xarari F. Teoriya grafov, per. s angl., M., 1973.

Сведения об авторах

Акиншин Николай Степанович, 1950 г.р., окончил ВАА им.Калинина (1978), профессор, доктор технических наук, заслуженный деятель науки РФ, нач.отдела АО ЦКБА г.Тула, автор более 300 научных работ, область научных интересов – радиотехнические системы, системотехника.

Старожук Евгений Андреевич, 1969 г.р. к.э.н., окончил МВТУ им. Н.Э. Баумана (1991), проректор по экономике МВТУ им. Н.Э. Баумана, автор более 50 работ, область научных интересов - информационная безопасность, математическое моделирование.