Оптимизация состава технологических переходов при многопозиционной автоматической обработке

М.А. Анфёров, e-mail: anfyorov@inbox.ru

«МИРЭА - Российский технологический университет»

Аннотация. *Рассматривается метод оптимизации структуры технологических операций наукоемкого производства, выполняемых на многопозиционном автоматическом оборудовании, использующий динамическое программирование и специально разработанные алгоритмы.*

Ключевые слова: *моделирование, оптимизация, динамическое программирование, САПР ТП, многопозиционная обработка.*

Введение

Высокий уровень интеграции информационных систем, задействованных в процессе жизненного цикла продукции в рамках CALS-стратегии [1], выдвигает соответствующие требования к уровню автоматизации поддерживаемых ими процессов. В отношении наукоемких изделий это, в первую очередь, относится к технической подготовке производства [2]. При этом наиболее трудной формализации поддается процесс автоматизированного проектирования технологических процессов ввиду значительной многовариантности их структуры, определяемой выбором состава технологических операций, этапов и методов обработки поверхностей, технологических баз, средств технологического оснащения и т.д., а также сложного влияния сочетания технологических воздействий на эксплуатационные свойства поверхностей деталей. Это определяет высокое значение моделей и алгоритмов, используемых в современных системах автоматизированного проектирования технологических процессов (САПР ТП) [2].

В работе рассматривается алгоритмизация оптимального формирования структуры технологических операций, требующих синхронизации машинного времени при многопозиционной обработке заготовок на автоматическом оборудовании: многошпиндельных токарных станках полуавтоматах, поточных автоматических и роторно-конвейерных линиях, роботизированных гибких автоматических модулях. В отличие от стохастической оптимизации с использованием

генетического алгоритма [3] в данной работе представлен численный метод, основанный на теории динамического программирования.

Актуальность данной задачи определяется, во-первых, значительным вниманием, уделяемым автоматизации современного машиностроительного производства, направленной на повышение конкурентоспособности выпускаемых изделий за счет сокращения их станкоемкости, а следовательно и себестоимости и, во-вторых, повышением эффективности САПР ТП применительно к такому производству.

1. Постановка задачи

Рассматриваемая задача распространяется, как отмечено выше, на многопозиционную технологическую обработку, предполагающую, что машинные времена на этих позициях $\{\tau_1, \tau_2, ..., \tau_r\}$ (τ_j – время обработки на j-й позиции, определяемое суммой времен выполняемых технологических переходов) были максимально равны друг другу. Это обеспечивает максимальную технологическую производительность, выступающую в данном случае в качестве критерия структурной оптимизации технологической операции.

Таким образом задача сводится к распределению основных технологических переходов $\mathbf{P} = \{p_i\}$, $i \in [1, n]$ между r позициями технологического комплекса. При этом имеет место отображение $\xi : \mathbf{P} \to \mathbf{T}$, где $\mathbf{T} = \{t_i\}$ – множество времен выполнения этих переходов.

С учетом сказанного условие оптимизации структуры технологической операции по критерию максимальной производительности можно записать следующим образом:

$$\tau_{sup} - \tau_{inf} = \varepsilon \to 0, \tag{1}$$

где $\tau_{sup} = \max\{\tau_j\}$, $\tau_{inf} = \min\{\tau_j\}$, $j = [1, r]$.

Это вытекает из того, что несмотря на различие времени выполнения обработки на отдельных позициях синхронизация времени пребывания заготовок на этих позициях предполагает одно значение – τ.

2. Метод оптимизации

Идея метода основана на заполнении временно**о**го интервала τ элементами множества $\mathbf{T} = \{t_i\}$ с выполнением условия

$$\tau - \sum_{t_i \in \mathbf{T}} t_i = \delta \to 0 . \tag{2}$$

Теоретическую основу метода составляет теория динамического программирования.

Для достижения поставленной цели использован метод динамического программирования при решении так называемой "задачи об оптимальной загрузке рюкзака" [4], которая подверглась модификации в связи с особенностями основной задачи.

При этом предварительно из элементов множества $\mathbf{T}$ формируется новое упорядоченное множество $\overline{\mathbf{T}} = \{t_i\}$ таким образом, что для $\forall\, i \in [1, n-1] \Rightarrow t_i \le t_{i+1}$.

Результат решения в постановке (2) определяется вектором $\mathbf{X} = \{x_i\}$, $i \in [1, n]$, компоненты которого соответствуют элементам множества $\overline{\mathbf{T}}$ и могут принимать значение "1" в случае, если соответствующий элемент из $\overline{\mathbf{T}}$ включен в интервал τ, либо значение "0" в противном случае.

Целевая функция определена следующим образом:

$$F(\mathbf{X}) = \sum_{i=1}^{n} c_i \cdot x_i\,, \tag{3}$$

где c_i – весовое значение соответствующего интервала из $\overline{\mathbf{T}}$. Механизм назначения оценок c_i заключается в следующем:

$$c_i = \begin{cases} c_i = 1 \text{ для } i = 1; \\ c_i = c_{i-1} + 1 \text{ для } \forall\, i \in [1, n-2] \text{ если } t_i > t_{i-1}; \\ c_i = c_{i-1} \text{ для } \forall\, i \in [1, n-2] \text{ если } t_i = t_{i-1}. \end{cases}$$

Используемое линейное ограничение имеет вид

$$\sum_{i=1}^{n} t_i \cdot x_i \le \tau\,, \tag{4}$$

который может быть приведен к целочисленному (это всегда можно сделать выбором более мелких единиц измерения интервалов t_i)

$$\sum_{i=1}^{n} t_i \cdot x_i \le t^* + \Delta t \cdot k^*, \tag{5}$$

где t^* – наименьший элемент из $\overline{\mathbf{T}}$, Δt – единичный интервал, значение k^* определяется из эквивалентности правых частей (4) и (5).

Если определить функцию φ_k как определяемое на k-м шаге максимальное значение целевой функции (3) на множестве всех допустимых значений вектора $\mathbf{X}$ при заданном ограничении (5), то

основное рекуррентное соотношение Беллмана для рассматриваемой задачи запишется в виде

$$\varphi_k = \max_{i \in \mathbf{I}_u} (c_i + \varphi_u), \quad u = k - t_i / \Delta t, \quad \mathbf{I}_u : u \geq -t^* / \Delta t. \tag{6}$$

Алгоритм решения задачи (2) состоит из двух частей, соответствующих прямому рекуррентному и обратному ходу, определяющему вектор **X.**

При этом для $\forall k$ используется вектор $\boldsymbol{\beta}_k = \{\beta_k^0, \beta_k^1, ...\}$, компонентам которого присваиваются значения оптимальных политик при проверке рекуррентного соотношения (6), а также скаляр α_k, равный количеству запомненных в $\boldsymbol{\beta}_k$ альтернативных политик.

Алгоритм прямого хода.

Шаг 1. Положить $k = 0$.

Шаг 2. Снять все метки с элементов $\overline{\mathbf{T}} = \{t_i\}$. Положить $\varphi_k = 0$.

Шаг 3. Найти i-й непомеченный элемент из $\{t_i\}$. Если поиск удачен, то пометить его и перейти к *шагу 4*, если нет – *к шагу 10.*

Шаг 4. Вычислить u по формуле из (6) и проверить справедливость неравенства из (6). Если неравенство справедливо, то перейти к *шагу 5*, если нет – *к шагу 3.*

Шаг 5. Если $u < 0 \Rightarrow \varphi_u = 0$. Перейти *к шагу 6.*

Шаг 6. Проверить справедливость неравенства $c_i + \varphi_u > \varphi_k$. Если неравенство справедливо, то перейти к *шагу 7*, если нет – к *шагу 9.*

Шаг 7. Проверить равенство $\beta_u^0 = i$. Если равенство справедливо, то перейти *к шагу 3*. (Выполнение данного шага исключает движение во время обратного хода по пути повторного выбора элементов из $\{t_i\}$ и ухода от оптимального решения, что проиллюстрировано в примере 1.)

Шаг 8. Положить $\varphi_k = c_i + \varphi_u$, $\beta_k^0 = i$, $\alpha_k = 0$ и перейти к *шагу 3.*

Шаг 9. Проверить справедливость равенства $c_i + \varphi_u = \varphi_k$. Если равенство справедливо, то положить $\alpha_k = \alpha_k + 1$, $\beta_k^{\alpha_k} = i$. Перейти к *шагу 3.*

Шаг 10. Положить $k = k + 1$. Если $k > k^*$, то закончить, если нет – перейти к *шагу 2*.

Алгоритм обратного хода.

Шаг 1. Положить $k = k^*$ и перейти *к шагу 2.*

Шаг 2. Положить $v = 0$ и перейти *к шагу 3.*

Шаг 3. Положить $i = \beta_k^v$ и проверить: если $x_i = 0$, то положить $x_i = 1$ и перейти *к шагу 5,* если нет – перейти *к шагу 4.*

Шаг 4. Положить $v = v + 1$ и проверить: если $v > \alpha_k$, то положить $k = k - 1$ и перейти к *шагу 2*, если нет – перейти к *шагу 3.*

Шаг 5. Вычислить $u = k - t_i / \Delta t$ и проверить, если $u < 0$, то закончить, если нет, то положить $k = u$ и перейти к *шагу 2.*

Работа метода заполнения интервала τ иллюстрируется примером 1 с целью демонстрации отличительных его особенностей.

Пример 1. Необходимо элементами множества $\overline{\mathbf{T}} = \{ 3, 4, 5 \}$ максимально заполнить в соответствии с (2) интервалы $\tau_1 = 10$, $\tau_2 = 8$ и $\tau_3 = 7$.

Весовые значения элементов $\overline{\mathbf{T}}$ определятся как $c_1 = 1$, $c_2 = 2$, $c_3 = 3$, а остальные параметры – $t^* = 3$, $\Delta t = 1$. Результат работы алгоритма прямого хода показан в табл.1.

Таблица 1

Пояснение к работе алгоритма прямого хода

Параметры	**Значение параметров**							
k	0	1	2	3	4	5	6	7
φ_k	1	2	3	3	3	4	6	6
$\boldsymbol{\beta}_k$	{1}	{2}	{3}	{3}	{1, 2, 3}	{1, 3}	{2}	{3}

Обратный ход для τ_1 выглядит следующим образом:

– на первой итерации $k = k^* = 7$, $i = \beta_7^1 = 3$ (номер включаемого в интервал τ элемента из $\overline{\mathbf{T}}$);

– на второй итерации $k = u = 7 - (5 / 1) = 2$, $i = \beta_2^1 = 3$ – т.е. идет процесс повторного включения в τ 3-го элемента из $\overline{\mathbf{T}}$, что вполне естественно с точки зрения максимального заполнения τ (согласно (2) $\delta = 10 - (5 + 5) = 0$), но является неприемлемым для решения нашей задачи. Включение в алгоритм проверки $x_i = 0$ в шаге 3 предопределяет пересчет величины k на шаге 4: $k = 2 - 1 = 1$.

– на следующей итерации $k = 1,\ i = \beta_1^1 = 2.$

Таким образом, в интервал τ_1 включены интервалы времени $t_2 = 4$ и $t_3 = 5$.

Применительно к интервалу τ_2 необходимо заметить следующее. Процесс обратного хода начинается с $k = k^* = 5$. Причем, благодаря включению в алгоритм прямого хода шага 7, в компонентах вектора $\boldsymbol{\beta}_5$ отсутствует значение "2", предопределяющее повторный выбор второго элемента из $\overline{\mathbf{T}}$. Это было бы вполне оправдано для задачи "оптимальной загрузки рюкзака", так как $\delta = 8 - (4 + 4) = 0$. Однако в нашем случае это привело бы к тому, что алгоритм обратного хода сформирует вектор $\mathbf{X} = \{1, 1, 0\}$, который не является оптимальным. В реальности же будет получен вектор $\mathbf{X} = \{1, 0, 1\}$.

Применительно к интервалу τ_3 необходимо заметить, что к правильному результату приводит выбор в качестве i из $\boldsymbol{\beta}_4$ значения "1", а не "3". Это обеспечивается, во-первых, порядком формирования векторов $\boldsymbol{\beta}_k$ при реализации прямого хода и, во-вторых, порядком просмотра компонентов $\boldsymbol{\beta}_k$ на обратном ходе. Выбор значения "3" является тоже правильным с точки зрения задачи "оптимальной загрузки рюкзака", так как суммарный вес решения получается один и тот же. В нашем же случае такое решение не будет оптимальным, то есть соответствовать условию (2).

Ниже приводится алгоритм решения основной задачи структурной оптимизации.

Алгоритм структурной оптимизации.

Шаг 1. Вычислить интервал τ по формуле

$$\tau = \begin{cases} \left\lceil \frac{1}{r}\sum_{i=1}^{n} t_i \right\rceil, \text{ если } \left\{ \frac{1}{r}\sum_{i=1}^{n} t_i \right\} = 0 \\ \left\lceil \frac{1}{r}\sum_{i=1}^{n} t_i \right\rceil + 1, \text{ если } \left\{ \frac{1}{r}\sum_{i=1}^{n} t_i \right\} \neq 0 \end{cases} \qquad (7)$$

и перейти к *шагу 2.*

Шаг 2. Положить $w = 1$ (w – номер позиции технологического оборудования) и перейти к *шагу 3*.

Шаг 3. Сформировать множества $\overline{\mathbf{T}}$ и $\{c_i\}$, рассчитать параметры t^*, Δt, k^* и перейти к *шагу 4а.*

Шаг 4а. Заполнить с помощью алгоритмов прямого и обратного хода интервал τ и проверить на допустимость[1] технологические переходы, соответствующие временным интервалам из $\overline{\mathbf{T}}$, отмеченные вектором **X**. Если недопустимых переходов нет – перейти к *шагу 4б* , если есть – удалить из **T** временные интервалы, соответствующие этим переходам, поместив их во временное множество **Q** , и перейти к *шагу 3.*

Шаг 4б. Закрепить за *w*-й технологической позицией отмеченные вектором **X** временные интервалы из $\overline{\mathbf{T}}$, удалить эти интервалы из **T**, переместить обратно из временного множества **Q** все элементы в **T** и перейти к *шагу 5.*

Шаг 5. Проверить, является ли множество **T** пустым: если да – перейти к *шагу 6*, если нет – положить $w = w + 1$ и перейти к *шагу 3.*

Шаг 6. Проверить справедливость равенства $w = r$. Если равенство справедливо, то закончить, если нет – положить[2] $\tau = \tau + \Delta\tau$, $w = 1$, восстановить множество **T** и перейти к *шагу 3.*

Проверка на допустимость технологических переходов состоит из двух этапов.

На первом этапе проверяется допустимость реализации перехода $p_i \in \mathbf{P}$ на рассматриваемой позиции технологического оборудования. Для этого на начальном этапе до выполнения оптимизации формируется матрица $\mathbf{A} \equiv [a_{ij}]$ размерностью $n \times r$. Если технологический переход p_i может выполняться на *j*-й технологической позиции, то элемент a_{ij} матрицы **A** принимает значение 1, в противном случае – 0. Таким образом допустимость перехода p_i сводится к проверке равенства $a_{iw} = 1$.

На втором этапе проверяется возможное условие необходимости предшествования технологическому переходу $p_i \in \mathbf{P}$ другого перехода из рассматриваемого множества **P**. Для этого на начальном этапе до выполнения оптимизации формируется матрица $\mathbf{B} \equiv [b_{ij}]$ размерностью $n \times n$. Каждая *i*-я строка матрицы соответствует переходу p_i, элементы

[1] Механизм проверки объясняется ниже после описания алгоритма.

[2] Интервал $\Delta\tau$ выбирается как $\Delta\tau = \Delta t$.

которой принимают значения 0 или 1. Если данному переходу в обязательном порядке должно предшествовать выполнение перехода под номером *j*, то элемент b_{ij} данной строки матрицы **B** принимает значение 1, в противном случае – 0. В результате проверки технологический переход является не допустимым если $\exists\, j \in [1, n]$, для которого истинно выражение $(b_{ij} = 1) \wedge ((t_j \in \mathbf{T}) \wedge (x_j = 0))$.

Работа данного алгоритма проиллюстрирована примером 2.

Пример 2. Требуется распределить пять технологических переходов между двумя технологическими позициями ($r = 2$). Вектор $\overline{\mathbf{T}}$, определяемый временами выполнения данных переходов, равен $\overline{\mathbf{T}} = \{3, 3, 4, 5, 5\}$.

Интервал τ рассчитывается как $\tau = (3+3+4+5+5) / 2 = 10$.

Работа алгоритма прямого хода при заполнении первой технологической позиции сведена в табл. 2.

Таблица 2

Пояснение к работе алгоритма прямого хода для первой технологической позиции

Параметры	**Значение параметров**			
k	0	1	2	3
φ_k	1	2	3	3
$\boldsymbol{\beta}_k$	{1, 2}	{3}	{4, 5}	{4, 5}
k	4	5	6	7
φ_k	3	4	5	6
$\boldsymbol{\beta}_k$	{1, 2, 3, 4, 5}	{1, 2 , 4, 5}	{3, 4, 5}	{4, 5}

При обратном ходе формируется вектор $\mathbf{X} = \{0, 0, 0, 1, 1\}$.

При заполнении второй технологической позиции используется вектор $\overline{\mathbf{T}} = \{3, 3, 4\}$. Работа алгоритма прямого хода при этом сведена в табл. 3.

Таблица 3

Пояснение к работе алгоритма прямого хода для второй технологической позиции

Параметры	**Значение параметров**			
k	0	1	2	3
φ_k	1	2	2	2
$\boldsymbol{\beta}_k$	{1, 2}	{3}	{3}	{1, 2, 3}

Продолжение табл.3

Параметры	**Значение параметров**			
k	4	5	6	7
φ_k	3	4	4	4
β_k	{1, 2, 3}	{1, 2, 3}	{3}	{1, 2, 3}

При обратном ходе формируется вектор **X** = {1, 1, 1}.

Заключение

Описанный метод оптимизации реализует структурное распределение технологических переходов между позициями автоматического технологического оборудования, обеспечивая синхронизации основного времени и, тем самым, повышая технологическую производительность на 10 – 15%.

Кроме этого формализация процедуры формирования структуры технологической операции на таком оборудовании повышает уровень автоматизации процесса проектирования средствами САПР ТП.

В целом полученные результаты способствуют повышению конкурентоспособности наукоемких изделий в области машиностроения.

Литература

1. ГОСТ Р 50.1.027-2001. Информационные технологии поддержки жизненного цикла продукции. Автоматизированный обмен технической информацией. Основные положения и общие требования.– Введен 2002-07-01.– М: Стандартинформ, 2016. – 36с.
2. Кондусова, В.Б. Повышение эффективности функционирования САПР на основе разработки методологии информационной поддержки жизненного цикла наукоемких изделий / В.Б. Кондусова, А.И. Сердюк, Д.В. Кондусов, А.И. Сергеев // Информационные технологии в проектировании и производстве.– 2020.– №2.– С. 16-20.
3. Анфёров, М. А. Генетический алгоритм структуризации технологических операций / М. А. Анфёров // Матер. XIV междунар. научно-метод. конф. «Информатика: проблемы, методология, технологии», г. Воронеж, 6-7 февраля 2014г.– сб. тр. в 3 т., Т. 3.– Воронеж: ВГУ, 2014.– С.82–85.
4. Беллман Р., Дрейфус С. Прикладные задачи динамического программирования.– М.: Наука, 1965.– 380 с.