ОБРАБОТКА ИЗОБРАЖЕНИЙ

О СИНТЕЗЕ ЭФФЕКТИВНОГО АЛГОРИТМА НАД МНОЖЕСТВОМ АЛГОРИТМОВ ВЫЧИСЛЕНИЯ СВЕРТКИ

В.В. Мясников
Институт систем обработки изображений РАН,
Самарский государственный аэрокосмический университет

Аннотация

В работе рассматривается проблема синтеза эффективного алгоритма, предназначенного для решения задачи вычисления линейной свертки. Для построения искомого алгоритма вводится замыкание заранее заданного множества алгоритмов по модели (преобразования), которое представляет собой новое множество алгоритмов. Алгоритм с наилучшими вычислительными характеристиками из замыкания называется индуцированным алгоритмом. Индуцированный алгоритм, по построению, использует для решения задачи вычисления свертки не только наиболее подходящее подмножество алгоритмов исходного множества, но и характеристики обрабатываемого сигнала с импульсной характеристикой. В работе доказывается ряд теорем, которые устанавливают необходимые и достаточные условия эффективности и строгой эффективности индуцированного алгоритма. Аналогичные теоремы доказываются для практически важного случая, когда в качестве исходного множества выбираются алгоритмы основных классов: алгоритма прямого вычисления свертки; алгоритмов, построенных на основе дискретных ортогональных преобразований (типа БПФ); и рекурсивных алгоритмов вычисления свертки (рекурсивных фильтров). Приводится общее описание метода синтеза эффективного алгоритма, который разработан на основе полученных теоретических результатов. Представлено детальное алгоритмизированное описание процедур, которые реализуют отдельные этапы предлагаемого метода. Приводятся несколько известных алгоритмов вычисления свертки, которые являются частными решениями рассматриваемой проблемы синтеза эффективного алгоритма.

Введение

Вычисление свертки является базовой и наиболее часто используемой операцией в цифровой обработке сигналов и изображений. По этой причине существует огромное количество различных алгоритмов расчета свертки. Условно их можно разделить на три группы.

В первую группу входит единственный алгоритм, который выполняет вычисления непосредственно в соответствие с выражением свертки.

Во вторую, наиболее многочисленную, группу входят быстрые алгоритмы вычисления свертки (типа БПФ), основанные на дискретных ортогональных преобразованиях (ДОП). Существует большое число работ, посвященных вопросам построения и использования быстрых алгоритмов типа БПФ для вычисления свертки. Среди крупных работ, посвященных этому направлению в ЦОС, можно выделить следующие: С.С. Агаяна [1], Н. Ахмеда и К.Р. Рао [3], Р. Блейхута [5], В.Г. Лабунца [7, 18, 19], Г. Нуссбаумера [28], Т.С. Хуанга [6], В.М. Чернова [37], Л.П. Ярославского [9, 39-41]. Теория синтеза алгоритмов этой группы характеризуется серьезной экспансией алгебраических методов и структур, которая привела к развитию целого ряда направлений в построении алгоритмов. Однако среди этих направлений существуют и общие черты. Например, как отмечено в работе В.М. Чернова [37, стр. 10], «структура БА представляет, как правило, некоторую рекурсивную процедуру, последовательно реализующую редукцию вычисления ДОП заданного объема к ДОП меньшего объема или более простых преобразуемых массивов. Типичными схемами таких редукций являются: редукция Кули-Тьюки, редукция Гуда-Томаса, редукция Рейдера, методы «совмещенного» вычисления ДОП».

В третью группу входят алгоритмы рекурсивного вычисления свертки. Алгоритмы этой группы заменяют прямое выражение свертки на рекурсивное, в котором текущие отсчеты выходного сигнала выражаются через предыдущие. Подобный способ вычислений в ЦОС именуется термином «рекурсивный фильтр» [8, 12, 29, 30, 35, 38-41]. Поскольку аналитический вид рекурсивного фильтра хорошо известен, синтез алгоритмов этой группы сводится, фактически, к выбору его числовых параметров. Ряд работ по рекурсивным алгоритмам, включая некоторые работы автора, связан с трансформацией рекурсивной вычислительной конструкции в параллельно-рекурсивную, в которой несколько рекурсивных вычислений производятся в параллельном режиме [24-26, 32, 40, 43-47, 52-63].

К сожалению, изобилие алгоритмов вычисления сверток и методов их синтеза не решает основную практическую проблему: как для конкретной задачи свертки указать наилучший алгоритм ее решения. Здесь следует отметить работы В.Г. Лабунца [19] и В.М. Чернова [37], в которых авторы обозначили эту проблему и предложили конструктивные авторские подходы к синтезу быстрых алгоритмов. Однако, следование этим подходам, с одной стороны, требует высочайшей математической квалификации, что не

приемлемо для конечного пользователя, которому надо просто быстро решить задачу вычисления свертки. С другой стороны (что более важно), указанные подходы не могут гарантировать, что полученный с их помощью алгоритм будут обладать наилучшими вычислительными характеристиками среди всех известных алгоритмов вычисления свертки.

Дополнительным аспектом в проблеме синтеза быстрого алгоритма, который обычно не затрагивается в работах по быстрым алгоритмам и ДОП, является использование аналитических свойств обрабатываемых сигналов и учет специфики самой задачи вычисления свертки. Применительно к этому аспекту следует отметить совместные работы Л.П. Ярославского, И.А. Овсиевичаи, В.И. Кобера [17, 39, 53, 54, 62]. Указанные авторы рассматривали возможность снижения сложности обработки за счет устранения «информационной избыточности» входного сигнала. В основном, авторы ограничивались использованием операциями дифференцирования и/или простейшего предсказания входного сигнала. К сожалению, эти работы не получили своего логического развития. Например, задача синтеза быстрых алгоритмов ДОП, учитывающих «информационную избыточность», фактически осталась не рассмотренной. Автору настоящей работы известно лишь два простейших решения этой задачи. Одно из них рассматривалось в монографии Л.П. Ярославского [39]. Другое решение приводилось в работе П.В. Раудина [31], выполненной под руководством В.М. Чернова.

Существует еще один аспект обозначенной проблемы, который не затрагивается в работах по синтезу быстрых алгоритмов. Обилие различных алгоритмов вычисления свертки уже стало фактом. Поэтому естественным представляется стремление воспользоваться существующими алгоритмами для синтеза лучшего алгоритма вычисления свертки. Здесь следует отметить то, что похожая ситуация уже возникала в кибернетике. А именно, почти 30 лет назад применительно к алгоритмам распознавания она была обозначена академиком Ю.И. Журавлевым следующим образом [16]: «…Существование … алгоритмов уже давно стало фактом. По-видимому, появление каждого из таких алгоритмов можно рассматривать как эксперимент, а со множеством таких экспериментов и результатов – работать как с новым для математики множеством объектов» (1978 год). В распознавании образов такая точка зрения привела к созданию одного из ключевых направлений в алгебраической теории распознавания образов – алгебре над множествами некорректных (эвристических) алгоритмов [14-16].

Настоящая работа посвящена решению указанной проблемы синтеза эффективного алгоритма, предназначенного для решения конкретной задачи вычисления свертки. Все три обозначенных выше аспекта этой проблемы приняты во внимание. В частности, предлагаемое в настоящей работе решение:

- учитывает специфику и ограничения задачи вычисления свертки,
- использует аналитические свойства сигнала и фактический вид импульсной характеристики, участвующей в операции свертки,
- использует любое предоставляемое множество алгоритмов при синтезе искомого алгоритма,
- гарантирует, что синтезированный алгоритм будет эффективным над (обозначенным ранее) множеством алгоритмов.

Последнее свойство подразумевает то, что синтезированный алгоритм в вычислительном плане:

- для любой задачи вычисления свертки окажется не хуже чем наилучший алгоритм из предоставленного множества алгоритмов,
- для некоторых задач окажется лучше наилучшего алгоритма из этого множества.

Настоящая работа организована следующим образом.

В первом разделе дается постановка задачи вычисления свертки, и приводятся основные определения. В частности, формализуются понятия задачи и алгоритма вычисления свертки, приводятся ограничения задачи. Далее вводится понятие сложности алгоритма, понятия эффективного и строго эффективного алгоритма, производится разделение всех алгоритмов вычисления свертки на два множества: алгоритмы постоянной и вариантной сложности. В этом же разделе рассматриваются основные классы алгоритмов вычисления свертки.

Во втором разделе рассматривается множество алгоритмов постоянной сложности, анализируются свойства этих алгоритмов. В частности, вводится понятие, и указываются методы распространения алгоритма постоянной сложности. Формулируются ограничения, накладываемые на функцию сложности алгоритма постоянной сложности задачей вычисления свертки. Вводится понятие «приведенные алгоритмы постоянной сложности», которое характеризует множество алгоритмов, которые удовлетворяют этим ограничениям. Наконец, вводится «компетентный» алгоритм, который решает конкретную задачу вычисления свертки, используя наилучший алгоритм из предоставленного множества. В этом же разделе доказывается ряд лемм и теорем, связанных со свойствами обозначенных алгоритмов.

Третий раздел является центральным в работе. В нем вводится модель (алгоритма), которая позволяет учитывать и специфику обрабатываемого сигнала с КИХ, и предоставляемое множество алгоритмов. Все множество алгоритмов, которое может быть получено в рамках этой модели (ее аргументами выступают и числовые параметры, и алгоритмы из предоставляемого множества), определено как замыкание предоставляемого множества алгоритмов. Вводится понятие «индуцированного алгоритма», как наилучшего алгоритма из замыкания для конкретной задачи. Приводится ряд лемм и теорем, которые выделяют необходимые и достаточные условия (строгой) эффективности индуцированного алгоритма.

В четвертом разделе производится адаптация общей теории, изложенной в третьем разделе, к основным (трем) классам алгоритмов вычисления свертки. Выделяется достаточное множество алгоритмов, для которых строится индуцированный алгоритм, при котором гарантируется его эффективность над всеми алгоритмами из основных (трех) классов.

Пятый раздел содержит общее описание метода синтеза эффективного алгоритма над множеством алгоритмов вычисления сверток.

Шестой раздел содержат алгоритмическую детализацию первого этапа синтеза эффективного алгоритма.

Седьмой раздел содержит теоретические основы и алгоритмическую детализацию для второго этапа синтеза эффективного алгоритма.

В восьмом разделе приводятся несколько известных алгоритмов вычисления свертки, которые оказываются частными решениями рассматриваемой общей проблемы синтеза эффективного алгоритма.

В заключение работы приводятся благодарности фондам, поддерживающим данную научную работу, а также список литературы.

1. Постановка задачи и основные определения

1.1. Постановка задачи Z вычисления свертки. Априорная информация о задаче Z

Рассмотрим задачу вычисления одномерной (линейной) свертки конечного *входного сигнала* $\{x(n)\}_{n=0}^{N-1}$ длины N и *конечной импульсной характеристики* (*КИХ*) $\{h(m)\}_{m=0}^{M-1}$ длины M:

$$y(n) = h(n) * x(n) = \sum_{m=0}^{M-1} h(m)x(n-m), \quad n = \overline{M-1, N-1}, \tag{1}$$

Результатом ее решения является получение сигнала $\{y(n)\}_{n=0}^{N-M+1}$ длины $N-M+1$, называемого *выходным сигналом*.

Эта задаче в подавляющем большинстве практических приложений имеет ряд ограничений следующего характера.

Ограничения задачи Z

(a) Длина ИХ существенно меньше длины обрабатываемого входного сигнала $M << N$. Для определенности в дальнейшем будем считать, что различие в длинах ИХ и сигнала не может быть менее порядка: $10 \cdot M < N$.

(b) Ограничение на КИХ:

$$h(0) \neq 0, h(M-1) \neq 0. \tag{2}$$

(c) Отсчеты КИХ $\{h(m)\}_{m=0}^{M-1}$ известны задолго до момента решения задачи.

(d) Отсчеты входного сигнала оказываются известными только на момент решения задачи, однако до момента решения задачи о входном сигнале может быть известна некоторая априорная информация, обозначаемая в дальнейшем $\Im_x$ и называемая *априорной информацией о входном сигнале*. Примерами такой априорной информации могут служить данные о длине входного сигнала N, информация о его непрерывности, существовании производных, о его статистических свойствах, о функциональной связи между отсчетами и т.п.

(e) Значения ИХ и входного сигнала вещественны: $\forall n \quad x(n) \in R, h(n) \in R$.

Как следует из указанных ограничений, данные о КИХ, а также некоторые (возможно пустые) данные $\Im_x$ об обрабатываемом сигнале известны задолго до наступления момента решения задачи. В связи с этим введем понятие априорной информации.

Определение 1. *Априорной информацией* задачи Z вычисления свертки (1) называется пара $\Im_0 = \left(\{h(n)\}_{n=0}^{M-1}, \Im_x\right)$.

Обозначим в дальнейшем задачу вычисления свертки (1) с учетом ограничений (a)-(d) как $Z\left(\Im_0, \{x(n)\}_{n=0}^{N-1}\right)$. В изложении аргументы задачи в обозначении могут быть опущены, если это не влияет на ясность изложения.

1.2. Представление априорной информации о входном сигнале $\Im_x$ в задаче Z

Формальный способ описания свойств сигнала фиксируется следующими определениями.

Определение 2. А*приорной информацией о свойствах сигнала* $\{x(n)\}$ называется множество, обозначаемое $\Im_{(x)} = \{\vartheta^x\}$, каждый элемент которого, обозначаемый ϑ^x, называются э*лементарной информацией о свойствах сигнала* и представляет собой числовой вектор вида

$$\vartheta^x = \left(K, \{a_k^x\}_{k=1}^K, p_x, D_x\right),$$

где:

K – натуральное число,

$\{a_k^x\}_{k=1}^K$ - некоторые вещественные величины,

$$p_x = P\left(x(n) = \sum_{k=1}^{K-1} x(n-k)\right),$$

$$D_x = E\left(\varepsilon(n) = x(n) - \sum_{k=1}^{K-1} x(n-k)\right) \text{ (среднее).}$$

Определение 3. *Априорной информацией о сигнале* $\{x(n)\}_{n=0}^{N-1}$ называется пара $\Im_x = \left(N, \Im_{(x)}\right)$, где $N \in \mathbb{Z}_+ \cup \Delta$ - число отсчетов сигнала x (Δ - символ неопределенности значения), $\Im_{(x)}$- априорная информация о свойствах сигнала.

1.3. Алгоритм решения задачи Z. Сложность алгоритма. Понятие эффективного и строго эффективного алгоритма

Будем использовать следующее определение (численного) алгоритма [20, 21].

Определение 4. *Алгоритм* – предписание, однозначно определяющее ход некоторых конструктивных процессов.

Поскольку основной интерес в настоящей работе представляют алгоритмы решения вполне конкретных численных задач вычисления свертки примем следующее

Определение 5. *Алгоритм решения задачи (вычисления свертки)* $Z = Z\left(\Im_0, \{x(n)\}_{n=0}^{N-1}\right)$ с априорной информацией $\Im_0 = \left(\{h(n)\}_{n=0}^{M-1}, \Im_x\right)$ – алгоритм, обозначаемый $A(Z)$, которой по входному сигналу $\{x(n)\}_{n=0}^{N-1}$ и КИХ $\{h(m)\}_{m=0}^{M-1}$ производит вычисление отсчетов сигнала $\{y(n)\}_{n=0}^{N-M+1}$, связанных с отсчетами входного сигнала и КИХ соотношением (1).

Везде в дальнейшем по умолчанию под (просто) алгоритмами подразумеваются именно алгоритмы решения различных задач вычисления свертки. При использовании понятия «алгоритм» в другом контексте, смысл этого понятия и/или его конкретизация будет производиться отдельно и явно.

Поскольку различные алгоритмы могут решать различные задачи вычисления сверток, поэтому вводится следующее определение, являющееся аналогом соответствующего определения в теории алгоритмов [20].

Определение 6. Алгоритм *A применим к задаче Z*, если он является алгоритмом решения задачи *Z*. Факт применимости обозначаем $!A(Z)$. Алгоритм, не являющийся применимым к задаче *Z*, называется *неприменимым к задаче Z.*

Факт получения результата при решении задачи в дальнейшем обозначается:

$$A : Z\left(\Im_0, \{x(n)\}_{n=0}^{N-1}\right)\Big|_{\Im_0=\left(\{h(n)\}_{n=0}^{M-1}, \Im_x\right)} \Rightarrow \{y(n)\}_{n=M-1}^{N-1},$$

при фиксированной задаче Z запись может быть сокращена: $A : Z \Rightarrow \{y(n)\}$.

Прежде чем перейти к следующим определениям, следует отметить один важный факт, связанный с формализацией понятия алгоритма. На данный факт, в частности, указано в книге А.А. Маркова и Н.М. Нагорного [20, стр.135-136]: «Слово «однозначно» в определении алгорифма не должно пониматься слишком буквально. Для состояния системы может быть установлено то или иное отношение одинаковости… Процесс, определяемый алгорифмом при задании начального состояния системы, должен определяться однозначно лишь «с точностью до одинаковости» получаемых в его ходе состояний. Потребовать, чтобы этот процесс был единственным, мы можем, лишь условившись единственным образом применять абстракцию отождествления к состояниям нашей системы и процессам, протекающим в ней». Поскольку для работы с алгоритмами вычисления сверток нам потребуется введение отношений, характеризующих их «похожесть», в работе принят изложенный ниже взгляд на понятие (численного) алгоритма. Этот взгляд в значительной степени отражает установившийся взгляд на процесс проектирования и описания программных изделий и систем, изложенный, например, в книге [4].

Условимся считать, что представление алгоритма может быть произведено на различных уровнях детализации. На самом верхнем, грубом, уровне детализации предписание содержит ссылку на метод и/или вычислительную конструкцию, которую реализует алгоритм. Например, ссылка на метод Ньютона является достаточно точным для описания некоторого алгоритма поиска экстремума функции.

Следующим уровнем представления является формула(ы), которая задает вычислительный процесс данного алгоритма. Например, формулу (1) можно считать достаточно точным предписанием для прямого алгоритма вычисления свертки.

Оба указанных выше метода составляют аналитическую часть представления численного алгоритма вычисления свертки. Связано это с тем, что ни метод, ни описание с использованием формул не является адаптированным к конкретной программной и/или аппаратной реализации алгоритма.

Последующие три уровня можно отнести к реализационной части представления численного алгоритма вычисления свертки. Эти части представления в различной степени учитывают особенности реализации алгоритма на конкретном средстве разработки и языке программирования, аппаратной платформе с учетом ее технических характеристик.

К уровням реализационной части представления относятся:

- блок-схема работы программы/устройства,
- описание алгоритма на конкретном языке программирования и/или псевдокоде,
- бинарный исполняемый код.

Следует также отметить, что научные работы, посвященные вопросам разработки быстрых алгоритмов вычисления свертки, как правило, ограничиваются описанием только аналитической части представления алгоритма, Это означает, в частности, что реализации алгоритма двумя различными людьми могут приводить к различным представлениям реализационной части. В ряде случаев, например, при реализации одного алгоритма на различных языках программирования и/или различных операционных системах, бинарный код представления алгоритма будет абсолютно разным.

С учетом изложенного взгляда на представление алгоритмов вычисления свертки, ниже принято следующее решение вопроса задания отношений, характеризующих «похожесть» алгоритмов.

Определение 7. *Алгоритмов тождественность* – это бинарное отношение, связывающее алгоритмы фиксированного типа и выражающее тот факт, что представления алгоритмов, задающие (вычислительный) процесс, абсолютно совпадают (совпадают и аналитическая и реализационная части представления). Алгоритмы, не являющиеся тождественными, называются *различными.*

Факт того, что два алгоритма $A1$ и $A2$ тождественны, будем обозначать $A1 = A2$. Факт того, что два алгоритма $A1$ и $A2$ различны, будем обозначать $A1 \neq A2$. Различные алгоритмы, в принципе, могут решать одну и ту же задачу, то есть выдавать один и тот же результат. Для отражения этого факта введем еще одно определение [20, 21].

Определение 8. *Алгоритмов эквивалентность* – бинарное отношение, связывающее алгоритмы фиксированного типа и выражающее тот факт, что у всяких двух связанных этим отношениям алгоритмов при совпадении определенного вида исходных данных совпадают результаты работы.

В рамках настоящего раздела рассматриваются только эквивалентные алгоритмы. Факт того, что два алгоритма $A1$ и $A2$ эквивалентны, будем обозначать $A1 \approx A2$.

Определение 9. *Алгоритмов подобность* – бинарное отношение, связывающее алгоритмы фиксированного типа и выражающее тот факт, что аналитические части представления алгоритмов, задающие (вычислительный) процесс, совпадают с точностью до обозначений и эквивалентных преобразований

Как видно из определений, понятие «тождественные» алгоритмы $(A_1 = A_2)$ предполагают абсолютное совпадение как аналитической (формулы и вычислительной конструкции - метода), так и реализационной (блок-схемы, псевдокода и бинарного кода реализации) частей представления двух алгоритмов.

Понятие «подобные алгоритмы» $(A_1 \cong A_2)$ предполагают менее строгое «совпадение» двух алгоритмов. Для «подобных» алгоритмов достаточно совпадения лишь аналитических частей в представлении алгоритмов, то есть формул, вычислительных конструкций и методов. При этом совпадение реализационной части представления (блок-схемы, псевдокода, кода реализации) не требуется.

Наконец, словосочетание «эквивалентные алгоритмы» $(A_1 \approx A_2)$ определяет наименее строгое понятие «похожести» алгоритмов. Для «эквивалентности» двух алгоритмов достаточно того, чтобы алгоритмы одинаково реагировали на входные данные. Ни аналитические, ни реализационные представления этих алгоритмов не влияют на отношение «эквивалентности».

На основании сделанных определений и комментариев к ним получаем

Следствие 1.

$A_1 = A_2 \Rightarrow A_1 \cong A_2$, $A_1 \cong A_2 \Rightarrow A_1 \approx A_2$.

Введем теперь в рассмотрение конечное множество (эквивалентных, но различных) алгоритмов $\{A\}$ решения задачи Z, $|\{A\}| < \infty$. Поскольку различные алгоритмы могут быть определены на различных подмножествах значений параметров задач, примем следующее определение.

Определение 10. *Полной областью определения,* обозначаемой $\aleph$, называется множество $\left\{Z\left(\Im_0, \{x(n)\}_{n=0}^{N-1}\right)\right\}$ всех возможных задач вычисления сверток.

Определение 11. Область определения $\aleph_A$ алгоритма A – множество задач, для которых алгоритм A применим:

$$\aleph_A = \{Z : Z \in \aleph \wedge !A(Z)\}.$$

Определение 12. Пусть фиксирована некоторая задача Z. *Множеством алгоритмов решения задачи Z*, задаваемым по отношению к множеству $\{A\}$ различных алгоритмов вычисления сверток, называется множество вида:

$$\{A(Z)\} = \{A: \quad A \in \{A\} \wedge !A(Z)\}.$$

Введем также следующее понятие, которое является аналогом соответствующего понятия в теории алгоритмов [20].

Определение 13. Пусть задан алгоритм A с областью определения $\aleph_A$. Алгоритм $\vec{\vec{A}}$ с областью определения $\aleph_{\vec{\vec{A}}}$ называется *распространением алгоритма A*, если

$$\aleph_A \subseteq \aleph_{\vec{\vec{A}}} \quad \wedge \quad \forall Z \in \aleph_A \ \vec{\vec{A}}(Z) \cong A(Z).$$

Все алгоритмы из множества $\{A(Z)\}$ алгоритмов решения задачи Z могут быть использованы для решения задачи Z и это решение - выходной сигнал - будет одинаковым для всех алгоритмов. Однако сложность/время решения задачи для разных алгоритмов оказывается различной и зависящей от *сложности алгоритма* [21]*:*

Определение 14 *Алгоритма сложность* вычислений – функция, дающая числовую оценку трудности (громоздкости) процессов применения алгоритма к исходным данным.

Известно, что сложность конкретного алгоритма может быть охарактеризована целым рядом показа-

телей [21], которые могут зависеть или не зависеть от конкретной реализации этого алгоритма на конкретной ПЭВМ (особенностей программной/аппаратной реализации, используемой элементной базы и архитектуры ПЭВМ). В частности, к зависящим от реализации показателям можно отнести: объем требуемой оперативной памяти, число обращений к оперативной памяти, число тактов процессора, временя работы алгоритма на ПЭВМ и т.д. К показателям, значения которых в значительной степени не зависят от конкретной реализации алгоритма решения задачи *Z*, можно отнести число требуемых для расчета выходного сигнала арифметических операций. Поэтому в дальнейшем в качестве составляющих сложности алгоритма решения задачи *Z* используются количества арифметических операций сложения и умножения, требуемых для получения одного отсчета выходного сигнала (удельное количество операций). Определим далее:

- $U_{add}(A(Z))$ - удельное количество сложений, требуемых в алгоритме *A* для решения задачи Z,
- $U_{mul}(A(Z))$ - удельное количество умножений, требуемых в алгоритме *A* для решения задачи Z.

Для фиксированной задачи *Z* аргумент для алгоритма в приведенных обозначениях будем опускать и писать следующим образом: $U_{add}(A)$ и $U_{mul}(A)$. Эту пару величин и их линейные комбинации в дальнейшем будем именовать *вычислительной сложностью алгоритма* или просто *сложностью*.

Поскольку для различных ПЭВМ реальное время выполнения операций сложения и умножения различаются, введем величину (удельной) сложности алгоритма *A* следующим образом:

$$U(A)=\xi_{add}U_{add}(A)+\xi_{mul}U_{mul}(A), \qquad (3)$$

где ξ_{add}, ξ_{mul} - вещественные неотрицательные величины, которые характеризуют относительное соотношение между сложностью выполнения операций, соответственно, вещественного сложения и вещественного умножения на конкретной ПЭВМ. В дальнейшем величины ξ_{add}, ξ_{mul} считаем заданными и фиксированными. Для удобства считаем, что они удовлетворяют уравнению: $\xi_{add}+\xi_{mul}=1$.

Справедливо следующее

Предложение 1. Вычислительные сложности подобных и тождественных алгоритмов равны:

$$A_1 = A_2 \Rightarrow U(A_1)=U(A_2),$$
$$A_1 \cong A_2 \Rightarrow U(A_1)=U(A_2).$$

Используя понятие сложности алгоритма, введем на множестве алгоритмов, применимых к задаче *Z*, отношение порядка.

Определение 15. Для задачи *Z* алгоритм $A'(Z)$

- *лучше* алгоритма $A''(Z)$, если $U(A'(Z))<U(A''(Z))$;
- *не хуже* алгоритма $A''(Z)$, если $U(A'(Z))\le U(A''(Z))$;
- *хуже* алгоритма $A''(Z)$, если $U(A'(Z))>U(A''(Z))$;
- *не лучше* алгоритма $A''(Z)$, если $U(A'(Z))\ge U(A''(Z))$.

Аналогичные отношения порядка можно ввести и просто для алгоритмов. В этом случае соответствующие неравенства должны выполняться для всех задач (вычисления сверток) из соответствующей области определения алгоритмов.

Определение 16. Алгоритм A' с областью определения $\aleph_{A'}$

- *лучше* алгоритма $A''(Z)$ с областью определения $\aleph_{A''}$ (обозначаем $U(A')<U(A'')$), если $\forall Z\in\aleph_{A''}\subseteq\aleph_{A'}\quad U(A'(Z))<U(A''(Z))$;
- *не хуже* алгоритма $A''(Z)$ с областью определения $\aleph_{A''}$ (обозначаем $U(A')\le U(A'')$), если $\forall Z\in\aleph_{A''}\subseteq\aleph_{A'}\quad U(A'(Z))\le U(A''(Z))$;
- *хуже* алгоритма $A''(Z)$ с областью определения $\aleph_{A''}$ (обозначаем $U(A')>U(A'')$), если $\forall Z\in\aleph_{A'}\subseteq\aleph_{A''}\quad U(A'(Z))>U(A''(Z))$;
- *не лучше* алгоритма $A''(Z)$ с областью определения $\aleph_{A''}$ (обозначаем $U(A')\ge U(A'')$), если $\forall Z\in\aleph_{A'}\subseteq\aleph_{A''}\quad U(A'(Z))\ge U(A''(Z))$.

В приводимых ниже определениях 17-19 и далее предполагается, что решается задача Z. Для ее решения дано некоторое множество алгоритмов $\{A\}$, причем оно может состоять не только из алгоритмов, решающих задачу *Z*. Таким образом, наряду с множеством $\{A\}$ существует его подмножество $\{A(Z)\}\subseteq\{A\}$, задаваемое определением 12. Также предполагается, что $\{A(Z)\}\neq\varnothing$. То есть если в множестве $\{A\}$ есть хотя бы один алгоритм, применимый к задаче Z.

Определение 17. *Наилучшим алгоритмом* из множества алгоритмов $\{A\}$ *для задачи Z* называется такой алгоритм $A^*(Z)\in\{A(Z)\}\subseteq\{A\}$, который не хуже любого другого алгоритма этого множества для этой задаче, то есть:

$$\forall\ A(Z)\in\{A(Z)\}\subseteq\{A\}\quad U(A^*(Z))\le U(A(Z)).$$

Для возможности сравнения сложности конкретного алгоритма и всех алгоритмов некоторого заданного множества введем еще два определения.

Определение 18. *Эффективным алгоритмом* над множеством алгоритмов $\{A\}$ *для задачи Z* называется такой алгоритм $A_{eff}(Z)\notin\{A\}$, который не хуже наилучшего алгоритма

$A^*(Z) \in \{A(Z)\} \subseteq \{A\}$ этого множества для этой задачи, то есть: $U\left(A_{eff}(Z)\right) \le U\left(A^*(Z)\right)$.

Определение 19. *Строго эффективным алгоритмом* над множеством алгоритмов $\{A\}$ для задачи Z называется такой алгоритм $A_{sef}(Z) \notin \{A\}$, который лучше наилучшего алгоритма $A^*(Z) \in \{A(Z)\} \subseteq \{A\}$ этого множества для этой задачи, то есть: $U\left(A_{sef}(Z)\right) < U\left(A^*(Z)\right)$.

Очевидно, что строго эффективный алгоритм является и эффективным.

Введение эффективного алгоритма (по отношению к строго эффективному) обусловлено одним важным фактом. А именно, для некоторых задач $Z \in \aleph$ и возможных множеств $\{A(Z)\}$ алгоритмов их решения существуют пары $(Z, \{A(Z)\})$, для которых построить алгоритм лучше наилучшего алгоритма этого множества оказывается не возможным.

Задаче построения эффективного и/или строго эффективного алгоритма над множествами алгоритмов вычисления свертки и посвящена настоящая работа.

1.4. Категории сложности алгоритмов вычисления свертки

Сложность (3) конкретного алгоритма, применимого к задаче *Z*, задаваемая величиной

$$U(A(Z)) = U\left(A\left(\Im_0, \{x(n)\}_{n=0}^{N-1}\right)\right),$$

в общем случае является сложной функцией собственно алгоритма, выбранного для решения задачи Z, и данных о задаче $Z = Z\left(\Im_0, \{x(n)\}_{n=0}^{N-1}\right)$. Степень использования заранее известных данных о задаче, то есть априорной информации $\Im_0 = \left(\{h(n)\}_{n=0}^{M-1}, (N, \Im_{(x)})\right)$, в каждом конкретном алгоритме может быть различной, поэтому и влияние их на значение сложности $U(A(Z))$ алгоритма оказывается разным. Для удобства разобьем все множество алгоритмов на категории, которые соответствуют различным аналитическим выражениям зависимостей величины сложности алгоритмов от частных параметров задачи – длины обрабатываемого сигнала *N* и размера ИХ *M*. В частности примем следующие определения.

Определение 20. Алгоритм $A(Z) \in \{A(Z)\}$ решения задачи $Z = Z\left(\Im_0, \{x(n)\}_{n=0}^{N-1}\right)$ с априорной информацией $\Im_0 = \left(\{h(n)\}_{n=0}^{M-1}, (N, \Im_{(x)})\right)$ называется *алгоритмом постоянной сложности*, если выражение (3) для вычислительной сложности этого алгоритма на всей области определения $\aleph_A$ алгоритма имеет вид аналитической функции $u(M, N)$, зависящей только от размеров входного сигнала *N* и длины ИХ *M*.

Определение 21. Алгоритм $A(Z) \in \{A(Z)\}$ решения задачи $Z = Z\left(\Im_0, \{x(n)\}_{n=0}^{N-1}\right)$ с априорной информацией $\Im_0 = \left(\{h(n)\}_{n=0}^{M-1}, (N, \Im_{(x)})\right)$ называется *алгоритмом вариантной сложности*, если он не является алгоритмом постоянной сложности.

Два приведенных определения позволяют разделить множество алгоритмов на два подмножества. В первом подмножестве сложность решения задачи Z зависит только от размеров сигнала *N* и импульсной характеристики *M* и не зависит от какой-либо другой части информации $\Im_0 = \left(\{h(n)\}_{n=0}^{M-1}, (N, \Im_{(x)})\right)$. Это означает, что сложность решения не изменится при изменении информации о свойствах входного сигнала $\Im_{(x)}$ и/или значений ИХ $\{h(n)\}_{n=0}^{M-1}$, то есть останется постоянной. Алгоритмы из подмножества алгоритмов вариантной сложности ведут себя по иному. А именно, при замене значений ИХ $\{h(n)\}_{n=0}^{M-1}$ и/или свойств входного сигнала $\Im_{(x)}$ сложность этих алгоритмов может измениться.

Далее, поскольку изменения отсчетов ИХ $\{h(n)\}_{n=0}^{M-1}$ и свойств обрабатываемого сигнала $\Im_{(x)}$ не влияют на сложность решения соответствующих этим данным задач, это означает также и то, что такие изменения не влияют и на саму способность алгоритма решать соответствующие (измененные) задачи. Получается, что если в область определения алгоритма постоянной сложности попадает задача *Z* с априорной информацией вида:

$$\Im_0 = \left(\{h(n)\}_{n=0}^{M-1}, (N, \Im_{(x)})\right),$$

то в эту же область определения входит также бесконечное множество других задач с теми же параметрами *M* и *N*, но с другими значениями ИХ $\{h(n)\}_{n=0}^{M-1}$, свойствами входного сигнала $\Im_{(x)}$ (удовлетворяющими ограничениям (а)-(е) задачи *Z*) и отсчетами входного сигнала $\{x(n)\}_{n=0}^{N-1}$. Более того, для всех этих задач сложность алгоритма постоянной сложности будет идентичной. Данный факт позволяет ввести классы эквивалентности на области определения алгоритмов постоянной сложности.

Определение 22. Задачи $Z' = Z\left(\Im'_0, \{x'(n)\}_{n=0}^{N'-1}\right)$ и $Z'' = Z\left(\Im''_0, \{x''(n)\}_{n=0}^{N''-1}\right)$ с априорными информациями, соответственно $\Im'_0 = \left(\{h'(n)\}_{n=0}^{M'-1}, (N', \Im'_{(x)})\right)$ и $\Im''_0 = \left(\{h''(n)\}_{n=0}^{M''-1}, (N'', \Im''_{(x)})\right)$, являются *эквивалентными относительно алгоритмов постоянной сложности* (далее – просто *эквивалентными*), если $M' = M''$, $N' = N''$.

Определение 23. *Классом эквивалентности задач* $\aleph(\widetilde{M}, \widetilde{N}) \subseteq \aleph$ является подмножество попар-

но эквивалентных задач из полной области определения с фиксированными значениями параметров $(M,N)=(\widetilde{M},\widetilde{N})$ и произвольными значениями других параметров:

$$\aleph(M,N)=\left\{\begin{array}{l} Z: Z\in\aleph\wedge Z=Z\left(\Im_0,\{x(n)\}_{n=0}^{N-1}\right)\\ \quad\wedge\Im_0=\left(\{h(n)\}_{n=0}^{M-1},\left(N,\Im_{(x)}\right)\right)\\ \quad N=\widetilde{N}\wedge M=\widetilde{M}\end{array}\right\}.$$

Справедливо следующее

Предложение 2. Полная область определения $\aleph$ представима в виде объединения классов эквивалентности задач:

$$\aleph=\bigcup_{(M,N)\in\{(M,N):\ M\in\mathbb{Z}_+,\ N\in\mathbb{Z}_+\}}\aleph(M,N), \qquad (4)$$

Определение 24. *Множеством индексов, порождающим область определения* $\aleph_A$ конкретного алгоритма A называется множество:

$$\aleph_{A(M,N)}=\{(M,N):\ \aleph(M,N)\subseteq\aleph_A\}.$$

Также справедливы следующие предложения.

Предложение 3. Область определения $\aleph_A$ конкретного алгоритма A представима в виде объединения классов эквивалентности задач, соответствующих индексам множества $\aleph_{A(M,N)}$:

$$\aleph_A=\bigcup_{(M,N)\in\aleph_{A(M,N)}}\aleph(M,N).$$

Предложение 4. Следующие утверждения эквивалентны
$(M,N)\in\aleph_{A(M,N)}\Leftrightarrow\aleph(M,N)\subseteq\aleph_A$.

Предложение 5. С учетом указанной *взаимной однозначной зависимости* классов эквивалентности задач $\aleph(M,N)$ и пар индексов (M,N) в дальнейшем будем считать *синонимами область определения* $\aleph_A$ алгоритма *и множество пар индексов* $\aleph_{A(M,N)}$, порождающих $\aleph_A$. Наименование «*область определения*» ниже будет применяться к множеству $\aleph_{A(M,N)}$ так же, как и к множеству $\aleph_A$. Следующие (части) высказывания далее считаются эквивалентными:

$$\forall Z\in\aleph_A\quad U(A(Z))\ldots\ \Leftrightarrow\ \forall(M,N)\in\aleph_{A(M,N)}\quad u_A(M,N)\ldots$$

Также верно и следующее

Предложение 6. Для любых алгоритмов постоянной сложности A_1,A_2 с областями определения $\aleph_{A_1}=\aleph_{A_2}\equiv\aleph_A$, порождаемыми множествами индексов $\aleph_{A_1(M,N)}=\aleph_{A_2(M,N)}\equiv\aleph_{A(M,N)}$, следующие утверждения эквивалентны:

$$\forall Z\in\aleph_A\quad U(A_1(Z))\le U(A_2(Z))\ \Leftrightarrow\ \forall(M,N)\in\aleph_{A(M,N)}\quad u_{A_1}(M,N)\le u_{A_2}(M,N).$$

Справедливость последнего предложения не меняется при замене обоих знаков "$\le$" на любые другие знаки отношения пар числовых величин.

1.5. Основные классы алгоритмов решения задачи Z вычисления свертки

В настоящее время существует большое число алгоритмов решения задачи Z вычисления одномерной свертки. В соответствии с принципами построения, это множество алгоритмов может быть подразделено на три классы (подмножества) алгоритмов. К ним относятся:

- класс алгоритмов прямого вычисления свертки $\{A_{DC}\}$,
- класс алгоритмов вычисления свертки на основе дискретных ортогональных преобразований (ДОП) $\{A_{FC}\}$,
- класс алгоритмов рекурсивного вычисления свертки $\{A_{RF}\}$.

Для этих классов алгоритмов справедливы следующие три предложения.

Предложение 7. Алгоритм прямого вычисления свертки является алгоритмом постоянной сложности.

Предложение 8. Алгоритмы вычисления свертки на основе ДОП являются алгоритмами постоянной сложности.

Предложение 9. Алгоритмы рекурсивного вычисления свертки являются алгоритмами вариантной сложности.

2. Алгоритмы постоянной сложности и их свойства

Выделение в отдельное подмножество алгоритмов постоянной сложности позволяет указать некоторые их особенности и выделить ряд полезных свойств таких алгоритмов и/или их подмножеств.

2.1. Распространение алгоритма постоянной сложности

Действие любого алгоритма постоянной сложности определено на области определения $\aleph_A$, порождаемой множеством индексов $\aleph_{A(M,N)}$. При этом, как правило $\aleph_{A(M,N)}\ne\{(M,N):\ M\ge1,N\ge1\}$, то есть множество индексов, связанное с конкретным алгоритмом, не покрывает все возможное множество индексов. Следовательно, ограничена и сама область определения $\aleph_A$. Данный факт приводит к необходимости распространения алгоритма. То есть построения на его основе другого алгоритма, область определения которого была бы заданной, при-

чем на ее подобласти $\aleph_A$ алгоритмы были бы подобными (см. определение 13).

Следует отметить, что способы построения распространения алгоритма могут быть различными. В частности, они могут учитывать специфику архитектуры конкретного алгоритма, специфику данных и т.д. В то же время, следуя принципам теории алгоритмов [21] и понятию «нормальный алгоритм», целесообразен такой способ построения распространения, который мог бы быть применен к любому алгоритму постоянной сложности. Несколько подобных способов следуют непосредственно из выражения (1) вычисления свертки. К ним относятся следующие способы.

Распространение алгоритма путем решения «суперзадачи» задачи Z

Произведем замену задачи $Z\left(\Im_0, \{x(n)\}_{n=0}^{N-1}\right), \Im_0 = \left(\{h(n)\}_{n=0}^{M-1}, \Im_x\right)$ на «суперзадачу» вида:

$$Z' = Z\left(\Im_0', \{x'(n)\}_{n=0}^{N+n+m-1}\right), \Im_0 = \left(\{h'(n)\}_{n=0}^{M+m-1}, \Im_x\right),$$

где $m \geq 0, n \geq 0$.

Для «суперзадачи» входной сигнал и ИХ задаются следующим образом:

$$x'(n) = \begin{cases} x(n), & n = \overline{0, N-1}, \\ 0, & n = \overline{N, N+n+m-1}. \end{cases}$$

$$h'(m) = \begin{cases} h(m), & m = \overline{0, M-1}, \\ 0, & m = \overline{M, M+m-1}. \end{cases}$$

Замена задачи на «суперзадачу» правомерна, поскольку количество отсчетов в полезной части выходного сигнала суперзадачи оказывается не меньше, чем для исходной задачи *Z:*

$$(N+n+m-1)-(M+m-1)+1 = N-M+1+n \geq \geq N-M+1.$$

Очевидно, что сложность решения исходной задачи *Z* для построенного распространения алгоритма $\vec{\vec{A}}$ составляет величину:

$$u_{\vec{\vec{A}}}(M,N) = \begin{pmatrix} u_A(M+m, N+m+n)\cdot \\ \cdot\dfrac{(N-M+n+1)}{(N-M+1)} \end{pmatrix}. \qquad (5)$$

Распространение алгоритма путем разбиения задачи Z на подзадачи с сегментацией ИХ

Представим соотношение (1), задающее выражение для свертки, в эквивалентном виде:

$$y(n) = \sum_{m=0}^{M-1} h(m)x(n-m) = \\ = \sum_{m=0}^{M_1-1} h(m)x(n-m) + \sum_{m=M_1}^{M-1} h(m)x(n-m), \\ n = \overline{M-1, N-1}.$$

Определим дополнительно величины

$$x'(n) = \begin{cases} x(n-M_1), & n = \overline{0, N-M_1-1}, \\ 0, & \text{иначе.} \end{cases}$$

$$h_1(m) = \begin{cases} h(m), & m = \overline{0, M_1-1}, \\ 0, & \text{иначе.} \end{cases}$$

$$h_2(m) = \begin{cases} h(m), & m = \overline{0, M-M_1-1}, \\ 0, & \text{иначе.} \end{cases}$$

Тогда выражение для свертки перепишется:

$$y(n) = \sum_{m=0}^{M_1-1} h_1(m)x(n-m) + \sum_{m=0}^{M-M_1-1} h_2(m)x(n-m), \\ n = \overline{M-1, N-1},$$

и решение одной задачи *Z* производится путем решения двух «подзадач», каждая из которых имеет тот же (возможно сдвинутый) входной сигнал, что и исходная задачи, и ИХ, которая является частью исходной ИХ.

Очевидно, что сложность решения исходной задачи *Z* для построенного распространения алгоритма $\vec{\vec{A}}$ составляет величину:

$$u_{\vec{\vec{A}}}(M,N) = u_A(M_1, N-(M-M_1)) + \\ u_A(M-M_1, N-M_1) + \xi_{add}. \qquad (6)$$

Следует отметить, что для конкретного набора индексов (M,N), которые характеризуют требуемую для решения задачу Z, в области определения исходного алгоритма может не оказаться ни одной пары $(M_1, N-(M-M_1)), (M-M_1, N-M_1)$. В этом случае для построения распространения можно использовать «гибридный» способ, который включает в себя и первый, и второй способ одновременно. А именно, на основании существующих в области определения алгоритма *A* пар $(M,N) \in \aleph_A$ построить с помощью рассматриваемого способа решение некоторой задачи, которая для искомой задачи является «суперзадачей». А затем воспользоваться первым способом. Одновременное использование сразу нескольких способов для построения распространения алгоритма постоянной сложности может применяться и совместно с третьим способом, рассматриваемым ниже.

Распространение алгоритма путем разбиения задачи Z на подзадачи с сегментацией входного сигнала

Из входного сигнала $\{x(n)\}_{n=0}^{N-1}$ синтезируем два сигнала $\{x_1(n)\}_{n=0}^{N_1-1}$ и $\{x_2(n)\}_{n=0}^{N-N_1+M-2}$ следующим образом:

$$x_1(n) = \begin{cases} x(n), & n = \overline{0, N_1-1}, \\ 0, & \text{иначе.} \end{cases}$$

$$x_2(n) = \begin{cases} x(n+N_1-M), & n = \overline{0, N-N_1+M-2}, \\ 0, & \text{иначе.} \end{cases}$$

Тогда значения отсчетов искомой свертки могут быть получены:

$$y(n)=\sum_{m=0}^{M-1}h(m)x_1(n-m),\quad n=\overline{M-1,N_1-1},$$
$$y(n+N_1-M+1)=\sum_{m=0}^{M-1}h(m)x_2(n-m),$$
$$n=\overline{M-1,N-N_1+M-2}.$$

Тогда решение одной задачи Z производится путем решения двух «подзадач», каждая из которых имеет ту же ИХ, что и исходная задачи, и более короткий входной сигнал, который является частью исходного сигнала. Подобная замена правомерна, поскольку количество получаемых в этих двух подзадачах отсчетов полезной части выходного сигнала оказывается таким же, как и у исходной задачи *Z:*

$$(N_1-M+1)+((N-N_1+M-2)-(M-1)+1)=$$
$$=N-M+1.$$

Сложность решения исходной задачи *Z* для построенного распространения алгоритма $\vec{A}$ составляет величину:

$$u_{\vec{A}}(M,N)=\frac{\begin{pmatrix}u_A(M,N_1)(N_1-M+1)+\\+u_A(M,N-N_1+M-1)(N-N_1)\end{pmatrix}}{N-M+1}. \quad (7)$$

Приведенные способы построения распространения алгоритма постоянной сложности, вероятно, могут быть дополнены. Однако, их рассмотрение и анализ не является предметом детального исследования данной работы.

Поэтому далее предполагается, что распространение любого алгоритма постоянной сложности производится некоторым образом, одним или несколькими из рассмотренных способов.

2.2. Приведенные алгоритмы постоянной сложности

Возможность распространения алгоритма постоянной сложности на задачи, которые не входят изначально в его область определение, имеет очевидное следствие. Действительно, если возможно определить работу алгоритма для задач, которые соответствуют парам $(M,N)\notin\aleph_{A(M,N)}$, то аналогичным образом можно рассчитать сложность и для пар $(M,N)\in\aleph_{A(M,N)}$, если таких пар более одной. И в случае, если для конкретной пары $(M,N)\in\aleph_{A(M,N)}$ реальная сложность алгоритма $u_A(M,N)$ оказывается выше сложности у алгоритма распространения, то целесообразным является «замена» действия алгоритма для задач с параметрами (M,N). Такое неформальное суждение приводит к следующим двум понятиям: корректной сложности и приведенным алгоритмам постоянной сложности.

Определение 25. Сложность $u_A(M,N)$ алгоритма A постоянной сложности называется *корректной*, если для любой пары $(M,N)\in\aleph_{A(M,N)}$ выполняется неравенство:

$$u_A(M,N)\le$$
$$\min\left(\begin{array}{l}\min\limits_{\substack{m=\overline{1,M-2}\\(m,N-(M-m))\in\aleph_{A(M,N)},\\(M-m,N-m)\in\aleph_{A(M,N)}}}\begin{pmatrix}u_A(m,N-(M-m))+\\u_A(M-m,N-m)+\\\xi_{add}\end{pmatrix},\\ \min\limits_{\substack{m=0,1,2...\\n=0,1,2,...\\(M+m,N+m+n)\in\aleph_{A(M,N)}}}\begin{pmatrix}u_A(M+m,N+m+n)\cdot\\\dfrac{(N-M+n+1)}{(N-M+1)}\end{pmatrix},\\ \min\limits_{\substack{n=\overline{1,\lfloor N/2\rfloor}\\(M,n)\in\aleph_{A(M,N)},\\\binom{M,}{N-n+M-1}\in\aleph_{A(M,N)}}}\dfrac{\begin{pmatrix}u_A(M,n)(n-M+1)+\\u_A(M,N-n+M-1)(N-n)\end{pmatrix}}{N-M+1}\end{array}\right). \quad (8)$$

Очевидно, что неравенство (8) на самом деле состоит из трех соотношений, которые взяты из выражений для построения распространения алгоритма (5)-(7):

$$u_A(M,N)\le$$
$$\min_{\substack{m=\overline{1,M-2}\\(m,N-(M-m))\in\aleph_{A(M,N)},\\(M-m,N-m)\in\aleph_{A(M,N)}}}\begin{pmatrix}u_A(m,N-(M-m))+\\u_A(M-m,N-m)+\xi_{add}\end{pmatrix}, \quad (9)$$
$$(M,N)\cdot(N-M+1)\le$$
$$\min_{\substack{m=0,1,2...\\n=0,1,2,...\\(M+m,N+m+n))\in\aleph_{A(M,N)}}}\begin{pmatrix}u_A(M+m,N+m+n)\cdot\\(N-M+n+1)\end{pmatrix},$$
$$u_A(M,N)\cdot(N-M+1)\le \quad (9')$$
$$\min_{\substack{n=\overline{1,\lfloor N/2\rfloor}\\(M,n)\in\aleph_{A(M,N)},\\\binom{M,}{N-n+M-1}\in\aleph_{A(M,N)}}}\begin{pmatrix}u_A(M,n)(n-M+1)+\\u_A(M,N-n+M-1)(N-n)\end{pmatrix}.$$

Определение 26 Алгоритм *A* постоянной сложности называется *приведенным*, если его сложность $u_A(M,N)$ корректна.

В силу выражения (8), справедливо следующее интуитивно понятное соотношение для любого приведенного алгоритма постоянной сложности:

$$u(M,N)\le\min_{\substack{m=1,2,...\\(M+m,N+m))\in\aleph_{A(M,N)}}}u(M+m,N+m). \quad (10)$$

Замечание Определения 25 и 26 не дают никакой информации относительно соотношения вычислительной сложности приведенного алгоритма постоянной сложности для случаев: $u(M,N)$ и $u(M+m,N)$.

Определения 25,26 позволяют выделить среди алгоритмов постоянной сложности подмножество, вычислительная сложность которых не может быть снижена тривиальным способом, подобным одному из способов построения распространения алгоритма. Поскольку подмножество приведенных алгоритмов в общем случае не совпадает с множеством алгоритмов постоянной сложности, покажем, что для остальных алгоритмов, не являющихся приведенными, можно произвести такую тривиальную модификацию. То есть синтезировать на основе существующего «не приведенного» алгоритма новый, который будет по сложности не хуже исходного.

Теорема 1 Для любого алгоритма постоянной сложности A с областью определения $\aleph_{A(M,N)}$ существует (может быть сконструирован) приведенный алгоритм постоянной сложности $\breve{A}$ с той же областью определения $\aleph_{A(M,N)}$, для которого справедливо неравенство $U(\breve{A}) \le U(A)$.

Доказательство: проведем путем построения искомого приведенного алгоритма.

Если алгоритм A является приведенным (его сложность является корректной), тогда определим алгоритм $\breve{A} = A$. Область определения алгоритма $\breve{A}$, очевидно, совпадает с областью определения алгоритма A.

Пусть A не является приведенным алгоритмом. То есть это алгоритм постоянной сложности $u_A(M,N)$ с областью определения $\aleph_{A(M,N)}$, у которого для некоторых пар (M,N) нарушается соотношение (8). Пусть на основании этого алгоритма A формируется новый приведенный алгоритм $\breve{A}$ с областью определения $\aleph_{A(M,N)}$ и со сложностью $u_{\breve{A}}(M,N)$, которая будет определена в процессе доказательства.

Обозначим алгоритм A как A_0, то есть:

$$A_0 = A,\ u_{A_0}(M,N) = u_A(M,N),$$
$$\aleph_{A_0(M,N)} = \aleph_{A(M,N)}.$$

Построим алгоритм A_1 с такой же областью определения $\aleph_{A_1(M,N)} = \aleph_{A_0(M,N)}$ и со сложностью $u_{A_1}(M,N)$. Для этого по функции сложности $u_{A_0}(M,N)$ определим любую пару (M,N), в которых происходит нарушение условия корректности сложности (8). Обозначим эту пару (M_0,N_0). Тогда для всех пар, для которых справедливо условие

$$(M,N) \in \aleph_{A(M,N)} \wedge (M,N) \ne (M_0,N_0),$$

определим действие алгоритма A_1 как выполнения алгоритма A_0. Для пары $(M,N) = (M_0,N_0)$ установим, какое именно из неравенств соотношения (9) нарушено.

Если нарушено первое неравенство соотношения (9) вида:

$$u_{A_0}(M_0,N_0) \le \min_{\substack{m=\overline{1,M-2} \\ (m,N_0-(M_0-m)) \in \aleph_{A(M,N)}, \\ (M_0-m,N_0-m) \in \aleph_{A(M,N)}}} \begin{pmatrix} u_{A_0}(m,N_0-(M_0-m)) + \\ u_{A_0}(M_0-m,N_0-m) + \xi_{add} \end{pmatrix},$$

следовательно, существует $\hat{M}$ $(0 < \hat{M} < M_0)$, такое что

$$u_{A_0}(\hat{M},N_0-(M_0-\hat{M})) + u_{A_0}(M_0-\hat{M},N_0-\hat{M}) + \xi_{add} = \min_{\substack{m=\overline{1,M-2} \\ (m,N-(M-m)) \in \aleph_{A(M,N)}, \\ (M-m,N-m) \in \aleph_{A(M,N)}}} \begin{pmatrix} u_{A_0}(m,N_0-(M_0-m)) + \\ u_{A_0}(M_0-m,N_0-m) + \xi_{add} \end{pmatrix},$$

для которого выполняется строгое неравенство

$$u_{A_0}(M_0,N_0) > u_{A_0}(\hat{M},N_0-(M_0-\hat{M})) + u_{A_0}(M_0-\hat{M},N_0-\hat{M}) + \xi_{add}.$$

Тогда в алгоритме A_1 при решении класса задач с параметрами (M_0,N_0) производится разбиение ИХ длины M_0 на два сегмента размерами $\hat{M}$ и $M_0-\hat{M}$. И решение исходной задач Z с параметрами (M_0,N_0) заменяется решением двух подзадач:

- подзадачи Z1 с параметрами $(\hat{M},N_0)$ и
- подзадачи Z2 с параметрами $(M_0-\hat{M},N_0)$.

для которых поведение алгоритма A_1 определено и совпадает с поведением алгоритма A_0. Разделение на подзадачи Z1 и Z2 правомерно, поскольку длина «полезной» части выходных сигналов в обеих задачах совпадает с длиной «полезной» части выходного сигнала исходной задачи Z: $M_0 = (M_0-\hat{M}) + \hat{M}$.

Входной сигнал длины N_0 для задач Z1 и Z2 также следует преобразовать, взяв первые $N_0 - M_0 + \hat{M}$ отсчетов входного сигнала при решении задачи Z1 и, соответственно, последние $N_0 + \hat{M}$ отсчетов входного сигнала при решении задачи Z2. Выходные сигналы (результаты решения) задач Z1 и Z2 суммируются поэлементно с нулевого отсчета до последнего. Полученный в результате поэлементного сложения сигнал - результат свертки для алгоритма A_1.

Учитывая, что длины «полезной» части выходных сигналов для задач Z, Z1 и Z2 совпадают, величины удельной сложности можно просто складывать. Следовательно, сложность алгоритма для задачи с параметрами (M_0,N_0) составляет величину

$$u_{A_1}(M_0,N_0)=u_{A_1}\left(\widehat{M},N_0-\left(M_0-\widehat{M}\right)\right)+ \\ +u_{A_1}\left(M_0-\widehat{M},N_0-\widehat{M}\right)+\xi_{add},$$

где последнее слагаемое ξ_{add} отвечает за одну операцию поэлементного сложения отсчетов двух сигналов – результатов сверток задач Z1 и Z2. Следовательно, после такого преобразования выражение для сложности $u_{A_1}(M,N)$ в точке (M_0,N_0) будет удовлетворять условию корректности сложности.

Если нарушено второе неравенство соотношения (9) вида:

$$u_{A_0}(M_0,N_0)\cdot(N_0-M_0+1)\le \\ \min_{\substack{m=0,1,2\dots\\ n=0,1,2,\dots\\ (M_0+m,N_0+m+n))\in\aleph_{A(M,N)}}}\begin{pmatrix}u_{A_0}(M_0+m,N_0+m+n)\cdot\\ \cdot(N_0-M_0+n+1)\end{pmatrix},$$

следовательно, также существует пара

$$\left(\widetilde{M},\widetilde{N}\right)\in\aleph_{A(M,N)} \\ \left(\widetilde{M}=M_0+\widetilde{m},\quad \widetilde{N}=N_0+\widetilde{n}+\widetilde{m},\quad \widetilde{m}\ge 0,\widetilde{n}\ge 0\right),$$

такая, что

$$u\left(\widetilde{M},\widetilde{N}\right)\cdot\left(\widetilde{N}-\widetilde{M}+1\right)= \\ =\min_{\substack{m=0,1,2\dots\\ n=0,1,2,\dots\\ (M_0+m,N_0+m+n))\in\aleph_{A(M,N)}}}\begin{pmatrix}u(M_0+m,N_0+m+n)\cdot\\ \cdot(N_0-M_0+n+1)\end{pmatrix},$$

для которой справедливо строгое неравенство

$$u(M_0,N_0)\cdot(N_0-M_0+1)>u\left(\widetilde{M},\widetilde{N}\right)\cdot\left(\widetilde{N}-\widetilde{M}+1\right). \quad (11)$$

Тогда в алгоритме A_1 при решении класса задач Z с параметрами (M_0,N_0) производится дополнение ИХ и входного сигнала (нулями) до длин $\widetilde{M}\ge M_0,\widetilde{N}\ge N_0$ и производится решение задачи с параметрами $\left(\widetilde{M},\widetilde{N}\right)$. Такая замена справедлива, поскольку «полезная» область обработки в обоих случаях составляет

- для исходной задачи с параметрами (M_0,N_0): N_0-M_0+1,

  для преобразованной задачи с параметрами

$$(M_0,N_0):\begin{pmatrix}\widetilde{N}-\widetilde{M}+1=\\ =\left(N_0+\widetilde{n}+\widetilde{m}\right)-\left(M_0+\widetilde{m}\right)+1=\\ =N_0-M_0+1+\widetilde{n}\end{pmatrix}.$$

То есть область выходных отсчетов преобразованной задачи с параметрами (M_0,N_0) в качестве подобласти содержит требуемые выходные отсчеты исходной задачи с параметрами (M_0,N_0).

Сложность алгоритма A_1 для пары $(M_0,N_0)\in\aleph_{A(M,N)}$ удовлетворяет соотношению

$$u_{A_1}(M_0,N_0)=u_{A_1}\left(\widetilde{M},\widetilde{N}\right)\cdot\frac{\left(\widetilde{N}-\widetilde{M}+1\right)}{(N_0-M_0+1)}<u_{A_0}(M_0,N_0)$$

в силу справедливости неравенства (11).

Если нарушено третье неравенство соотношения (9), поступаем аналогично первым двум случаям.

Если нарушены все неравенства в соотношении ((9), тогда действие A_1 определяется исходя из того, какая из коррекций даст наименьшее значение сложности в точке (M_0,N_0).

Сложность полученного на *l*-ом шаге алгоритма A_l $(l=1,2,\dots)$ проверяется на удовлетворение условию корректности сложности (8). Если условие выполняется, тогда текущий алгоритм A_l - искомый, и полагается $\breve{A}=A_l$. Осталось доказать, что существует некоторый шаг *l,* когда это условие выполниться, то есть когда итерационный процесс завершиться.

Доказательство того, что итерационный процесс перехода от алгоритма A_{l-1} к алгоритму A_l завершится $(l=1,2,\dots)$.

Построение алгоритма A_l на основе алгоритма A_{l-1} приводит к тому, что сложности полученного на *l*-ом шаге алгоритма справедливо неравенство:

$$u_{A_l}(M,N)\begin{vmatrix}=u_{A_{l-1}}(M,N), & (M,N)\ne(M_{l-1},N_{l-1}),\\ <u_{A_{l-1}}(M,N), & (M,N)=(M_{l-1},N_{l-1})\end{vmatrix} \quad (12)$$

или

$$\forall(M,N)\in\aleph_{A(M,N)} \\ u_{A_l}(M,N)\le u_{A_{l-1}}(M,N)\le\dots \quad (13) \\ \le u_{A_0}(M,N)=u_A(M,N).$$

Введем в рассмотрение числовую последовательность $\{s_l\}_{l=0,1,\dots}$, каждый элемент которой определяется следующим образом:

$$s_l=\sum_{(M,N)\in\aleph_{A(M,N)}}u_{A_l}(M,N).$$

В силу неотрицательности значений функции сложности $u_{A_l}(M,N)$, элементы последовательности $\{s_l\}_{l=0,1,\dots}$ является также неотрицательными числами (ограничены снизу). Кроме того, с учетом справедливости неравенства (12), имеем:

$$s_0>s_1>s_2>s_3>\dots, \quad (14)$$

то есть последовательность $\{s_l\}_{l=0,1,\dots}$ - строго убывающая.

По теореме Вейерштрасса [27], любая убывающая ограниченная снизу последовательность имеет предел, то есть

$$\exists\, k\ge 0\quad \forall t\ge 0:\quad s_{k+t}=s_k.$$

Последнее соотношение означает, в частности, что модификации сложности полученного алгоритма A_k (то есть модификации величин $u_{A_k}(M,N)$ при построении алгоритма A_k) больше не производятся. Такое может произойти, только если функция сложности $u_{A_k}(M,N)$ удовлетворяет ограничениям (8), потому что в противном случае значения сложности $u_{A_k}(M,N)$ были бы уменьшены и, следовательно, была бы уменьшена величина s_k. Обозначим полученный алгоритм $\breve{A} \equiv A_k$.

Таким образом доказано, что алгоритм $\breve{A}$ удовлетворяет условию (8), а это означает, что полученный алгоритм постоянной сложности – приведенный. Кроме того, в силу (13) справедливо:

$$\forall (M,N) \in \aleph_{A(M,N)}$$
$$u_{\breve{A}}(M,N) = u_{A_k}(M,N) \;\le\; u_A(M,N),$$

что для алгоритма постоянной сложности, в силу определения 16 и предложения 5, эквивалентно выполнению искомого неравенства $U(\breve{A}) \le U(A)$, что и требовалось доказать.

■

В случае если A - исходный алгоритм постоянной сложности , а $\breve{A}$ - соответствующий ему приведенный алгоритм, тогда будем говорить, что $\breve{A}$ *построен на основе (построен из)* A и обозначать $A \underset{u(M,N)}{\to} \breve{A}$. Факт «наследования» области определения приведенным алгоритмом, обозначенный в теореме, обозначаем далее:

$$A \underset{u(M,N)}{\to} \breve{A} \;\Rightarrow\; \aleph_{\breve{A}(M,N)} = \aleph_{A(M,N)}.$$

Следствие 2. (Теоремы 1) Пусть $A \underset{u(M,N)}{\to} \breve{A}$. Тогда справедливо соотношение (15) или ему эквивалентное (15'):

$$\forall (M,N) \in \aleph_{A(M,N)} \;\; u_{\breve{A}}(M,N) \le u_A(M,N), \quad (15)$$

$$\forall Z \in \aleph_A \;\; U(\breve{A}(Z)) \le U(A(Z)). \quad (15')$$

Очевидными также являются следующее предложение и его следствие.

Предложение 10. Пусть A – приведенный алгоритм постоянной сложности и $A \underset{u(M,N)}{\to} \breve{A}$.

Тогда $\breve{A} \cong A$.

Следствие 3. (Предложения 10) Пусть A – приведенный алгоритм постоянной сложности и $A \underset{u(M,N)}{\to} \breve{A}$. Тогда $U(\breve{A}) = U(A)$

Лемма 1 (о неподвижной точке)**.** Пусть $A \underset{u(M,N)}{\to} \breve{A}$ $\aleph_{A(M,N)} = \aleph_{\breve{A}(M,N)} \equiv \aleph_{A(M,N)}$. Тогда: $\exists (M,N) \in \aleph_{A(M,N)}$: $u_A(M,N) = u_{\breve{A}}(M,N)$.

Доказательство в целях сокращения изложения не приводится.

Для двух приведенных алгоритмов, построенных из разных алгоритмов постоянной сложности, справедлива

Лемма 2. Пусть

$A_1 \underset{u(M,N)}{\to} \breve{A}_1$, $A_2 \underset{u(M,N)}{\to} \breve{A}_2$, $\aleph_{A_1} \subseteq \aleph_{A_2}$

и $\forall (M,N) \in \aleph_{A(M,N)}$ $u_{A_1}(M,N) \le u_{A_2}(M,N)$.

Тогда

$\forall (M,N) \in \aleph_{A(M,N)}$ $u_{\breve{A}_1}(M,N) \le u_{\breve{A}_2}(M,N)$.

Доказательство в целях сокращения изложения не приводится.

2.3. Компетентный алгоритм над множеством алгоритмов постоянной сложности

Определение 27. *Компетентным алгоритмом* над множеством $\{A\}$ алгоритмов постоянной сложности называется алгоритм $A^{\oplus} \notin \{A\}$ с областью определения $\aleph_{A^{\oplus}(M,N)} = \bigcup_{A \in \{A\}} \aleph_{A(M,N)}$, в котором $\forall (M,N) \in \aleph_{A^S(M,N)}$ решение любой задачи $Z \in \aleph_{A^{\oplus}}$ с параметрами (M,N) осуществляется наилучшим алгоритмом $A^*(Z) \in \{A\}$:
$(M,N) \in \aleph_{A^*(M,N)} \wedge u_{A^*}(M,N) = \min_{A \in \{A\}} u_A(M,N)$.

Как следует из определения, компетентный алгоритм строится таким образом, что для любой задачи (1) с параметрами (M,N) он использует для решения наилучший алгоритм из доступного множества $\{A\}$. При такой стратегии формирования компетентного алгоритма, очевидно, справедливы следующие два предложения.

Предложение 11. Компетентный алгоритм $A^{\oplus}$ над множеством алгоритмов постоянной сложности $\{A\}$ является алгоритмом постоянной сложности.

Доказательство: непосредственно следует из определения компетентного алгоритма.

■

Предложение 12. Пусть $\{A\}$ - множество алгоритмов постоянной сложности, и $A^{\oplus}$ - компетентный алгоритм над этим множеством с областью определения $\aleph_{A^{\oplus}(M,N)}$. Тогда для

$\forall (M,N) \in \aleph_{A^{\oplus}(M,N)}$ компетентный алгоритм $A^{\oplus}$ является наилучшим алгоритмом множества алгоритмов $\{A\} \cup \{A^{\oplus}\}$, при этом справедливы следующие соотношения (16)-(17) и им эквивалентные (16')-(17'):

$$\forall Z \in \bigcup_{A \in \{A\}} \aleph_A \quad \forall A(Z) \in \{A(Z)\} \cup \{A^{\oplus}\} \quad U(A^{\oplus}(Z)) \le U(A(Z)), \tag{16}$$

$$\forall (M,N) \in \aleph_{A^{\oplus}(M,N)} \quad \forall A : A \in \{A\} \wedge (M,N) \in \aleph_{A(M,N)} \quad u_{A^{\oplus}}(M,N) \le u_A(M,N), \tag{16'}$$

$$\forall Z \in \bigcup_{A \in \{A\}} \aleph_A \quad U(A^{\oplus}(Z)) = \min_{A \in \{A(Z)\}} U(A(Z)), \tag{17}$$

$$\forall (M,N) \in \aleph_{A^{\oplus}(M,N)} \quad u_{A^{\oplus}}(M,N) = \min_{\{A:\ A \in \{A\} \wedge (M,N) \in \aleph_{A(M,N)}\}} u_A(M,N). \tag{17'}$$

Доказательство: непосредственно следует из определения компетентного алгоритма.

■

По аналогии с последним предложением может быть доказано, что компетентный алгоритм для любой задачи является эффективным алгоритмом над множеством $\{A\}$ алгоритмов постоянной сложности, но не является строго эффективным.

Непосредственным и очевидным следствием Предложения 11 и Теоремы 1 является тот факт, что на основе компетентного алгоритма может быть построен приведенный компетентный алгоритм. При этом, исходный компетентный алгоритм может быть построен как над множеством «не приведенных» алгоритмов постоянной сложности, так и над соответствующим множеством приведенных алгоритмов. Следующая лемма устанавливает связь между приведенными компетентными алгоритмами над этими двумя множествами.

Лемма 3. Пусть $\{A_i\}_{i \in I}$ - множество из $|I|$ алгоритмов постоянной сложности, из которых построено множество $\{\breve{A}_i\}_{i \in I}$ приведенных алгоритмов постоянной сложности: $A_i \underset{u(M,N)}{\to} \breve{A}_i \quad (i \in I)$. Пусть $\breve{A}_1^{\oplus}$ - приведенный компетентный алгоритм над множеством $\{A_i\}_{i \in I}$ с областью определения $\aleph_{\breve{A}_1^{\oplus}}$, а $\breve{A}_2^{\oplus}$ - приведенный компетентный алгоритм над множеством $\{\breve{A}_i\}_{i \in I}$ с областью определения $\aleph_{\breve{A}_2^{\oplus}}$. Тогда выполняются соотношения:

$$\aleph_{\breve{A}_1^{\oplus}} = \aleph_{\breve{A}_2^{\oplus}} \left(\equiv \aleph_{\breve{A}^{\oplus}}\right), \tag{18}$$

$$\forall Z \in \aleph_{\breve{A}^{\oplus}} \quad U(\breve{A}_1^{\oplus}(Z)) = U(\breve{A}_2^{\oplus}(Z)). \tag{19}$$

Доказательство:

Часть 1. Доказываем соотношение (18).

Для доказательства достаточно показать, что при выполнении условий леммы справедливо соотношение, эквивалентное (18):

$$\aleph_{\breve{A}_1^{\oplus}(M,N)} = \aleph_{\breve{A}_2^{\oplus}(M,N)}. \tag{18'}$$

Пусть алгоритмы постоянной сложности исходного множества $\{A_i\}_{i \in I}$ имеют следующее множество областей определения $\{\aleph_{A(M,N)}\}_{A \in \{A\}}$. Тогда, в соответствие с Теоремой 1:

$$\forall i \in I \quad A_i \underset{u(M,N)}{\to} \breve{A}_i \quad \Rightarrow \quad \aleph_{\breve{A}_i(M,N)} = \aleph_{A_i(M,N)}.$$

В соответствие с определением 27 компетентного алгоритма, его область определения задается как объединение областей определений частных алгоритмов. Тогда для компетентного алгоритма $A_1^{\oplus}$ над множеством $\{A_i\}_{i \in I}$ получаем:

$$\aleph_{A_1^{\oplus}(M,N)} = \bigcup_{i \in I} \aleph_{A_i(M,N)},$$

а для компетентного алгоритма $A_2^{\oplus}$ над множеством приведенных алгоритмов $\{\breve{A}_i\}_{i \in I}$ имеем:

$$\aleph_{A_2^{\oplus}(M,N)} = \bigcup_{i \in I} \aleph_{\breve{A}_i(M,N)} = \bigcup_{i \in I} \aleph_{A_i(M,N)}.$$

Из последних двух соотношений справедливо равенство областей определений для компетентных алгоритмов:

$$\aleph_{A_1^{\oplus}(M,N)} = \aleph_{A_2^{\oplus}(M,N)}. \tag{20}$$

Далее, из алгоритмов $A_1^{\oplus}$ и $A_2^{\oplus}$ строятся приведенные алгоритмы, соответственно, $\breve{A}_1^{\oplus}$ и $\breve{A}_2^{\oplus}$. Тогда, в соответствие с Теоремой 1, справедливо:

$$A_1^{\oplus} \underset{u(M,N)}{\to} \breve{A}_1^{\oplus} \quad \Rightarrow \quad \aleph_{\breve{A}_1^{\oplus}(M,N)} = \aleph_{A_1^{\oplus}(M,N)},$$

$$A_2^{\oplus} \underset{u(M,N)}{\to} \breve{A}_2^{\oplus} \quad \Rightarrow \quad \aleph_{\breve{A}_2^{\oplus}(M,N)} = \aleph_{A_2^{\oplus}(M,N)}. \tag{21}$$

И соотношение (18') и эквивалентное ему искомое соотношение (18) следуют непосредственно из (20) и (21).

Часть 2. Доказываем соотношение (19).

Пусть $A_1^{\oplus}$ и $A_2^{\oplus}$ - компетентные алгоритмы над множествами $\{A_i\}_{i \in I}$ и $\{\breve{A}_i\}_{i \in I}$, соответственно. Пусть также $A_1^{\oplus} \underset{u(M,N)}{\to} \breve{A}_1^{\oplus}$ и $A_2^{\oplus} \underset{u(M,N)}{\to} \breve{A}_2^{\oplus}$.

Часть 2а. Докажем вначале, что

$$\forall (M,N) \in \aleph_{\breve{A}^{\oplus}(M,N)} \quad u_{\breve{A}_1^{\oplus}}(M,N) \ge u_{\breve{A}_2^{\oplus}}(M,N). \tag{22}$$

В соответствии с Теоремой 1 справедливо:

$$\forall\, i \in I \;\; \forall\, (M,N) \in \aleph_{A_i(M,N)} \quad u_{A_i}(M,N) \geq u_{\breve{A}_i}(M,N),$$

откуда

$$\forall\, (M,N) \in \bigcup_{i\in I} \aleph_{A_i(M,N)} \quad \min_{i\in I} u_{A_i}(M,N) \geq \min_{i\in I} u_{\breve{A}_i}(M,N).$$

Далее, в соответствие с выражением (17) Предложения 12, выражения, стоящие в неравенстве – это значения сложности компетентных алгоритмов с одинаковой областью определения (по первой части доказательства) $\bigcup_{i\in I} \aleph_{A_i(M,N)} = \aleph_{A^{\oplus}(M,N)}$, поэтому

$$\forall\, (M,N) \in \aleph_{A^{\oplus}(M,N)} \quad u_{A_1^{\oplus}}(M,N) \geq u_{A_2^{\oplus}}(M,N).$$

Последнее неравенство и данные $A_1^{\oplus} \underset{u(M,N)}{\rightarrow} \breve{A}_1^{\oplus}$ и $A_2^{\oplus} \underset{u(M,N)}{\rightarrow} \breve{A}_2^{\oplus}$ приводят, на основании Леммы 2, к соотношению (22). Часть 2а завершена.

Часть 2б. Докажем теперь, что

$$\forall\, (M,N) \in \aleph_{\breve{A}^{\oplus}(M,N)} \quad u_{\breve{A}_1^{\oplus}}(M,N) \leq u_{\breve{A}_2^{\oplus}}(M,N). \tag{23}$$

В соответствие с Предложением 12, для компетентного алгоритма справедливо соотношение (16'):

$$\forall\, (M,N) \in \aleph_{A^{\oplus}(M,N)} \;\; \forall\, A: \;\; A \in \{A\} \wedge (M,N) \in \aleph_{A(M,N)} \quad u_{A^{\oplus}}(M,N) \leq u_A(M,N).$$

Тогда, в соответствии с Леммой 2, если $A^{\oplus} \underset{u(M,N)}{\rightarrow} \breve{A}_1^{\oplus}$ и $A_i \underset{u(M,N)}{\rightarrow} \breve{A}_i \;\; (i \in I)$, имеем

$$\forall\, (M,N) \in \aleph_{A^{\oplus}(M,N)} \;\; \forall\, \breve{A}: \;\; \breve{A} \in \{\breve{A}_i\}_{i\in I} \wedge (M,N) \in \aleph_{\breve{A}_i(M,N)} \quad u_{\breve{A}^{\oplus}}(M,N) \leq u_{\breve{A}}(M,N),$$

что эквивалентно

$$\forall\, (M,N) \in \aleph_{A^{\oplus}(M,N)} \quad u_{\breve{A}_1^{\oplus}}(M,N) \leq \min_{\breve{A}\in\{\breve{A}_i\}_{i\in I}} u_{\breve{A}}(M,N) = u_{A_2^{\oplus}}(M,N),$$

где $A_2^{\oplus}$ - компетентный алгоритм над множеством приведенных алгоритмов постоянной сложности $\{\breve{A}_i\}_{i\in I}$. Далее на основании сравниваемых в этом неравенстве алгоритмов $\breve{A}_1^{\oplus}$ и $A_2^{\oplus}$ формируются приведенные алгоритмы: $\breve{A}_1^{\oplus} \underset{u(M,N)}{\rightarrow} \breve{\breve{A}}_1^{\oplus}$, $A_2^{\oplus} \underset{u(M,N)}{\rightarrow} \breve{A}_2^{\oplus}$. Тогда, в соответствие с Леммой 2 и подобности алгоритмов $\breve{A}_1^{\oplus} \cong \breve{\breve{A}}_1^{\oplus}$ (предложение 10 и следствие 3), получаем продолжение в виде:

$$\forall\, (M,N) \in \aleph_{A^{\oplus}(M,N)} \quad u_{\breve{A}_1^{\oplus}}(M,N) \leq u_{\breve{A}_2^{\oplus}}(M,N),$$

что тождественно неравенству (23).

Далее, поскольку нестрогие неравенства (23) и (22) должны выполняться совместно, следовательно, справедливо равенство:

$$\forall\, (M,N) \in \aleph_{A^{\oplus}(M,N)} \quad u_{\breve{A}_1^{\oplus}}(M,N) = u_{\breve{A}_2^{\oplus}}(M,N),$$

или ему эквивалентное

$$\forall\, Z \in \aleph_{A^{\oplus}} \quad U\left(\breve{A}_1^{\oplus}(Z)\right) = U\left(\breve{A}_2^{\oplus}(Z)\right),$$

которое и требовалось доказать.

■

Компетентный алгоритм обладает удобным свойством, отражаемым леммой, приведенной после следующего вспомогательного определения. [1]

Определение 28. *Покрытием интервала* $D = [a,b]$ называется множество непустых дискретных интервалов $\{D_s\}_{s=0}^{S-1}$ вида $D_s = [d_0^s, d_1^s]$, для которых выполняются условия:

$$d_1^s \geq d_0^s,\; D_s \subseteq D,\; D_s \neq D_j \quad \textit{при } s \neq j, \quad s,j = \overline{0,S-1}; \quad \bigcup_{s=\overline{0,S-1}} D_s = D \tag{24}$$

Очевидно, покрытие задается не однозначно.

Лемма 4. Пусть $\breve{A}$ - приведенный алгоритм постоянной сложности со сложностью $u_{\breve{A}}(M,N)$. Тогда для любого покрытия $\{D_s\}_{s=0}^{S-1}$ области $D = [0, M-1]$ справедливо:

$$u_{\breve{A}}(M,N) \leq \sum_{s=0}^{S-1} u_{\breve{A}}\left(|D_s|, N-M+|D_s|\right) + (S-1)\xi_{add}. \tag{25}$$

Доказательство:

Производится непосредственной проверкой справедливости неравенства (25).

■

Лемма 5. Пусть $\{A_i\}_{i\in I}$ - множество алгоритмов постоянной сложности $\{u_{A_i}(M,N)\}_{i\in I}$. Пусть $\breve{A}^{\oplus}$ -

---

[1] Равенство и неравенство двух интервалов $D_s = [d_0^s, d_1^s]$ и $D_j = [d_0^s, d_1^s]$ определяется следующим образом $(s \neq j)$:

$$D_s = D_j \Leftrightarrow \left(d_0^s = d_0^j \wedge d_1^s = d_1^j\right),$$

$$D_s \neq D_j \Leftrightarrow \left(d_0^s \neq d_0^j \vee d_1^s \neq d_1^j\right).$$

приведенный компетентный алгоритм над множеством $\{A_i\}_{i\in I}$ со сложностью $u_{\breve{A}^{\oplus}}(M,N)$. Пусть задана произвольная функция $i(s)$: $\begin{matrix}\{s\}_{s=0}^{S-1}\to I\\ s\ \mapsto\ i\end{matrix}$. Тогда для любого покрытия $\{D_s\}_{s=0}^{S-1}$ области $D=[0,M-1]$ справедливо:

$$\begin{aligned} &u_{\breve{A}^{\oplus}}(M,N)\le\\ &\le\sum_{s=0}^{S-1}u_{\breve{A}^{\oplus}}\left(|D_s|,N-M+|D_s|\right)+(S-1)\xi_{add}\\ &\le\sum_{s=0}^{S-1}u_{A_{i(s)}}\left(|D_s|,N\right)+(S-1)\xi_{add}. \end{aligned}\qquad(26)$$

Доказательство:

Доказываемое соотношение (26) состоит из двух неравенств. Второе неравенство

$$\begin{aligned} &\sum_{s=0}^{S-1}u_{\breve{A}^{\oplus}}\left(|D_s|,N-M+|D_s|\right)+(S-1)\xi_{add}\le\\ &\qquad\le\sum_{s=0}^{S-1}u_{A_{i(s)}}\left(|D_s|,N\right)+(S-1)\xi_{add} \end{aligned}\qquad(27)$$

является непосредственным следствием свойств компетентного (16') и приведенного (15) алгоритмов:

$$\begin{aligned} &\forall\,(M,N)\in\aleph_{A^{\oplus}(M,N)}\\ &\forall\ A:\quad A\in\{A\}\wedge(M,N)\in\aleph_{A(M,N)}\\ &u_{\breve{A}^{\oplus}}(M,N)\le u_{A^{\oplus}}(M,N)\le u_A(M,N). \end{aligned}$$

Первое неравенство (26), формулируемое в виде

$$\begin{aligned} &u_{\breve{A}^{\oplus}}(M,N)\le\\ &\sum_{s=0}^{S-1}u_{\breve{A}^{\oplus}}\left(|D_s|,N-M+|D_s|\right)+(S-1)\xi_{add}, \end{aligned}\qquad(28)$$

является следствием Леммы 4, поскольку $\breve{A}^{\oplus}$ является приведенным алгоритмом постоянной сложности.

Объединяя доказанные неравенства (27) и (28) в одно, получаем искомое неравенство (26). Что и требовалось доказать.

■ Доказанные леммы 4-5 устанавливают следующий замечательный факт. При решении задачи вычисления свертки с параметрами (M,N) множеством алгоритмов постоянной сложности наименьшую сложность будет иметь приведенный компетентный алгоритм, решающий именно исходную задачу с параметрами (M,N). Причем попытка построения лучшего покрытия области определения ИХ (и соответствующих – лучших – алгоритмов) из рассматриваемого множества может только увеличить эту сложность.

3. Построение замыкания множества алгоритмов. Индуцированный алгоритм и основные теоремы о его эффективности

3.1. Модель алгоритмов CR. Построение замыкания по модели CR множества алгоритмов

Пусть решается задача $Z=Z\left(\Im_0,\{x(n)\}_{n=0}^{N-1}\right)$ с априорной информацией $\Im_0=\left(\{h(n)\}_{n=0}^{M-1},\left(N,\Im_{(x)}\right)\right)$. Введем в рассмотрение две конечных ИХ $\{g_h(k)\}_{k=0}^{K_h-1}$ и $\{g_x(k)\}_{k=0}^{K_x-1}$, отсчеты которых удовлетворяют условию:

$$\begin{aligned} &g_h(0)\ne 0,\quad g_h(K_h-1)\ne 0,\\ &g_x(0)\ne 0,\quad g_x(K_x-1)\ne 0. \end{aligned}\qquad(29)$$

Считаем, что длины K_h и K_x этих ИХ-к и собственно значения их отсчетов определяются некоторым образом по априорной информации о задаче $\Im_0$. В частности, ИХ $\{g_h(k)\}_{k=0}^{K_h-1}$ определяется значениями ИХ $\{h(n)\}_{n=0}^{M-1}$, а ИХ $\{g_x(k)\}_{k=0}^{K_x-1}$ определяется априорной информацией о сигнале $\Im_x=\left(N,\Im_{(x)}\right)$.

Рассмотрим далее результаты линейного преобразования исходного сигнала $x(n)$ и ИХ $h(m)$ с использованием введенных ИХ-к, а именно сигналы:

$$\widetilde{h}(m)=\sum_{k=0}^{K_h-1}g_h(k)h(m-k),\quad n=\overline{0,M+K_h-2}\ ,\qquad(30)$$

$$\widetilde{x}(n)=\sum_{k=0}^{K_x-1}g_x(k)x(n-k),\quad n=\overline{0,N-1}\,.\qquad(31)$$

Введенные сигналы $\{g_h(k)\}_{k=0}^{K_h-1}$ и $\{g_x(k)\}_{k=0}^{K_x-1}$, исходные и преобразованные сигналы и ИХ $x(n)$, $h(m)$, $\widetilde{x}(n)$, $\widetilde{h}(m)$ очевидным образом связаны со значениями выходного сигнала $y(n)$ задачи Z очевидным образом:

$$\begin{aligned} &\sum_{k=0}^{K_h-1}\sum_{l=0}^{K_x-1}g_h(k)g_x(l)y(n-k-l)=\\ &=\sum_{m=0}^{M-1}\widetilde{h}(m)\widetilde{x}(n-m)=\\ &=\sum_{m=0}^{M-1}\left(\sum_{k=0}^{K_h-1}g_h(k)h(m-k)\right)\left(\sum_{l=0}^{K_x-1}g_x(l)x(n-m-l)\right),\\ &\qquad n=\overline{M-1,N-1}. \end{aligned}$$

Определив, для удобства, обозначение

$$\begin{aligned} &g(t)=\sum_{k=\max(0,t-K_x)}^{\min(K_h-1,t)}g_h(k)g_x(t-k),\\ &\qquad t=\overline{0,K_x+K_h-2}, \end{aligned}\qquad(32)$$

и учитывая выполнение условий (29), получим следующее выражение:

$$g(0)y(n)+\sum_{t=1}^{K_h+K_x-2}g(t)y(n-t)=$$
$$=\sum_{m=0}^{M-1}\left(\sum_{k=0}^{K_h-1}g_h(k)h(m-k)\right)\left(\sum_{l=0}^{K_x-1}g_x(l)x(n-m-l)\right),$$
$$n=\overline{M-1,N-1}.$$

В силу соотношений (29) и (32), значение $g(0)=g_h(0)g_x(0)\neq 0$, поэтому полученное выражение может быть переписано в виде, который задает альтернативное представление для выражения (1):

$$y(n)=\sum_{t=1}^{K_h+K_x-2}\tilde{g}(t)y(n-t)+$$
$$+\sum_{m=0}^{M-1}\left(\sum_{k=0}^{K_h-1}\tilde{g}_h(k)h(m-k)\right)\left(\sum_{l=0}^{K_x-1}\tilde{g}_x(l)x(n-m-l)\right), \quad (33)$$
$$n=\overline{0,N-1}.$$

Замена диапазона выходного сигнала с $\overline{M-1,N-1}$ на $\overline{0,N-1}$ вызвана сменой интерпретации получаемых соотношений. В частности, соотношение, предшествующее (33), рассматривается как равенство. А соотношение (33) рассматривается как способ вычисления значений сигнала $y(n)$ через его предшествующие значения. В последнем случае необходимо рассчитывать и значения сигнала $y(n)$ на интервале $\overline{0,M-2}$.

В выражении (33) приняты обозначения:

$$\tilde{g}(t)=\frac{g(t)}{g(0)},\quad t=\overline{1,K_h+K_x-2},$$
$$\tilde{g}_h(k)=\frac{g_h(k)}{g_h(0)},\quad k=\overline{0,K_h-1},\quad .$$
$$\tilde{g}_x(k)=\frac{g_x(k)}{g_x(0)},\quad k=\overline{0,K_x-1}.$$

Из этих выражений, очевидно, следует: $\tilde{g}_h(0)=\tilde{g}_x(0)=1$. Таким образом, далее без ограничений общности изложения удобно положить: $g_h(0)=g_x(0)=1$.

Тогда

$$\tilde{g}_h(k)=g_h(k),\quad k=\overline{0,K_h-1},$$
$$\tilde{g}_x(k)=g_x(k),\quad k=\overline{0,K_x-1}.$$

Примем во внимание, что априорная информация $\Im_0=\left(\left\{h(n)\right\}_{n=0}^{M-1},\left(N,\Im_{(x)}\right)\right)$ о задаче Z доступна задолго до момента ее решения (см. п. 1.1). Тогда вычисление выражения (30), расположенного в ПЧ соотношения (33) может быть произведено заблаговременно. Таким образом, вид ИХ $\tilde{h}(m)$ известен до решения задачи Z.

Для дальнейшего изложения введем ряд определений.

Определение 29. *Допустимым покрытием области определения D функции* $\tilde{h}(m)$, заданной на дискретном интервале $D=[a,b]$, называется множество непустых дискретных интервалов $\{D_s\}_{s=0}^{S-1}$ вида $D_s=[d_s,d_{s+1}]$, для которых выполняются условия:

$$d_{s+1}\geq d_s,\quad D_s\subseteq D,\quad D_s\neq D_j$$
$$при\quad s\neq j,\quad s,j=\overline{0,S-1};\quad \bigcup_{s=\overline{0,S-1}}D_s\subseteq D \quad (34)$$

$$и\ \forall\quad m\in D\setminus\bigcup_{s=\overline{0,S-1}}D_s\quad \tilde{h}(m)\equiv 0. \quad (35)$$

Очевидно, что любое покрытие области определения *D*, введенное ранее определением 28, является допустимым. Однако не всякое допустимое покрытие является покрытием. Так же как и покрытие, допустимое покрытие задается не однозначно. Поэтому вводятся следующие два определения.

Определение 30. Два допустимых покрытия $\{D_s^1\}_{s=0}^{S_1-1}$ и $\{D_t^2\}_{t=0}^{S_2-1}$ области определения *D* функции $\tilde{h}(m)$, заданной на дискретном интервале $D=[a,b]$, называются *различными* если условие:

$$(S_1\neq S_2)\ \vee(S_1=S_2\ \wedge\ \exists\, s\in[0,S_1-1]:$$
$$:\forall\, t\in[0,S_2-1]\quad D_s^1\neq D_t^2).$$

Определение 31. Множеством допустимых покрытий области определения D функции $\tilde{h}(m)$, заданной на дискретном интервале $D=[a,b]$, называется объединение всех различных допустимых покрытий.

В силу конечности области определения *D* ИХ $\tilde{h}(m)$, заданной на дискретном интервале, множество допустимых покрытий этой области определения также конечно.

Пусть, далее, для ИХ $\tilde{h}(m)$, заданной выражением (30) определено некоторое допустимое покрытие $\{D_s\}_{s=0}^{S-1}$ ее области определения. Тогда справедливо следующее представление для ИХ:

$$\tilde{h}(m)=\sum_{s=0}^{S-1}\tilde{h}_s(m), \quad (36)$$

где функции $\{\tilde{h}_s(m)\}_{s=0}^{S-1}$ определены таким образом, чтобы обеспечить выполнение равенства (36).

Например:

$$\forall s = \overline{0,S-1} \quad (\ )$$

$$\widetilde{h}_s(m) = \begin{cases} \dfrac{\widetilde{h}(m)}{\left|\left\{D_t : m \in D_t, \quad t = \overline{0,S-1}\right\}\right|}, & m \in D_s, \\ 0, & m \notin D_s. \end{cases} \quad (37)$$

Другой возможный способ определения:

$$\forall s = \overline{0,S-1}$$

$$\widetilde{h}_s(m) = \begin{cases} \widetilde{h}(m), & m \in D_s \ \wedge \\ & s = \arg \min\limits_{t=0,S-1}(t : m \in D_t), \\ 0, & \textit{иначе}. \end{cases} \quad (37')$$

С учетом представления (36) разложения выражение (33) перепишется в окончательном виде:

$$y(n) = \sum_{t=1}^{K_h+K_x-2} \widetilde{g}(t) y(n-t) + \sum_{s=0}^{S-1} \sum_{m \in D_s} \widetilde{h}_s(m) \left(\sum_{l=0}^{K_x-1} \widetilde{g}_x(l) x(n-m-l) \right), \quad (38)$$

$$n = \overline{0,N-1}.$$

Первое слагаемое полученного выражения при $K_h = K_x = 1$ полагается равным нулю:

$$\sum_{t=1}^{K_h+K_x-2} \widetilde{g}(t) y(n-t) = \sum_{t=1}^{0} \widetilde{g}(t) y(n-t) = 0.$$

Модель алгоритма «CR»

Выражение (38) обосновывает принцип построения замыкания множества (быстрых) алгоритмов вычисления свертки (1). Для формализации понятия замыкания опишем вначале модель алгоритма CR вычисления свертки (1), порождаемую полученным выражением (38). Наименование модели «CR» возникает из принципа построения этой модели, связанным с *редукцией свертки (Convolution Reduction)* на свертки меньшей длинны.

Алгоритм «CR»

Шаг 1. *Предобработка входного сигнала*. На этом шаге производится вычисление свертки входного сигнала $x(n)$ и ИХ $\{g_x(k)\}_{k=0}^{K_x-1}$ по формуле (31):

$$\widetilde{x}(n) = \sum_{k=0}^{K_x-1} \widetilde{g}_x(k) x(n-k), \quad n = \overline{0,N-1}. \quad (31')$$

Шаг 2. Для каждого дискретного интервала $D_s = \left[d_0^s, d_1^s\right]$ из множества интервалов в выбранном допустимом покрытии $\{D_s\}_{s=0}^{S-1}$ ИХ $\widetilde{h}(m)$ производится вычисление S сверток по всем интервалам допустимого покрытия $\{D_s\}_{s=0}^{S-1}$:

$$\tilde{y}_s(n) = \sum_{m \in D_s} \tilde{h}_s(m) \tilde{x}(n-m) \quad \left(n = \overline{0,N-1}, \quad s = \overline{0,S-1}\right). \quad (39)$$

Шаг 3. Суммирование результатов сверток (39):

$$\widetilde{y}(n) = \sum_{s=0}^{S-1} \widetilde{y}_s(n). \quad (40)$$

Шаг 4. *Постобработка результата*. На последнем шаге производится рекурсивное вычисление результата решения задачи Z:

$$y(n) = \sum_{t=1}^{K_h+K_x-2} \widetilde{g}(t) y(n-t) + \widetilde{y}(n), \quad n = \overline{0,N-1}. \quad (41)$$

Модель алгоритма - Конец

Параметры модели алгоритма «CR»

Приведенная модель алгоритма «CR», как и выражение (38), отражает лишь принцип вычисления свертки на основании этого выражения. Конкретный алгоритм вычисления свертки (1) определен, если заданы параметры, конкретизирующие все операции в приведенной модели.

А именно, следует определить следующие параметры:

(a) размер K_h ИХ $\{g_h(k)\}_{k=0}^{K_h-1}$ предобработки для КИХ $\{h(n)\}_{n=0}^{M-1}$;

(b) размер K_x ИХ предобработки $\{g_x(k)\}_{k=0}^{K_x-1}$ входного сигнала $\{h(n)\}_{n=0}^{M-1}$;

(c) число S интервалов допустимого покрытия $\{D_s\}_{s=0}^{S-1}$ области определения ИХ $\widetilde{h}(m)$;

(d) отсчеты ИХ $\{g_h(k)\}_{k=0}^{K_h-1} \quad (g_h(0)=1)$;

(e) отсчеты ИХ $\{g_x(k)\}_{k=0}^{K_x-1} \quad (g_x(0)=1)$;

(f) границы интервалов $D_s = \left[d_0^s, d_1^s\right]$ в допустимом покрытии $\{D_s\}_{s=0}^{S-1}$ области определения ИХ $\widetilde{h}(m)$;

(g) способ определения множества функций $\{\widetilde{h}_s(m)\}_{s=0}^{S-1}$, заданных на соответствующих интервалах допустимого покрытия $\{D_s\}_{s=0}^{S-1}$, и представляющих функцию $\widetilde{h}(m)$ в виде разложения (36);

(h) алгоритм A_{prep}, вычисляющий свертку в выражении (31'), то есть решающий задачу

$$Z_{prep} \equiv Z\left(\Im_0^{prep}, \{x(n)\}_{n=0}^{N+K_x-1}\right) \quad (42)$$

с начальной информацией $\Im_0^{prep} = \left(\{g_x(k)\}_{k=0}^{K_x-1}, \left(N+K_x, \Im_{(x)}\right)\right)$ (входной сигнал дополняется нулями до длины $N+K_x$);

(i) набор из S алгоритмов $\{A_s\}_{s=0}^{S-1}$, вычисляющих S сверток в выражении (39), то есть решающих S задач

$$Z_s \equiv Z\left(\Im_0^s, \{\widetilde{x}(n)\}_{n=0}^{N+K_x-1}\right) \quad \left(s = \overline{0,S-1}\right) \quad (43)$$

с различными начальными информациями вида:

$$\Im_0^s = \left(\left\{ \widetilde{h}_s(m) \right\}_{m \in D_s}, \left(N + K_x, \Im_{(\widetilde{x})} \right) \right).$$

Очевидно, что множество параметров может быть разделено на две принципиально различные группы:

- числовые параметры (a)-(g) и
- параметры – алгоритмы (h)-(i).

Ограничения на числовые параметры (a)-(g) возникают только из условий вывода алгоритма. Например, необходимо выполнение условий (29), также существуют ограничения на допустимое покрытие $\{D_s\}_{s=0}^{S-1}$ области определения ИХ $\widetilde{h}(m)$ в виде соотношений (34)-(35). Наконец, выбор множества функций в разложении (36) также не произволен.

Определение 32. Подмножество числовых значений, которые могут принимать числовые параметры (a)-(g) в рамках модели CR, называются *допустимыми значениями числовых параметров* модели CR.

В отличие от ограничений на числовые параметры, ограничения на параметры – алгоритмы (h)-(i) определяются внешними условиями, то есть способностью человека или машины (ПЭВМ) предоставить в распоряжение алгоритма модели CR конкретные алгоритмы вычисления сверток (31') и (32).

Определение 33. Порядком модели CR называется тройка положительных целых чисел (K_h, K_x, S).

В дальнейшем считаем, что для порядков модели CR, то есть для троек чисел (K_h, K_x, S), введено упорядочивание. Упорядочивание производится по значениям числового вектора (K_h, K_x, S) в лексикографическом смысле. Например: $(2,7,6) \prec (8,1,6)$, $(4,1,1) \succ (3,7,7)$.

Сложность алгоритмов модели CR

Относительно явного выражения для сложности следует отметить, что приведенное выше описание модели однозначно определяет сложность любого алгоритма A^{CR} модели CR, используемого для вычисления свертки (1) с использованием выражения (38).

А именно, первый шаг алгоритма модели, на котором производится вычисление свертки по выражению (31'), имеет сложность

$$U_{add}\left(A^{CR}_{Step\_1}\right) = U_{add}\left(A_{prep}\left(Z_{prep}\right)\right),$$
$$U_{mul}\left(A^{CR}_{Step\_1}\right) = U_{mul}\left(A_{prep}\left(Z_{prep}\right)\right),$$

определяемую как сложность решения задачи Z_{prep} (42) с начальной информацией $\Im_0^{prep} = \left(\left\{ g_x(k) \right\}_{k=0}^{K_x-1}, \left(N + K_x, \Im_{(x)} \right) \right)$ выбранным алгоритмом A_{prep}.

Второй шаг алгоритма модели, на котором производится вычисление S сверток вида (39), имеет сложность

$$U_{add}\left(A^{CR}_{Step\_2}\right) = \sum_{s=0}^{S-1} U_{add}\left(A_s\left(Z_s\right)\right),$$
$$U_{mul}\left(A^{CR}_{Step\_2}\right) = \sum_{s=0}^{S-1} U_{mul}\left(A_s\left(Z_s\right)\right),$$

определяемую как сложность решения S задач $\{Z_s\}_{s=0}^{S-1}$ с начальными информациями $\Im_0^s = \left(\left\{ \widetilde{h}_s(m) \right\}_{m \in D_s}, \left(N + K_x, \Im_{(\widetilde{x})} \right) \right)$ выбранными алгоритмами $A_s \left(s = \overline{0, S-1} \right)$.

Третий шаг алгоритма, на котором производится вычисление выражения (40), очевидно, имеет сложность

$$U_{add}\left(A^{CR}_{Step\_3}\right) = S-1, \quad U_{mul}\left(A^{CR}_{Step\_3}\right) = 0,$$

поскольку требует только операций сложения.

Наконец, четвертый шаг, на котором производится рекурсивное вычисление результата решения задачи Z на основании выражения (34), имеет сложность

$$U_{add}\left(A^{CR}_{Step\_4}\right) = U_{mul}\left(A^{CR}_{Step\_4}\right) = K_h + K_x - 2.$$

Суммирую соответствующие выражения сложности, получаем итоговое выражение для сложности произвольного алгоритма модели CR

$$\begin{aligned} U_{add}\left(A^{CR}\right) &= U_{add}\left(A_{prep}\left(Z_{prep}\right)\right) + \\ &+ \sum_{s=0}^{S-1} U_{add}\left(A_s\left(Z_s\right)\right) + K_h + K_x - 2 + (S-1), \\ U_{mul}\left(A^{CR}\right) &= U_{mul}\left(A_{prep}\left(Z_{prep}\right)\right) + \\ &+ \sum_{s=0}^{S-1} U_{mul}\left(A_s\left(Z_s\right)\right) + K_h + K_x - 2. \end{aligned} \tag{44}$$

С учетом относительной сложности реализаций операций сложения и умножения, получаем выражение:

$$\begin{aligned} U\left(A^{CR}\right) &= \left[\begin{array}{l} \xi_{add} U_{add}\left(A_{prep}\left(Z_{prep}\right)\right) + \\ + \xi_{mul} U_{mul}\left(A_{prep}\left(Z_{prep}\right)\right) \end{array} \right] + \\ &+ \sum_{s=0}^{S-1} \left[\xi_{add} U_{add}\left(A_s\left(Z_s\right)\right) + \xi_{mul} U_{mul}\left(A_s\left(Z_s\right)\right) \right] + \\ &+ \left(K_h + K_x - 2\right)\left(\xi_{add} + \xi_{mul}\right) + \xi_{add}(S-1). \end{aligned}$$

Поскольку в первых двух строках этой формулы расположены выражения для сложностей алгоритмов A_{prep} и $\{A_s\}_{s=0}^{S-1}$, получаем окончательно:

$$\begin{aligned} U\left(A^{CR}\right) &= U\left(A_{prep}\left(Z_{prep}\right)\right) + \\ &+ \left[\sum_{s=0}^{S-1} U\left(A_s\left(Z_s\right)\right) + \xi_{add}(S-1) \right] + \\ &+ \left(K_h + K_x - 2\right) \end{aligned} \tag{45}$$

Очевидно, выражения для сложности (44)- (45) определены также с точностью до числовых параметров модели и параметров-алгоритмов A_{prep} и $\{A_s\}_{s=0}^{S-1}$. Только после указания всех этих параметров и алгоритм модели CR, и выражения для его сложности приобретают окончательный вид.

Пусть, далее, задано некоторое множество $\{A\}$ алгоритмов решения различных задач из множества $\{Z\}$ задач вычисления сверток вида (1).

Определение 34. *Замыканием по модели CR порядка* (K_h, K_x, S) *множества алгоритмов* $\{A\}$ *на задаче* Z (далее – просто замыканием $CR(K_h, K_x, S)$) называется множество алгоритмов модели CR, обозначаемое $[\{A\}]_{CR(K_h,K_x,S)}$, с заданными K_h, K_x, S и допустимыми значениями других числовых параметров и алгоритмами из $\{A\}$ на первом и втором шагах модели CR, для которых задачи Z_{prep} (42) и Z_s $\left(s=\overline{0,S-1}\right)$ из (43) попадают в область определения соответствующих алгоритмов:

$$A_{prep} \in \{A(Z_{prep})\} \subseteq \{A\},$$
$$A_s \in \{A(Z_s)\} \subseteq \{A\} \quad \left(s=\overline{0,S-1}\right).$$

Определение 35. *Замыканием по модели CR множества алгоритмов* $\{A\}$ *на задаче* Z (далее – просто замыканием) называется множество алгоритмов модели CR, обозначаемое $[\{A\}]$, следующего вида:

$$[\{A\}] = \bigcup_{\substack{K_h=\overline{1,M-1} \\ K_x=\overline{1,N-1} \\ S=1,2,\ldots}} [\{A\}]_{CR(K_h,K_x,S)}.$$

Предложение 13. Для алгоритмов из замыкания $[\{A\}]_{CR(1,1,1)}$ область допустимых покрытий удовлетворяет условию $D_0 = D = [0, M-1]$.

Доказательство*:* следует непосредственно из определения допустимого покрытия (34)-(35) и условия (2) на ИХ $\{h(n)\}_{n=0}^{M-1}$: $h(0) \neq 0, \quad h(M-1) \neq 0$.

■

Предложение 14.

Пусть $\{\widetilde{A}\} \subseteq \{A\}$. Тогда $[\{\widetilde{A}\}] \subseteq [\{A\}]$.

Доказательство*:* производится очевидным образом, путем указания для каждого конкретного алгоритма из замыкания $[\{\widetilde{A}\}]$ тождественного ему алгоритма из $[\{A\}]$ с теми же числовыми параметрами и теми же алгоритмами из $\{\widetilde{A}\} \subseteq \{A\}$ для расчета сверток.

■

Предложение 15. Пусть дано множество алгоритмов $\{A\}$ и для конкретной задачи Z построено замыкание этого множества по модели CR $[\{A\}]_{CR(1,1,1)}$. Тогда:

$$\forall A \in \{A\} \quad \exists A^{CR} \in [\{A\}]_{CR(1,1,1)}: \quad A^{CR} \cong A \quad \wedge$$
$$\forall A^{CR} \in [\{A\}]_{CR(1,1,1)} \quad \exists A \in \{A\}: \quad A \cong A^{CR} \quad \wedge$$
$$\min_{A \in \{A(Z)\} \subseteq \{A\}} U(A(Z)) = \min_{A^{CR} \in [\{A\}]} U(A^{CR}(Z))$$

Доказательство*:* производится очевидным образом, путем указания для каждого конкретного алгоритма из $\{A\}$ подобного ему алгоритма из $[\{A\}]_{CR(1,1,1)}$. И наоборот.

■

Следствие 4. (Предложения 15) Между множествами $\{A\}$ и $[\{A\}]_{CR(1,1,1)}$ существует взаимнооднозначное соответствие.

3.2. Замыкание по модели CR основных классов алгоритмов. Подклассы алгоритмов модели CR

Замыкание по модели CR может быть построено для произвольного множества алгоритмов. Однако, учитывая, что для решения задачи вычисления свертки существуют три основных класса алгоритмов вычисления свертки (п.1.5), особый интерес представляют замыкания множеств алгоритмов именно этих классов. Следует особо отметить, что алгоритмы из замыкания в общем случае отличаются от алгоритмов основных классов. Таким образом, алгоритмы замыкания формируют другие классы алгоритмов вычисления свертки, которые могут не попадать ни в один из основных классов. Для удобства будем именовать эти новые классы как *подклассы алгоритмов модели CR*. Все множество подклассов алгоритмов модели CR сведено в таблицу 1.

Следует также отметить, что некоторые подклассы алгоритмов можно считать известными в ЦОС, поскольку в литературе существуют эвристически построенные одиночные алгоритмы или множества алгоритмов с вычислительной структурой, подобной структуре алгоритмов из некоторого подкласса. Однако другие подклассы алгоритмов являются действительно новыми и в литературе ранее не рассматривались. Ниже даны комментарии относительно того, какие подклассы алгоритмов могут считаться относительно известными (выделены в таблице серым цветом), а какие – действительно новыми.

Разбор подклассов алгоритмов начнем с первой колонки таблицы 1.3.1, то есть с алгоритмов из замыкания порядка (K_h, K_x, S): $K_h = K_x = S = 1$. Алгоритмы этих подклассов, в соответствие с предложением 15 и следствием 4, подобны алгоритмам из основных подклассов алгоритмов вычисления свертки. То есть являются известными.

Таблица 1. *Подклассы алгоритмов модели CR* $\left[\{A\}\right]_{CR(K_h,K_x,S)}$

(K_h, K_h, S) / $\{A\}$	$K_h = K_x = S = 1$	$K_h = K_x = 1, \quad S > 1$	$K_h = S = 1, \quad K_x > 1$	$K_x = 1, K_h > 1, S >$	$K_h > 1, K_x > 1, S > 1$
алгоритмы прямого вычисления свертки: $\{A_{DC}\}$	алгоритмы прямого вычисления свертки: $\{A_{DC}\}$	алгоритмы прямого вычисления свертки с пространственной декомпозицией ИХ	алгоритмы прямого вычисления свертки с пред- и постобработкой данных $\{A_{RF}\}$	алгоритмы параллельно-рекурсивного вычисления свертки	алгоритмы рекурсивного вычисления свертки с предобработкой
алгоритмы вычисления свертки на основе ДОП: $\{A_{FC}\}$	алгоритмы вычисления свертки на основе ДОП: $\{A_{FC}\}$	алгоритмы вычисления свертки на основе ДОП с пространственной декомпозицией ИХ	алгоритмы вычисления свертки на основе ДОП с пред- и постобработкой данных	алгоритмы спектрально-рекурсивного вычисления свертки	алгоритмы спектрально-рекурсивного вычисления свертки с предобработкой
алгоритмы рекурсивного вычисления свертки: $\{A_{RF}\}$	алгоритмы рекурсивного вычисления свертки: $\{A_{RF}\}$	алгоритмы параллельно-рекурсивного вычисления свертки	алгоритм последовательного рекурсивного вычисления свертки	алгоритмы итеративно-рекурсивного вычисления свертки	алгоритмы итеративно-рекурсивного вычисления свертки с предобработкой
основные классы алгоритмов вычисления свертки постоянной сложности: $\{A_{DC}\} \cup \{A_{FC}\}$	основные классы алгоритмов вычисления свертки постоянной сложности: $\{A_{DC}\} \cup \{A_{FC}\}$	основные классы алгоритмов вычисления свертки постоянной сложности с пространственной декомпозицией ИХ	основные классы алгоритмов вычисления свертки постоянной сложности с пред- и постобработкой данных	алгоритмы обобщенно-спектрально-рекурсивного вычисления свертки	алгоритмы обобщенно-спектрально-рекурсивного вычисления свертки с предобработкой
основные классы алгоритмов вычисления свертки: $\{A_{DC}\} \cup \{A_{FC}\} \cup \{A_{FC}\}$	основные классы алгоритмов вычисления свертки: $\{A_{DC}\} \cup \{A_{FC}\} \cup \{A_{FC}\}$	параллельные алгоритмы вычисления свертки	основные классы алгоритмов вычисления свертки с пред- и постобработкой данных	обобщенные рекурсивные алгоритмы вычисления свертки	обобщенные рекурсивные алгоритмы вычисления свертки с предобработкой

Алгоритмы из замыкания порядка $(K_h, K_x, S): K_h = K_x = 1,\ S > 1$ отличаются от алгоритмов модели $CR(1,1,1)$ только тем, что ИХ $\{h(n)\}$ представлена в виде разложения (36).

Подобное разложение может производиться по одной из двух причин. Во-первых, по причине наличия большого количества «нулевых» отсчетов в ИХ. Такая модификация алгоритмов свертки известна и используется в работах по ЦОС [9, 17, 31, 39, 62] для снижения сложности, она не требует каких-либо более серьезных обоснований. Второй причиной, которая служит объяснением такого способа вычисления свертки, является возможность распараллеливания процесса вычислений. Здесь особо хочется отметить подкласс алгоритмов параллельно-рекурсивного вычисления свертки. В этих алгоритмах ИХ представляется в виде набора рекурсивно реализуемых звеньев.

Публикации по тематике построения и использования таких алгоритмов имеются у различных авторов, включая автора настоящей работы [24-26, 33, 38, 40, 41, 43-47, 52, 55-62]: Л.П. Ярославский, В.В. Сергеев, Н.И. Глумов, В.В. Мясников, А.В. Чернов, M. Hatamian, M.F. Zakaria и др.. Таким образом, поскольку и первая, и вторая причины и соответствующие им алгоритмы рассматривались в литературе, считаем эти подклассы алгоритмов вычисления свертки также известными.

Алгоритмы из замыкания порядка (K_h, K_x, S), где $K_h = S = 1,\ \ K_x > 1$, встречаются в литературе значительно реже, чем их рассмотренные предшественники. А именно, некоторые частные эвристические алгоритмы, которые соответствуют подклассам, выделенным в соответствующем столбце таблицы серым цветом, и именование которых включает оборот «с пред- и постобработкой данных», были рассмотрены в нескольких работах Ярославского Л.П. и Кобера В.И [9,17,53,54,62]. Алгоритмы указанных авторов использовали предварительное дифференцирование входных сигналов для установки в «0» большого числа значений обрабатываемого сигнала. Такая предобработка позволяла использовать ускоренные алгоритмы БПФ, которые были адаптированы к сигналам с редкими «полезными» значениями, рассмотренными, например, в работе [31] В.М. Чернова и П.В. Раудина. Автору данной работы не известны работы других авторов с примерами использования такого же подхода для рекурсивных фильтров, поэтому два подкласса алгоритмов в рассматриваемой колонке таблицы остались не выделенными.

Алгоритмы из замыкания порядков (K_h, K_x, S), где $K_x = 1, K_h > 1, S > 1$ или $K_h > 1, K_x > 1, S > 1$, соответствующие последним двум столбцам таблицы 1.3.1, в явном виде в литературе не встречались. Из этих алгоритмов хотелось бы особо отметить подкласс алгоритмов спектрально- рекурсивного вычисления свертки. Указанное название не только отражает способ вычисления свертки в алгоритмах этого подкласса, но также специально ассоциировано с названием «спектрально-рекуррентный», которые был введен для алгоритма фильтрации двумерного сигнала разделимой ИХ В.А. Сойфером и А.Г. Храмовым в работе [33], изложенной также в монографии [8].

В заключение настоящего раздела следует отметить, что, несмотря на конструктивный характер введенного понятия «замыкания алгоритмов», приведенная таблица используется в настоящей работе в основном для классификационных целей, давая определения для подклассов алгоритмов из соответствующих замыканий.

3.3. Основные теоремы об эффективности алгоритма, индуцированного априорной информацией задачи Z

Пусть решается задача Z с априорной информацией $\Im_0$:

$$\begin{aligned} Z &= Z\left(\Im_0, \{x(n)\}_{n=0}^{N-1}\right), \\ \Im_0 &= \left(\{h(n)\}_{n=0}^{M-1}, (N, \Im_{(x)})\right). \end{aligned} \tag{46}$$

Определение 36 Алгоритм $A^{\Im}(Z) \in [\{A\}]$ называется *алгоритмом, индуцированным априорной информацией* $\Im_0$ задачи (46) Z (далее – *индуцированным алгоритмом*), если алгоритм $A^{\Im}(Z)$ является наилучшим алгоритмом для задачи Z в замыкании $[\{A\}]$ по модели CR и, кроме того, в замыкании $[\{A\}]$ нет другого алгоритма меньшего порядка со сложностью, равной сложности алгоритма $A^{\Im}(Z)$:

$$\begin{aligned} & U\left(A^{\Im}(Z)\right) = \min_{A(Z)\in[\{A\}]} U(A(Z)) \\ & \wedge \quad A^{\Im}(Z) \in [\{A\}]_{CR(K_h,K_x,S)} \quad \wedge \\ & \wedge \left(\begin{array}{l} \forall \left(\widetilde{K}_h, \widetilde{K}_x, \widetilde{S}\right) \prec (K_h, K_x, S) \\ \min\limits_{A(Z)\in[\{A\}]_{CR}\left(\widetilde{K}_h,\widetilde{K}_x,\widetilde{S}\right)} U(A(Z)) > U\left(A^{\Im}(Z)\right) \end{array} \right) \end{aligned} \tag{47}$$

Теорема 2 Индуцированный алгоритм $A^{\Im} \in [\{A\}]_{CR(K_h,K_x,S)} \subseteq [\{A\}]$ является строго эффективным над множеством $\{A\}$ для задачи Z тогда и только тогда, когда $(K_h > 1) \vee (K_h > 1) \vee (S > 1)$.

Доказательство:

Достаточность: доказывается отдельно по каждому из неравенств в условии теоремы.

Вариант 1. Пусть $A^{\Im}(Z) \in [\{A\}]$ - индуцированный алгоритм (аргумент задачи Z ниже в доказательстве опускаем для краткости изложения), и выполняются неравенства: $K_h > 1$ и $K_x \geq 1, S \geq 1$. Тогда, по определению индуцированного алгоритма:

$$U\left(A^{\Im}\right)=\min_{A\in[\{A\}]}U(A(Z))=\min_{\substack{A\in[\{A\}]_{CR\left(\widetilde{K}_h,\widetilde{K}_x,\widetilde{S}\right)}\\ \widetilde{K}_h=1,2,...\\ \widetilde{K}_x=1,2,...\\ \widetilde{S}=1,2,...}}U(A)=$$

$$=\min_{\substack{A\in[\{A\}]_{CR\left(K_h,\widetilde{K}_x,\widetilde{S}\right)}\\ \widetilde{K}_x=1,2,...\\ \widetilde{S}=1,2,...}}U(A).$$

В силу минимальности вектора $\left(K_h,K_x,S\right)$ для индуцированного алгоритма, справедливо неравенство:

$$\forall\widetilde{K}_x,\widetilde{S}\quad U\left(A^{\Im}\right)=\min_{\substack{A\in[\{A\}]_{CR\left(K_h,\widetilde{K}_x,\widetilde{S}\right)}\\ \widetilde{K}_x=1,2,...\\ \widetilde{S}=1,2,...}}U(A)<$$

$$<\min_{\substack{A\in[\{A\}]_{CR\left(1,\widetilde{K}_x,\widetilde{S}\right)}\\ \widetilde{K}_x=1,2,...\\ \widetilde{S}=1,2,...}}U(A)\le\min_{A\in[\{A\}]_{CR(1,1,1)}}U(A).$$

В силу предложения 13, $D_0=D$ и справедливо:

$$\min_{A\in[\{A\}]_{CR(1,1,1)}}U(A(Z))=$$
$$=\min_{A\in\{A(Z)\}}U(A(Z))=U\left(A^*\right),\qquad(48)$$

где A^* - наилучший алгоритм в множестве алгоритмов решения задачи Z: $A^*\in\{A(Z)\}\subseteq\{A\}$.

Окончательно $U\left(A^J\right)<U\left(A^*\right)$, и по определению 19 алгоритм $A^{\Im}\in[\{A\}]$ - строго эффективный над множеством $\{A(Z)\}\subseteq\{A\}$ для задачи Z.

Вариант 2. Пусть $A^{\Im}\in[\{A\}]$ - индуцированный алгоритм, и выполняются неравенства: $K_h=1$ и $K_x>1,S\ge1$.

Тогда по определению индуцированного алгоритма:

$$U\left(A^{\Im}\right)=\min_{A\in[\{A\}]}U(A)\ =\min_{\substack{A\in[\{A\}]_{CR\left(\widetilde{K}_h,\widetilde{K}_x,\widetilde{S}\right)}\\ \widetilde{K}_h=1,2,...\\ \widetilde{K}_x=1,2,...\\ \widetilde{S}=1,2,...}}U(A)=$$

$$=\min_{\substack{A\in[\{A\}]_{CR\left(1,K_x,\widetilde{S}\right)}\\ \widetilde{S}=1,2,...}}U(A).$$

В силу минимальности вектора $\left(K_h,K_x,S\right)$ для индуцированного алгоритма справедливо неравенство:

$$\forall\tilde{S}\ U\left(A^{\Im}\right)=\min_{\substack{A\in[\{A\}]_{CR(1,K_x,\tilde{S})}\\ \tilde{S}=1,2,...}}U(A)<$$

$$<\min_{\substack{A\in[\{A\}]_{CR(1,1,\tilde{S})}\\ \tilde{S}=1,2,...}}U(A)\le\min_{A\in[\{A\}]_{CR(1,1,1)}}U(A).$$

Используя Предложение 15, получаем окончательно требуемое неравенство $U\left(A^{\Im}\right)<U\left(A^*\right)$ для строго эффективного алгоритма над множеством $\{A(Z)\}\subseteq\{A\}$ для задачи Z.

Вариант 3. Пусть $A^{\Im}\in[\{A\}]$ - индуцированный алгоритм, и выполняются неравенства: $K_h=K_x=1$ и $S>1$. Доказательство полностью аналогично вариантам 1 и 2.

Итак, достаточность доказана.

Необходимость: докажем от противного.

Пусть $A^{\Im}(Z)\in[\{A\}]$ - индуцированный алгоритм и он строго эффективен, то есть для него выполняется неравенство

$$U\left(A^J(Z)\right)<U\left(A^*(Z)\right),\ A^*(Z)\in\{A\}.\qquad(49)$$

Допустим, что при этих условиях соотношения ($K_h>1\vee K_x>1\vee S>1$) не выполняются, то есть выполняется $\left(K_h=1\wedge K_x=1\wedge S=1\right)$. Тогда для сложности индуцированного алгоритма справедливо равенство

$$U\left(A^{\Im}\right)=\min_{A\in[\{A\}]_{CR(1,1,1)}}U(A).$$

С учетом Предложения 15, гарантирующего существование для алгоритма из замыкания по модели $CR(1,1,1)$ алгоритма из множества $\{A\}$ с той же сложностью, получаем:

$$U\left(A^{\Im}\right)=\min_{A^{CR}\in[\{A(Z)\}]}U\left(A^{CR}(Z)\right)=$$
$$=\min_{A\in\{A(Z)\}}U(A(Z))=U\left(A^*\right)\qquad(50)$$

Неравенства (49) и (50) несовместны, следовательно, посылка $\left(K_h=1\right)\wedge\left(K_x=1\right)\wedge\left(S=1\right)$ неверна, а верно обратное неравенство.

Что и требовалось доказать.

■

Приводимое ниже следствие данной теоремы выделяет достаточные условия того, что индуцированный алгоритм является строго эффективным, а формулировка этого следствия удобна для проверки индуцированного алгоритма на то, что он строго эффективен.

Следствие 5. Если $A^{\Im}(Z)\in[\{A\}]$ - индуцированный алгоритм для задачи Z и выполняется одно из следующих условий:

- $K_h>1$,
- $K_x>1$,
- $S>1$,

тогда $A^{\Im}(Z)$ - строго эффективный алгоритм над множеством $\{A\}$ для задачи Z.

Следующая теорема обобщает формулировку Теоремы 2, указывая на (не строгую) эффективность любого индуцированного алгоритма.

Теорема 3. Индуцированный алгоритм $A^{\Im}(Z)\in[\{A\}]$ для задачи Z является эффективным алгоритмом над множеством алгоритмов $\{A\}$ для задачи Z.

Доказательство:

Для доказательства достаточно показать, что для индуцированного алгоритма выполняется условие эффективного алгоритма: $U\left(A^{\Im}\right)\le U\left(A^{*}\right)$. Здесь A^{*} - наилучший алгоритм множества алгоритмов решения задачи Z: $A^{*}\in\{A(Z)\}\subseteq\{A\}$, а множество $\{A\}$ - есть множество алгоритмов, над котором строится замыкание в модели CR и, соответственно, выбор индуцированного алгоритма. Для этого рассмотрим две взаимоисключающие ситуации.

Ситуация 1: $A^{\Im}\in\left[\{A\}\right]_{CR(K_h,K_x,S)}$ при $K_h>1$ или $K_x>1$ или $S>1$.

Условия этой ситуации удовлетворяют условиям Теоремы 2. Следовательно, алгоритм $A^{\Im}\in\left[\{A\}\right]$ является строго эффективным и, следовательно, эффективным алгоритмом для задачи Z.

Ситуация 2: $A^{\Im}\in\left[\{A\}\right]_{CR(K_h,K_x,S)}$ при $K_h=K_x=S=1$.

В соответствие с определением индуцированного алгоритма $A^{\Im}$, он является наилучшим алгоритмом в замыкании $A^{\Im}\in\left[\{A\}\right]_{CR(1,1,1)}$. Для замыкания этого порядка выражение сложности (35) имеют вид:

$$U\left(A^{\Im}(Z)\right)=\min_{A^{CR}\in\left[\{A\}\right]_{CR(1,1,1)}}U\left(A^{CR}(Z)\right)\quad.$$

С учетом Предложения 15, гарантирующего существование для алгоритма из замыкания по модели $CR(1,1,1)$ алгоритма из множества $\{A\}$ с той же сложностью, получаем продолжение равенства:

$$U\left(A^{\Im}(Z)\right)=\min_{A_0\in\{A(Z_0)\}}U\left(A_0(Z_0)\right)=$$
$$=\min_{A\in\{A(Z)\}}U\left(A(Z)\right)=U\left(A^{*}(Z)\right)$$

Таким образом, и ситуация 1, и ситуация 2 удовлетворяют неравенству

$$U\left(A^{\Im}(Z)\right)\le U\left(A^{*}(Z)\right).$$

Что и требовалось доказать.

■

3.4. Об эффективности индуцированного алгоритма из замыкания по модели CR приведенных алгоритмов постоянной сложности

Следующие леммы и теорема касаются частного случая, когда работа производится с приведенными алгоритмами постоянной сложности или одним из них.

Лемма 6. Пусть КИХ $\{h(n)\}_{n=0}^{M-1}$ в задаче Z (46) удовлетворяет условию: $\forall\, m\in[0,M-1]\quad h(n)\ne 0$. Пусть $A^{\Im}(A)\in\left[\{A\}\right]$ - индуцированный алгоритм задачи Z над множеством $\left\{\breve{A}\right\}$, которое состоит из одного приведенного алгоритма постоянной сложности $\breve{A}$. Тогда $\forall\, S>1\quad A^{\Im}(Z)\notin\left[\{A\}\right]_{CR(1,1,S)}$.

Доказательство.

Пусть $A^{\Im}(Z)\in\left[\{A\}\right]$ - индуцированный алгоритм задачи Z над множеством $\left\{\breve{A}\right\}$, где $\breve{A}$ - некоторый приведенный алгоритм постоянной сложности. Допустим, что $K_h=K_x=1,\quad S>1$. Тогда сложность (45) для индуцированного алгоритма, с учетом выражения (47), имеет вид:

$$U\left(A^{\Im}(Z)\right)=\min_{A\in\left[\left\{\breve{A}\right\}\right]_{CR(1,1,S)}}U\left(A(Z)\right)=$$
$$=\min_{\substack{\{D_s\}_{s=0}^{S-1},\\ A_s=\breve{A}}}\left(\sum_{s=0}^{S-1}U\left(A_s(Z_s)\right)+\xi_{add}(S-1)\right)=$$
$$\min_{\{D_s\}_{s=0}^{S-1}}\left(\sum_{s=0}^{S-1}u_{\breve{A}}\left(|D_s|,N\right)+\xi_{add}(S-1)\right).\qquad(51)$$

Поскольку по условиям теоремы $\{h(n)\}_{n=0}^{M-1}$ удовлетворяет ограничению

$$\forall\, m\in[0,M-1]\quad h(n)\ne 0,$$

то для допустимого покрытия $\{D_s\}_{s=0}^{S-1}$ области определения ИХ $D=[0,M-1]$, по условиям его определения (34)-(35), справедливо: $\bigcup_{s=0}^{S-1}D_s=D$. Тогда, учитывая что $\breve{A}$ - приведенный алгоритм постоянной сложности, на основании Леммы 4 получим продолжение выражения (51):

$$\min_{\{D_s\}_{s=0}^{S-1}}\left(\sum_{s=0}^{S-1}u_{\breve{A}}\left(|D_s|,N\right)+\xi_{add}(S-1)\right)\ge$$
$$\ge u_{\breve{A}}\left(|D|,N\right)=u_{\breve{A}}(M,N)$$

и итоговое неравенство:

$$U\left(A^{\Im}(Z)\right)\ge u_{\breve{A}}(M,N).\qquad(52)$$

С другой стороны, по Теореме 3 индуцированный алгоритм $A^{\Im}(Z)\in\left[\left\{\breve{A}\right\}\right]$ является эффективным алгоритмом над множеством $\left\{\breve{A}\right\}$ для задачи Z, следовательно:

$$U\left(A^{\Im}(Z)\right)\le u_{\breve{A}}(M,N).\qquad(53)$$

Объединяя (52) и (53), получим окончательно равенство:

$$U\left(A^{\Im}(Z)\right)=u_{\breve{A}}(M,N).\qquad(54)$$

В то же время, для наилучшего алгоритма модели $CR(1,1,1)$ справедливо:

$$\min_{A\in\left[\left\{\breve{A}\right\}\right]_{CR(1,1,1)}}U\left(A(Z)\right)=U\left(\breve{A}(Z)\right)=u_{\breve{A}}(M,N).\qquad(55)$$

Полученные равенства (54) и (55), с учетом минимальности порядка индуцированного алгоритма, означают, что $A^{\Im}(Z)\in\left[\left\{\breve{A}\right\}\right]_{CR(1,1,1)}$. Однако, по предположению $K_h=K_x=1,\ S>1$. Следовательно, предположение неверно, и $S=1$. Что и требовалось доказать.

■

Теорема 4. Пусть КИХ $\{h(n)\}_{n=0}^{M-1}$ в задаче Z (46) удовлетворяет условию: $\forall m\in[0,M-1]\ \ h(n)\neq 0$. Индуцированный алгоритм $A^{\Im}(Z)\in\left[\{A\}\right]_{CR(K_h,K_x,S)}\subseteq\left[\{A\}\right]$ задачи Z над множеством $\left\{\breve{A}\right\}$, состоящего из одного приведенного алгоритма постоянной сложности $\breve{A}$, является строго эффективным тогда и только тогда, когда $(K_h>1)\vee(K_x>1)$.

Доказательство:

Достаточность: Является следствием Теоремы 2 и ее Следствия 5.

Необходимость. Пусть $A^{\Im}(Z)\in\left[\{A\}\right]$ - индуцированный алгоритм задачи Z над множеством $\left\{\breve{A}\right\}$, где $\breve{A}$ - некоторый приведенный алгоритм постоянной сложности. Покажем, что если выполняется $A^{\Im}(Z)\in\left[\left\{\breve{A}\right\}\right]_{CR(1,1,S)}$ (для любого *S*), тогда индуцированный алгоритм не является строго эффективным алгоритмом. Для этого рассмотрим две ситуации: *S*=1 и *S*>1.

Ситуация 1 (*S*=1). Индуцированный алгоритм $A^{\Im}(Z)\in\left[\{A\}\right]$ не является строго эффективным в соответствие с более общей Теоремой 2.

Ситуация 2 (*S*>1). Такая ситуация для индуцированного алгоритма над множеством из одного приведенного алгоритма постоянной сложности, в соответствии с только что доказанной Леммой 6, не возможна. Как следует из доказательства указанной Леммы: *S*=1 и, следовательно, индуцированный алгоритм $A^{\Im}(Z)\in\left[\{A\}\right]$ не является строго эффективным.

Что и требовалось доказать.

■

Лемма 7. Пусть решается задача Z (46). Пусть $\{A\}$ - множество алгоритмов постоянной сложности, $\breve{A}^{\oplus}$ - приведенный компетентный алгоритм над множеством алгоритмов $\{A\}$. Пусть $A^{CR}\in\left[\{A\}\right]_{CR(K_h,K_x,S)}$ - некоторый алгоритм из замыкания по модели CR множества алгоритмов $\{A\}$ на задаче Z с набором допустимых числовых параметров, $A_{\oplus}^{CR}\in\left[\left\{\breve{A}^{\oplus}\right\}\right]_{CR(K_h,K_x,S)}$ - алгоритм из замыкания по модели CR множества (из одного элемента) алгоритмов $\left\{\breve{A}^{\oplus}\right\}$ на задаче Z с тем же набором допустимых числовых параметров. Тогда справедливо:

$$U\left(A_{\oplus}^{CR}(Z)\right)\le U\left(A^{CR}(Z)\right). \tag{56}$$

Доказательство:

Пусть $A^{CR}\in\left[\{A\}\right]_{CR(K_h,K_x,S)}$ - некоторый алгоритм из замыкания по модели CR множества алгоритмов $\{A\}$ на задаче Z с набором допустимых числовых параметров. Тогда его сложность определяется выражением (45):

$$U\left(A^{CR}(Z)\right)=U\left(A_{prep}\left(Z_{prep}\right)\right)+\left[\sum_{s=0}^{S-1}U\left(A_s(Z_s)\right)+\xi_{add}(S-1)\right]+(K_h+K_x-2)$$

где алгоритмы A_{prep} и $\{A_s\}_{s=0}^{S-1}$ выбираются для решения соответствующих задач Z_{prep} и $Z_s\ \left(s=\overline{0,S-1}\right)$ из множества алгоритмов $\{A\}$: $A_{prep}\in\{A\},\ A_s\in\{A\}\ \ \left(s=\overline{0,S-1}\right)$. В соответствие со свойством компетентного алгоритма

$$\forall\, Z\in\bigcup_{A\in\{A\}}\aleph_A\quad\forall\, A(Z)\in\{A(Z)\}\cup\left\{A^{\oplus}\right\}\quad U\left(A^{\oplus}(Z)\right)\le U(A(Z)).$$

Тогда

$$\begin{aligned}U\left(A^{CR}(Z)\right)&=U\left(A_{prep}\left(Z_{prep}\right)\right)+\left[\sum_{s=0}^{S-1}U\left(A_s(Z_s)\right)+\xi_{add}(S-1)\right]+(K_h+K_x-2)\ \ge\\&\ge U\left(A^{\oplus}\left(Z_{prep}\right)\right)+\left[\sum_{s=0}^{S-1}U\left(A^{\oplus}(Z_s)\right)+\xi_{add}(S-1)\right]+(K_h+K_x-2).\end{aligned}\tag{57}$$

Кроме того, если $A\underset{u(M,N)}{\to}\breve{A}$, тогда справедливо соотношение (15') в виде:

$$\forall\, Z\in\aleph_A\ \ U\left(\breve{A}(Z)\right)\le U(A(Z)).$$

Следовательно, выражение (57) можно продолжить:

$$\begin{aligned}U\left(A^{CR}(Z)\right)&\ge U\left(A^{\oplus}\left(Z_{prep}\right)\right)+\left[\sum_{s=0}^{S-1}U\left(A^{\oplus}(Z_s)\right)+\xi_{add}(S-1)\right]+(K_h+K_x-2)(\xi_{add}+\xi_{mul})\ge\\&\ge U\left(\breve{A}^{\oplus}\left(Z_{prep}\right)\right)+\left[\sum_{s=0}^{S-1}U\left(\breve{A}^{\oplus}(Z_s)\right)+\xi_{add}(S-1)\right]+(K_h+K_x-2)=U\left(A_{\oplus}^{CR}(Z)\right).\end{aligned}$$

Откуда и получаем искомое неравенство при условии, что ни порядок модели (K_h, K_x, S), ни задачи $A_{prep} \in \{A\}$, $A_s \in \{A\}$ $\left(s = \overline{0, S-1}\right)$ не меняются. Последнее происходит, если все числовые параметры (порядок модели, границы допустимого покрытия, отсчеты вспомогательных сигналов $\{g_h(k)\}_{k=0}^{K_h-1}$ и $\{g_x(k)\}_{k=0}^{K_x-1}$ и т.д.) не изменились.

■

Теорема 5. Пусть решается задача Z (46). Пусть $\{A\}$ - множество алгоритмов постоянной сложности, $\breve{A}^{\oplus}$ - приведенный компетентный алгоритм над множеством алгоритмов $\{A\}$. Пусть $A^{\Im}(Z) \in [\{A\}]$, $A_{\oplus}^{\Im}(Z) \in \left[\left\{\breve{A}^{\oplus}\right\}\right]$ - индуцированные алгоритмы задачи Z над множествами алгоритмов, соответственно, $\{A\}$ и $\left\{\breve{A}^{\oplus}\right\}$. Тогда справедливо:

$$U\left(A_{\oplus}^{\Im}(Z)\right) \le U\left(A^{\Im}(Z)\right). \quad (58)$$

Доказательство:

Пусть индуцированный алгоритм $A^{\Im}(Z) \in [\{A\}]$ принадлежит замыканию порядка (K_h, K_x, S), а также имеет некоторый набор допустимых числовых параметров. Рассмотрим алгоритм $A_{\oplus}^{CR} \in \left[\left\{\breve{A}^{\oplus}\right\}\right]_{CR(K_h,K_x,S)}$ из замыкания по модели CR порядка (K_h, K_x, S) множества алгоритмов $\left\{\breve{A}^{\oplus}\right\}$ (с одним элементом $\breve{A}^{\oplus}$) с теми же числовыми параметрами, что и у алгоритма $A^{\Im}(Z) \in [\{A\}]$. Тогда, в соответствие с только что доказанной Леммой 7 выполняется неравенство:

$$U\left(A_{\oplus}^{CR}(Z)\right) \le U\left(A^{\Im}(Z)\right). \quad (59)$$

Кроме того, поскольку $A_{\oplus}^{\Im}(Z) \in \left[\left\{\breve{A}^{\oplus}\right\}\right]$ - индуцированный алгоритм задачи Z, он имеет минимальную сложность (47) среди всех алгоритмов из замыкания. В частности:

$$U\left(A_{\oplus}^{\Im}(Z)\right) = \min_{A(Z) \in \left[\left\{\breve{A}^{\oplus}\right\}\right]} U\left(A(Z)\right) \le U\left(A_{\oplus}^{CR}(Z)\right). \quad (60)$$

Объединяя неравенства (59) и (60) получим окончательно:

$$U\left(A_{\oplus}^{\Im}(Z)\right) \le U\left(A^{\Im}(Z)\right),$$

что и требовалось доказать.

■

Последняя теорема фиксирует очень важный факт, который формулирует приведенные ниже следствия.

Следствие 6. (Теоремы 5) Для любой задачи Z индуцированный алгоритм над приведенным компетентным алгоритмом, построенным над множеством $\{A\}$ алгоритмов постоянной сложности, не хуже, чем индуцированный алгоритм, над тем же множеством алгоритмов.

Теперь можно сформулировать конструктивное правило, которое следует использовать при построении эффективного алгоритма для задачи Z над множеством алгоритмов постоянной сложности.

Следствие 7 (Теоремы 5) Для построения алгоритма, эффективного для задачи Z над множеством $\{A\}$ алгоритмов постоянной сложности, достаточно построить индуцированный алгоритм над приведенным компетентным алгоритмом, построенным в свою очередь над этим же множеством $\{A\}$ алгоритмов постоянной сложности.

4. Об эффективном алгоритме над множеством алгоритмов из основных классов алгоритмов решения задачи Z

Полученные выше леммы и теоремы не дают ответа, какие именно подмножества основных классов алгоритмов и/или алгоритмы следует использовать при построении эффективного алгоритма. Приведенные ниже леммы и теоремы отвечают на этот вопрос.

Определение 37. Множество алгоритмов $\{A\}$ является *полным* относительно множества алгоритмов $\{\widetilde{A}\}$, если:

$$\forall\, \widetilde{A} \in \{\widetilde{A}\} \quad \exists\, A \in \{A\}: \quad A \cong \widetilde{A}\,.$$

Предложение 16. Пусть множество алгоритмов $\{A\}$ является *полным* относительно множества алгоритмов $\{\widetilde{A}\}$. Тогда:

$$\forall\, \widetilde{A} \in \{\widetilde{A}\} \quad \forall Z \in \aleph_{\widetilde{A}}$$
$$U\left(\widetilde{A}(Z)\right) \ge \min_{A \in \{A(Z)\} \subseteq \{A\}} U\left(A(Z)\right).$$

Доказательство: Очевидным образом следует из Определения 37 полного множества алгоритмов и Предложения 1 о равенстве сложностей подобных алгоритмов.

■

Это предложение имеет очевидное

Следствие 8. (Предложения 16) Пусть множество алгоритмов $\{A\}$ является *полным* относительно множества алгоритмов $\{\widetilde{A}\}$. Тогда:

$$\forall Z \in \bigcup_{\widetilde{A} \in \{\widetilde{A}\}} \aleph_{\widetilde{A}}$$
$$\min_{\widetilde{A} \in \{\widetilde{A}(Z)\} \subseteq \{\widetilde{A}\}} U\left(\widetilde{A}(Z)\right) \ge \min_{A \in \{A(Z)\} \subseteq \{A\}} U\left(A(Z)\right). \quad (61)$$

Лемма 8. Пусть $\{A_{DC}\}$ множество, содержащее единственный алгоритм A_{DC} прямого вычисления свертки, аналитическое представление которого задается формулой (1). Пусть $\{A_{RF}\}$ - множество алгоритмов рекурсивного вычисления свертки, каждый из которых имеет аналитическое представление в виде формулы.

$$y(n)=\sum_{k=1}^{K}a_k y(n-k)+\sum_{m\in D}b_m x(n-m). \qquad (62)$$

Тогда замыкание $[\{A_{DC}\}]$ является полным относительно множества $\{A_{RF}\}$.

Доказательство:

Пусть $A_{RF}\in\{A_{RF}\}$ - некоторый произвольный алгоритм рекурсивного вычисления свертки (рекурсивный фильтр) с порядком фильтра *K,* произвольным множеством (различных) индексов *D* и некоторыми значения коэффициентов $\{a_k\}_{k=1}^{K}$ и $\{b_m\}_{m\in D}$ $(a_0\neq 0)$. Формула вычисления свертки алгоритмом рекурсивного вычисления свертки задается выражением (62) [11, 12, 29, 30, 34, 35, 39].

Для доказательства достаточно показать, что для этого алгоритма рекурсивного вычисления свертки найдется подобный ему алгоритм из замыкания $[\{A_{DC}\}]$. То есть алгоритм из замыкания $[\{A_{DC}\}]$, для которого формула вычисления свертки (38) будет тождественна с точностью до обозначений формуле (62).

Вначале проведем дополнительное построение и определим для множества индексов *D* функцию $m(s)$ $\left(s=\overline{0,|D|-1}\right)$, которая по номеру элемента в (неупорядоченном) множестве индексов *D* выдает значение индекса в упорядоченном множестве тех же индексов из *D*:

$$m(s)=m:\left(\begin{array}{l} m\in D \quad \wedge \\ \left|\{n: n\in D\wedge n<m\}\right|=s-1\end{array}\right). \qquad (63)$$
$$\left(s=\overline{0,|D|-1}\right)$$

Очевидно, множества неупорядоченных индексов *D* и множество индексов $\{m(s)\}_{s=0}^{|D|-1}$ совпадают.

Теперь построим алгоритм из замыкания $[\{A_{DC}\}]$ со следующими параметрами:

(a) $K_h=K+1$,

(b) $K_x=1$,

(c) $S=|D|$,

(d) $g_h(t)=\begin{cases}1, & t=0,\\ -a_t, & t=\overline{1,K_h-1.}\end{cases}$;

(e) $g_x(0)=1$;

(f) границы интервалов покрытия $D_s=[m(s),m(s)]$ $\left(s=\overline{0,S-1}\right)$, где $m(s)$ определена в (63);

(g) $\tilde{h}_s(m)=b_{m(s)}$ $\left(s=\overline{0,S-1}\right)$;

(h) алгоритм A_{prep} - отсутствует;

(i) алгоритмы $A_s=A_{DC}$ $\left(s=\overline{0,S-1}\right)$.

Подставим приведенные параметры в формулу расчета свертки в алгоритме модели CR (38). Получим промежуточный результат:

$$y(n)=\sum_{t=1}^{K_h+K_x-2}\tilde{g}(t)y(n-t)+$$
$$+\sum_{s=0}^{S-1}\sum_{m\in D_s}\tilde{h}_s(m)\left(\sum_{l=0}^{K_x-1}g_x(l)x(n-m-l)\right)=$$
$$=\sum_{t=1}^{K_h-1}\tilde{g}(t)y(n-t)+\sum_{s=0}^{S-1}b_{m(s)}x(n-m(s)). \qquad (64)$$

Для дальнейших преобразований требуется определить выражение для последовательности $\tilde{g}(t)=-g(t)/g(0)$ и, соответственно для последовательности $g(t)$, задаваемой соотношением (32):

$$g(t)=\sum_{k=\max(0,t-K_x)}^{\min(K_h-1,t)}g_h(k)g_x(t-k)=$$
$$=\sum_{k=t}^{t}g_h(k)=g_h(t) \quad \left(t=\overline{0,K_h-2}\right)$$

С учетом того, что $g_h(0)=1$ (параметр (d)) имеем:

$$\tilde{g}(t)=g_h(t) \quad \left(t=\overline{1,K_h-1}\right).$$
$$\tilde{g}(t)=\begin{cases}1, & t=0,\\ -g_h(t)/g_h(0)=a_t, & t=\overline{1,K_h-2.}\end{cases}$$

Тогда формула вычисления свертки в алгоритме модели CR из замыкания $[\{A_{DC}\}]$ примет вид:

$$y(n)=\sum_{t=1}^{K_h-1}a_t y(n-t)+\sum_{s=0}^{S-1}b_{m(s)}x(n-m(s)).$$

Подставляя значение параметра $K_h=K+1$, $S=|D|$ и заменяя индекс *t* на *k*, получим выражение

$$y(n)=\sum_{k=1}^{K-1}a_k y(n-k)+\sum_{s=0}^{|D|-1}b_{m(s)}x(n-m(s)),$$

которое отличается от соотношения (62) только второй суммой в правой части. Делаем эквивалентные преобразования этой суммы:

$$\sum_{s=0}^{|D|-1}b_{m(s)}x(n-m(s))=\sum_{s\in\{s\}_{s=0}^{|D|-1}}b_{m(s)}x(n-m(s))=$$
$$=\sum_{m\in\{m(s)\}_{s=0}^{|D|-1}}b_m x(n-m)=\sum_{m\in D}b_m x(n-m).$$

Тогда получаем окончательную формулу вычисления свертки в алгоритме модели CR из замыкания $[\{A_{DC}\}]$

$$y(n)=\sum_{k=1}^{K}a_k y(n-k)+\sum_{m\in D}b_m x(n-m),$$

которая абсолютно эквивалентна формуле вычисления свертки для рекурсивного алгоритма (62).

Единственным принципиальным **реализационным** отличием полученного выражения от выражения (62) является то, что в алгоритме из замыкания $\left[\{A_{DC}\}\right]$ вычисление $|D|$ вырожденных сверток (39) вида $\left\{\tilde{y}_s(n)=b_{m(s)}x(n-m(s))\right\}_{s=\overline{0,|D|-1}}$ выполняет алгоритм A_{DC}, действие которого заключается в непосредственном вычислении $|D|$ соответствующих произведений $b_m x(n-m)$ для вырожденных сверток. На полученную формулу вычисления искомой свертки этот реализационный факт, как очевидно из представленных выражений, не оказывает влияния.

Таким образом показано, что для произвольного алгоритма рекурсивного вычисления свертки из множества $\{A_{RF}\}$ найдется подобный ему алгоритм из замыкания $\left[\{A_{DC}\}\right]$. А следовательно, по определению 37, замыкание $\left[\{A_{DC}\}\right]$ является полным относительно множества алгоритмов $\{A_{RF}\}$. Что и требовалось доказать.

■

Теорема 6 Пусть $\{A_{DC}\}$ - множество, содержащее единственный алгоритм A_{DC} прямого вычисления свертки; $\{A_{FC}\}$ - множество алгоритмов вычисления свертки на основе ДОП; $\{A_{RF}\}$ - множество алгоритмов рекурсивного вычисления свертки. Пусть $\breve{A}^{\oplus}$-приведенный компетентный алгоритм над множеством алгоритмов постоянной сложности $\{A_{DC}\}\cup\{A_{FC}\}$. Тогда для любой задачи $Z\in\bigcup_{A\in\{A_{DC}\}\cup\{A_{FC}\}\cup\{A_{RF}\}}\aleph_A$ индуцированный алгоритм $A^{\Im}_{\oplus}(Z)\in\left[\left\{\breve{A}^{\oplus}\right\}\right]$ является эффективным над множеством алгоритмов $\{A_{DC}\}\cup\{A_{FC}\}\cup\{A_{FR}\}$.

Доказательство:

Рассмотрим индуцированный алгоритм $A^{\Im}(Z)\in\left[\{A_{DC}\}\cup\{A_{FC}\}\right]$ задачи Z над множеством алгоритмов постоянной сложности $\{A_{DC}\}\cup\{A_{FC}\}$. Тогда, в соответствие с Теоремой 5 для этой задачи Z справедливо выражение (58):

$$U\left(A^{\Im}_{\oplus}(Z)\right)\le U\left(A^{\Im}(Z)\right). \qquad (65)$$

В то же время, по Теореме 3, индуцированный алгоритм $A^{\Im}(Z)$ является эффективным алгоритмом для задачи Z над множеством $\{A_{DC}\}\cup\{A_{FC}\}$, то есть по определению эффективного алгоритма:

$$U\left(A^{\Im}(Z)\right)\le\min_{A\in\{A(Z)\}\subseteq\{A_{DC}\}\cup\{A_{FC}\}}U(A(Z)). \qquad (66)$$

Неравенства (65) и (66) приводят к соотношению:

$$U\left(A^{\Im}_{\oplus}(Z)\right)\le U\left(A^{*}(Z)\right)=\min_{A\in\{A(Z)\}\subseteq\{A_{DC}\}\cup\{A_{FC}\}}U(A(Z)). \qquad (67)$$

С другой стороны, в соответствие с Леммой 8, замыкание $\left[\{A_{DC}\}\cup\{A_{FC}\}\right]$ является полным относительно множества алгоритмов $\{A_{RF}\}$ и поэтому выполняется следствие 8 и, соответственно, неравенство (61). Тогда для данной задачи Z справедливо (если задача не попадает в область определения рекурсивных алгоритмов, тогда эти алгоритмы ее в принципе не решают и говорить о сложности решения такой задачи рекурсивными алгоритмами бессмысленно):

$$\text{если}\quad Z\in\bigcup_{A_{RF}\in\{A_{RF}\}}\aleph_{A_{RF}}$$
$$\min_{\substack{A_{RF}\in\{A_{RF}(Z)\}\\ \subseteq\{A_{RF}\}}}U(A_{RF}(Z))\ge\min_{\substack{A\in\{A(Z)\}\\ \subseteq[\{A_{DC}\}]}}U(A(Z)).$$

В соответствии с Предложением 14, справедливо:

$$\{A_{DC}\}\subseteq\{A_{DC}\}\cup\{A_{FC}\}\ \Rightarrow$$
$$\Rightarrow\left[\{A_{DC}\}\right]\subseteq\left[\{A_{DC}\}\cup\{A_{FC}\}\right].$$

Тогда получаем:

$$\text{если}\quad Z\in\bigcup_{A_{RF}\in\{A_{RF}\}}\aleph_{A_{RF}}$$
$$\min_{\substack{A_{RF}\in\{A_{RF}(Z)\}\\ \subseteq\{A_{RF}\}}}U(A_{RF}(Z))\ge\min_{\substack{A\in\{A(Z)\}\\ \subseteq[\{A_{DC}\}\cup\{A_{FC}\}]}}U(A(Z))=$$
$$=U\left(A^{\Im}(Z)\right)\ge U\left(A^{\Im}_{\oplus}(Z)\right).$$

Объединяя это неравенство с (67), имеем

$$U\left(A^{\Im}_{\oplus}(Z)\right)\le\min_{A\in\{A(Z)\}\subseteq\{A_{DC}\}\cup\{A_{FC}\}}U(A(Z))\ \wedge$$
$$U\left(A^{\Im}_{\oplus}(Z)\right)\le\min_{\substack{A\in\{A(Z)\}\\ \subseteq\{A_{RF}\}}}U(A(Z)),\ Z\in\bigcup_{A_{RF}\in\{A_{RF}\}}\aleph_{A_{RF}}.$$

Откуда окончательно.

$$U\left(A^{\Im}_{\oplus}(Z)\right)\le$$
$$\le\min_{A\in\{A(Z)\}\subseteq\{A_{DC}\}\cup\{A_{FC}\}\cup\{A_{RF}\}}U(A(Z))=$$
$$=U\left(A^{*}(Z)\right)$$

где $A^{*}(Z)$ - наилучший алгоритм для задачи Z из множества $\{A_{DC}\}\cup\{A_{FC}\}\cup\{A_{RF}\}$.

Последнее неравенство, в соответствие с определением 18, означает, что алгоритм $A^{\Im}_{\oplus}(Z)\in\left[\left\{\breve{A}^{\oplus}\right\}\right]$ - эффективный алгоритмом для задачи Z над множеством алгоритмов $\{A_{DC}\}\cup\{A_{FC}\}\cup\{A_{FR}\}$.

■

Доказанная теорема позволяет сформулировать важное конструктивное следствие, которое указывает метод построения эффективного алгоритма.

Следствие 9 (Теоремы 6) Для построения эффективного алгоритма для задачи Z над множеством алгоритмов $\{A_{DC}\}\cup\{A_{FC}\}\cup\{A_{FR}\}$ достаточно:

- построить приведенный компетентный алгоритм $\breve{A}^{\oplus}$ над множеством алгоритмов $\{A_{DC}\}\cup\{A_{FC}\}$,
- построить индуцированный алгоритм $A_{\oplus}^{\Im}(Z)\in\left[\left\{\breve{A}^{\oplus}\right\}\right]$ задачи Z над множеством $\left\{\breve{A}^{\oplus}\right\}$ из одного приведенного компетентного алгоритма $\breve{A}^{\oplus}$.

Приведенное следствие будет использовано в следующем разделе при синтезе метода построения эффективного алгоритма.

Замечание. Ниже предполагается, что алгоритмы постоянной сложности, участвующие в построении эффективного алгоритма, распространены на всю требуемую область определения. Поэтому этап построения распространения соответствующих алгоритмов в рассматриваемом методе не используется.

5. Описание метода синтеза эффективного алгоритма над множеством алгоритмов решения задачи Z

Для формализации метода представим его описание в виде двух частей: исходных данных и реализации.

5.1. Формализованное описание метода синтеза эффективного алгоритма

Исходные данные

Исходная информация метода

- допустимые максимальный размер КИХ $M_{\max}$ и максимальный размер обрабатываемого сигнала $N_{\max}$;
- множество (приведенных и «неприведенных») алгоритмов постоянной сложности $\{A_i\}_{i=0}^{I-1}$;
- информация об области определения $\aleph_{A_i}\ \ (i=0,I-1)$ каждого из алгоритмов, задаваемая перечислением пар (M,N), которые входят в эту область определения;

  информация о сложности каждого алгоритма из множества $\{A_i\}_{i=0}^{I-1}$, заданная для каждого алгоритма матрицей

  $\left\|u_{A_i}(M,N)\right\|_{\substack{M=1,M_{\max}\\N=1,N_{\max}}}\ \ u_{A_i}(M,N)\in R_+\cup\{\infty\}$.

Априорная информация $\Im_0$ для задачи Z

- отсчеты ИХ $\{h(m)\}_{m=0}^{M-1}\ \ (M\le M_{\max})$,
- априорная информация о входном сигнале $\Im_x=\left(N,\Im_{(x)}\right)$, которая включает в себя:

  - информацию о размере обрабатываемого сигнала $N\ \ (N\le N_{\max})$,

  - априорную информацию о свойствах сигнала $\Im_{(x)}$ (см. п.1.2).

Реализация метода

Предварительный этап

- построение (синтез) приведенного компетентного алгоритма $\breve{A}^{\oplus}$ над множеством алгоритмов постоянной сложности $\{A_i\}_{i=0}^{I-1}$.

Подготовительный этап (решения задачи Z)

- построение (ниже - синтез) индуцированного алгоритма $A_{\oplus}^{\Im}(Z)\in\left[\left\{\breve{A}^{\oplus}\right\}\right]$ задачи Z над множеством $\left\{\breve{A}^{\oplus}\right\}$ из алгоритма $\breve{A}^{\oplus}$.

5.2. Комментарии относительно метода синтеза эффективного алгоритма

Как видно из представленного формализованного описания, исходные данные метода разделены на две категории:

- исходная информация метода и
- априорная информация $\Im_0$ для задачи Z.

Подобное разделение обусловлено различными этапами использования различных частей исходных данных в реализации метода. А именно, исходная информация метода, куда относится исходное множество алгоритмов решения задачи Z и информация об их сложности и области определения, требуется для предварительного этапа, на котором производится построение приведенного компетентного алгоритма $\breve{A}^{\oplus}$. Причем момент построения алгоритма $\breve{A}^{\oplus}$ не связан с моментом решения какой-либо конкретной задачи Z, а выполняется предварительно, например, в момент инициализации программы или устройства, реализующего рассматриваемый метод синтеза эффективного алгоритма. При (алгоритмической, программной, аппаратной) реализации метода параметры построенного алгоритма $\breve{A}^{\oplus}$ сохраняются и в неизменном виде используются в рамках каждого подготовительного этапа, выполняемого при смене априорной информации $\Im_0=\left(\{h(n)\}_{n=0}^{M-1},\Im_x\right)$ задачи Z.

Вторая категория исходных данных метода - априорная информации $\Im_0=\left(\{h(n)\}_{n=0}^{M-1},\Im_x\right)$ для задачи Z - используется уже на подготовительном этапе решения задачи Z совместно с построенным приведенным компетентным алгоритмом $\breve{A}^{\oplus}$.

Представленное формализованное описание метода синтеза эффективного алгоритма определяет принцип работы метода. Однако для построения алгоритма синтеза, реализующего данный метод, требуется указание конкретного способа синтеза при-

веденного компетентного алгоритма $\breve{A}^{\oplus}$ на предварительном этап и определенного способа синтеза индуцированного алгоритма $A_{\oplus}^{\Im}(Z) \in \left[\left\{\breve{A}^{\oplus}\right\}\right]$ на подготовительном этапе решения задачи Z.

Синтез приведенного компетентного алгоритма $\breve{A}^{\oplus}$ рассматривается в разделе 2.2.

Синтез алгоритма $A_{\oplus}^{\Im}(Z) \in \left[\left\{\breve{A}^{\oplus}\right\}\right]$, индуцированного априорной информацией задачи Z, рассматривается в разделе 2.3.

6. Синтез приведенного компетентного алгоритма над множеством алгоритмов постоянной сложности

6.1. Формализованное описание синтеза приведенного компетентного алгоритма над множеством алгоритмов постоянной сложности

Данные

- допустимые максимальный размер КИХ $M_{\max}$ и максимальный размер обрабатываемого сигнала $N_{\max}$;
- множество (приведенных и «неприведенных») алгоритмов постоянной сложности $\{A_i\}_{i=0}^{I-1}$;
- информация об области определения $\aleph_{A_i}$ $(i = 0, I-1)$ каждого из алгоритмов, задаваемая перечислением пар (M,N), которые входят в эту область определения;
- информация о сложности каждого алгоритма из множества $\{A_i\}_{i=0}^{I-1}$, заданная для каждого алгоритма матрицей $\left\|u_{A_i}(M,N)\right\|_{\substack{M=1,M_{\max}\\N=1,N_{\max}}}$ $u_{A_i}(M,N) \in R_+ \cup \{\infty\}$.

Реализация

- синтез компетентного алгоритма $A^{\oplus}$ над множеством алгоритмов постоянной сложности $\{A_i\}_{i=0}^{I-1}$, определенного на области определения вида $\aleph_{A^{\oplus}(M,N)} = \left\{(M,N):\ \begin{array}{c} 1 \le M \le M_{\max}, \\ 1 \le N \le N_{\max} \end{array}\right\}$.
- синтез приведенного компетентного алгоритма $\breve{A}^{\oplus}$, определенного на той же области определения $\aleph_{\breve{A}^{\oplus}(M,N)} = \aleph_{A^{\oplus}(M,N)}$, на основе компетентного алгоритма $A^{\oplus}$: $A^{\oplus} \to \breve{A}^{\oplus}$.

6.2. Комментарии относительно синтеза приведенного компетентного алгоритма над множеством алгоритмов постоянной сложности

Как видно из формализованного описания процесса синтеза приведенного компетентного алгоритма над множеством алгоритмов постоянной сложности, его реализация выполнятся в два шага. На первом шаге производится построение компетентного алгоритма над множеством алгоритмов постоянной сложности, а на втором шаге на основе полученного компетентного алгоритма формируется приведенный компетентный алгоритм.

Обоснованием именно такой структуры метода, а также приведенных ниже детализаций его реализации являются:

- Определение 27 компетентного алгоритма, которое фактически указывает способ его построения,
- Теорема 1, которая, с одной стороны, доказывает собственно возможность построения приведенного алгоритма для любого алгоритма постоянной сложности и, с другой стороны, доказательство которой практически в явной форме содержит способ синтеза приведенного алгоритма на основании конкретного алгоритма постоянной сложности.
- Лемма 3, которая доказывает, что сложности приведенного компетентного алгоритма (построенного над множеством алгоритмов постоянной сложности) и приведенного компетентного алгоритма (построенного над множеством тех же приведенных алгоритмов постоянной сложности) совпадают. Эта лемма позволяет исключить этап построения для всех алгоритмов из множества $\{A_i\}_{i=0}^{I-1}$ соответствующих им приведенных алгоритмов.

Процедура синтеза компетентного алгоритма $A^{\oplus}$ над множеством алгоритмов постоянной сложности $\{A_i\}_{i=0}^{I-1}$ рассматривается в разделе 6.3.

Процедура синтеза приведенного компетентного алгоритма $\breve{A}^{\oplus}$ на основе компетентного алгоритма $A^{\oplus}$ рассматривается в разделе 6.4.

6.3 Процедура синтеза компетентного алгоритма над множеством алгоритмов постоянной сложности

Результатом работы процедуры синтеза компетентного алгоритма над множеством алгоритмов постоянной сложности является определенный своими (внутренними) параметрами компетентный алгоритм $A^{\oplus}$, заданный на области определения $\aleph_{A^{\oplus}(M,N)} = \{(M,N): 1 \le M \le M_{\max}, 1 \le N \le N_{\max}\}$.

Внутренними параметрами компетентного алгоритма являются:

- величины $M_{\max}$, $N_{\max}$,
- *матрица сложности* компетентного алгоритма $\left\|u_{A^{\oplus}}(M,N)\right\|_{(M,N)\in[1,M_{\max}]\times[1,N_{\max}]}$,
- *матрица индексов* компетентного алгоритма $\left\|a(M,N)\right\|_{(M,N)\in[1,M_{\max}]\times[1,N_{\max}]}$, каждый элемент которой содержит либо номер наилучшего алгоритма постоянной сложности для задачи с соот-

ветствующими параметрами $(M,N)\in\aleph_{A^{\oplus}(M,N)}$,

либо символ неопределенности «Δ«, указывающий на отсутствие алгоритма решения соответствующей задачи:

$a(M,N)\in\{i:\ \ i\in[0,I-1]\}\cup\{\Delta\}$.

Исходные данные процедуры синтеза

- допустимые максимальный размер КИХ $M_{\max}$ и максимальный размер обрабатываемого сигнала $N_{\max}$,
- множество (приведенных и «неприведенных») алгоритмов постоянной сложности $\{A_i\}_{i=0}^{I-1}$,
- информация об области определения $\aleph_{A_i}\ \ (i=0,I-1)$ каждого из алгоритмов, задаваемая перечислением пар (M,N), которые входят в эту область определения;
- информация о сложности каждого алгоритма из множества $\{A_i\}_{i=0}^{I-1}$, заданная для каждого алгоритма матрицей $\left\|u_{A_i}(M,N)\right\|_{\substack{M=1,M_{\max}\\N=1,N_{\max}}}\quad u_{A_i}(M,N)\in R_+\cup\{\infty\}$.

Требуемый результат процедуры

Внутренние параметры компетентного алгоритма.

Реализация процедуры

Величины $M_{\max}$, $N_{\max}$ берутся из исходных данных процедуры.

Матрицы сложности и индексов задаются следующим образом:

$$u_{A^{\oplus}}(M,N)=\begin{cases}\min\limits_{\left\{j:\ \substack{j\in[0,I-1]\\(M,N)\in\aleph_{A_j}(M,N)}\right\}} u_{A_j}(M,N),\\ \qquad\text{если}\quad (M,N)\in\bigcup\limits_{i=0}^{I-1}\aleph_{A_i(M,N)},\\ \infty,\quad\text{если}\quad (M,N)\notin\bigcup\limits_{i=0}^{I-1}\aleph_{A_i(M,N)}.\end{cases}$$

$$a(M,N)=\begin{cases}\arg\min\limits_{\left\{j:\ \substack{j\in[0,I-1]\\(M,N)\in\aleph_{A_j(M,N)}}\right\}} u_{A_j}(M,N),\\ \qquad\text{если}\quad (M,N)\in\bigcup\limits_{i=0}^{I-1}\aleph_{A_i(M,N)},\\ \Delta,\ \text{если}\ (M,N)\notin\bigcup\limits_{i=0}^{I-1}\aleph_{A_i(M,N)}.\end{cases}$$

В заключение настоящего раздела приведем описание функционирования построенного компетентного алгоритма.

Функционирование компетентного алгоритма

Входные данные о задаче $Z\left(\Im_0,\{x(n)\}_{n=0}^{N-1}\right)$ компетентного алгоритма

- отсчеты входного сигнала $\{x(n)\}_{n=0}^{N-1}$,
- априорная информация $\Im_0=\left(\{h(n)\}_{n=0}^{M-1},\Im_x\right)$ о задаче Z, включающая:
  - отсчеты ИХ $\{h(m)\}_{m=0}^{M-1}\quad(M\le M_{\max})$,
  - априорная информация о входном сигнале $\Im_x=\left(N,\Im_{(x)}\right)$ (не используется).

Реализация компетентного алгоритма

АЛГОРИТМ $A^{\oplus}$ *- НАЧАЛО*
IF $(1\le M\le M_{\max})\wedge(1\le N\le N_{\max})$ $\wedge\left(a(M,N)\ne\Delta\right)$
THEN Выполняем алгоритм $A_{a(M,N)}(Z)$
ELSE «Задача не решается»;
АЛГОРИТМ $A^{\oplus}$ *- КОНЕЦ.*

6.4. Процедура синтеза приведенного компетентного алгоритма

Результатом работы процедуры синтеза приведенного компетентного алгоритма $\breve{A}^{\oplus}$ на основе компетентного алгоритма $A^{\oplus}$ является определенный своими (внутренними) параметрами приведенный компетентный алгоритм $\breve{A}^{\oplus}$, заданный на области определения

$$\aleph_{\breve{A}^{\oplus}(M,N)}=\aleph_{A^{\oplus}(M,N)}=$$
$$=\ \{(M,N):\ 1\le M\le M_{\max},1\le N\le N_{\max}\}.$$

Внутренними параметрами приведенного компетентного алгоритма являются:

- величины $M_{\max}$, $N_{\max}$,
- *матрица сложности* алгоритма $\left\|u_{\breve{A}^{\oplus}}(M,N)\right\|_{(M,N)\in[1,M_{\max}]\times[1,N_{\max}]}$,
- *матрица переадресаций* алгоритма $\|\Xi(M,N)\|_{(M,N)\in[1,M_{\max}]\times[1,N_{\max}]}$. Каждый элемент матрицы переадресации содержит:
  - либо множество (перечисление) $\left\{\left(M_s,N_s,\widehat{M}_s,\widehat{N}_s\right)\right\}_{s=0}^{|\Xi(M,N)|-1}$ из $|\Xi(M,N)|$ переадресаций вида $\left(M_s,N_s,\widehat{M}_s,\widehat{N}_s\right)$, удовлетворяющих условию $\widehat{M}_s\le M_s\wedge\widehat{N}_s\le N_s\wedge N_s-\widehat{N}_s\ge M_s-\widehat{M}_s$,
  - либо символ пустого множества «∅«, указывающий на отсутствие переадресации,
  - либо символ «Δ», указывающий на невозможность решения соответствующей задачи.

  Первая пара индексов (M_s,N_s) в четверке $\left(M_s,N_s,\widehat{M}_s,\widehat{N}_s\right)$ является собственно переадресацией. Вторая пара индексов $\left(\widehat{M}_s,\widehat{N}_s\right)$ является комментарием или дополнительной информацией, указывающей на то, какое именно полезное количество отсчетов сигнала и ИХ обрабатывается при решении соответствующей задачи Z.

Исходные данные процедуры

Исходными данными процедуры синтеза являются внутренние параметры компетентного алгоритм $A^{\oplus}$, которые включают в себя:

- величины $M_{\max}$, $N_{\max}$,
- матрица сложности компетентного алгоритма $\left\| u_{A^{\oplus}}(M,N) \right\|_{(M,N)\in[1,M_{\max}]\times[1,N_{\max}]}$,
- матрица индексов компетентного алгоритма $\left\| a(M,N) \right\|_{(M,N)\in[1,M_{\max}]\times[1,N_{\max}]}$.

Требуемый результат процедуры

Внутренние параметры приведенного компетентного алгоритма.

Реализация процедуры

Величины $M_{\max}$, $N_{\max}$ берутся из исходных данных процедуры (внутренних параметров компетентного алгоритма).

Формирование матриц сложности и переадресаций производится итерационным образом, который условно можно разделить на три этапа – этап инициализации, основной итерационный этап и завершающий этап.

Предварительный этап

$$u_{\bar{A}^{\oplus}}(M,N) = u_{A^{\oplus}}(M,N),$$

$$\Xi(M,N) = \begin{cases} \varnothing, & a(M,N) \neq \Delta, \\ \Delta, & a(M,N) = \Delta. \end{cases} \quad (M,N) \in \aleph_{\bar{A}^{\oplus}}.$$

.

Основной итерационный этап

WHILE (TRUE) BEGIN

$$(M,N) := \arg \min_{(M,N)\in\aleph_{\bar{A}^{\oplus}}(M,N)} \left(u_{\bar{A}^{\oplus}}(M,N) > \min \left(\begin{array}{l} \min\limits_{\substack{m=\overline{1,M-2} \\ (m,N-(M-m))\in\aleph_{\bar{A}^{\oplus}}(M,N), \\ (M-m,N-m)\in\aleph_{\bar{A}^{\oplus}}(M,N)}} \left(\begin{array}{l} u_{\bar{A}^{\oplus}}(m,N-(M-m)) + \\ + u_{\bar{A}^{\oplus}}(M-m,N-m) + \xi_{add} \end{array} \right), \\ \min\limits_{\substack{m=0,1,2\ldots \\ n=0,1,2,\ldots \\ (M+m,N+m+n))\in\aleph_{\bar{A}^{\oplus}}(M,N)}} \left(\begin{array}{l} u_{\bar{A}^{\oplus}}(M+m,N+m+n)\cdot \\ \cdot \dfrac{(N-M+n+1)}{(N-M+1)} \end{array} \right) \\ \min\limits_{\substack{n=\overline{1,\lfloor N/2 \rfloor} \\ (M,n)\in\aleph_{\bar{A}^{\oplus}}(M,N), \\ (M,N-n+M-1)\in\aleph_{\bar{A}^{\oplus}}(M,N)}} \dfrac{\left(\begin{array}{l} u_{\bar{A}^{\oplus}}(M,n)(n-M+1) + \\ u_{\bar{A}^{\oplus}}(M,N-n+M-1)(N-n) \end{array} \right)}{N-M+1} \end{array} \right) \right)$$

IF $\neg\exists(M,N)$ *THEN EXIT; // завершаем*

// если пара (M,N) *найдена*

IF «Минимальное значение дает первое выражение»

THEN

$$u_{\bar{A}^{\oplus}}(M,N) := \min_{\substack{m=\overline{1,M-2} \\ (m,N-(M-m))\in\aleph_{A}(M,N), \\ (M-m,N-m)\in\aleph_{A}(M,N)}} \left(\begin{array}{l} u_{A^{\oplus}}(m,N-(M-m)) + \\ + u_{\bar{A}^{\oplus}}(M-m,N-m) + \\ + \xi_{add} \end{array} \right),$$

$$\Xi(M,N) := \begin{Bmatrix} (M_1,N_1,M_1,N_1), \\ (M_0,N_0,M_0,N_0) \end{Bmatrix},$$

$$\text{где} \left(\begin{array}{l} (M_0,N_0)\in\aleph_{\bar{A}^{\oplus}} \wedge (M_1,N_1)\in\aleph_{\bar{A}^{\oplus}} \wedge \\ (M_0+M_1=M)\wedge \\ (N_0-M_0=N-M)\wedge \\ (N_1-M_1=N-M)\wedge \\ \left(\begin{array}{l} u_{\bar{A}^{\oplus}}(M,N) = \\ = u_{\bar{A}^{\oplus}}(M_0,N_0) + u_{\bar{A}^{\oplus}}(M_1,N_1) + \xi_{add} \end{array} \right) \end{array} \right);$$

IF «Минимальное значение дает второе выражение»

THEN

$$u_{\bar{A}^{\oplus}}(M,N) := \min_{\substack{m=0,1,2\ldots \\ n=0,1,2,\ldots \\ (M+m,N+m+n))\in\aleph_{\bar{A}^{\oplus}}(M,N)}} \left(u_{\bar{A}^{\oplus}}(M+m,N+m+n)\frac{(N-M+n+1)}{(N-M+1)} \right),$$

$$\Xi(M,N) := \{(M_0,N_0,M,N)\},$$

$$\text{где} \left(\begin{array}{l} (M_0,N_0)\in\aleph_{\bar{A}^{\oplus}} \wedge (M_0\geq M)\wedge \\ (N_0-M_0\geq N-M)\wedge \\ \wedge \left(\begin{array}{l} u_{\bar{A}^{\oplus}}(M,N)(N-M+1) = \\ = u_{\bar{A}^{\oplus}}(M_0,N_0)(N_0-M_0+1) \end{array} \right) \end{array} \right);$$

IF «Минимальное значение дает третье выражение»

THEN

$$u_{\bar{A}^{\oplus}}(M,N) := \min_{\substack{n=\overline{1,\lfloor N/2 \rfloor} \\ (M,n)\in\aleph_{\bar{A}^{\oplus}}(M,N), \\ (M,N-n+M-1)\in\aleph_{\bar{A}^{\oplus}}(M,N)}} \frac{\left(\begin{array}{l} u_{\bar{A}^{\oplus}}(M,n)(n-M+1) + \\ u_{\bar{A}^{\oplus}}(M,N-n+M-1)(N-n) \end{array} \right)}{N-M+1},$$

$$\Xi(M,N) := \begin{Bmatrix} (M,N_1,M,N_1), \\ (M,N_0,M,N_0) \end{Bmatrix},$$

$$\text{где} \left(\begin{array}{l} (M,N_0)\in\aleph_{\bar{A}^{\oplus}} \wedge (M,N_1)\in\aleph_{\bar{A}^{\oplus}} \wedge \\ (N_0+N_1=N+M-1)\wedge \\ \left(\begin{array}{l} u_{\bar{A}^{\oplus}}(M,N) = \\ = u_{\bar{A}^{\oplus}}(M_0,N_0) + u_{\bar{A}^{\oplus}}(M_1,N_1) + \xi_{add} \end{array} \right) \end{array} \right);$$

END; // WHILE (TRUE);.

Как видно из псевдокода, итерации повторяются до тех пор, пока происходят изменения значений функции сложности $u_{\bar{A}^{\oplus}}(M,N)$. Факт того, что эти итерации будут завершены, доказан в Теореме 1.

Завершающий этап

На завершающем этапе производится замена «косвенных» переадресаций $(M_s,N_s,\widehat{M}_s,\widehat{N}_s)\in\Xi(M,N)$, которые указывают на индексы (M_s,N_s), содержащие сами по себе переадресации (например, $\Xi(M_s,N_s,\widehat{M}_s,\widehat{N}_s)=\{(M_0^s,N_0^s,\widehat{M}_0^s,\widehat{N}_0^s),(M_1^s,N_1^s,\widehat{M}_1^s,\widehat{N}_1^s)\}$), на «явные» переадресации, в которых указание производится только на те пары (M,N), где $\Xi(M,N)=\varnothing$. В силу Леммы 1, существует хотя бы одна такая пара (сложность которой не поменяется). Реализацию этого этапа вначале проиллюстрируем на примере.

Пусть для некоторой пары (M,N): $\Xi(M,N)=\{(M_0,N_0,M_0,N_0),(M_1,N_1,M_1,N_1)\}$. Тогда:

$$\Xi(M_0,N_0)=\{(M_0^0,N_0^0,M_0^0,N_0^0),(M_1^0,N_1^0,M_1^0,N_1^0)\},$$
$$\Xi(M_1,N_1)=\{(M_0^1,N_0^1,M_1,N_1)\},$$
$$\Xi(M_1^0,N_1^0)=\{(M_0^2,N_0^2,M_0^2,N_0^2),(M_1^2,N_1^2,M_1^2,N_1^2)\}$$

и

$$\Xi(M_0^0,N_0^0)=\Xi(M_0^1,N_0^1)=\varnothing,$$
$$\Xi(M_0^2,N_0^2)=\Xi(M_1^2,N_1^2)=\varnothing.$$

Тогда искомая (явная) переадресация имеет вид:

$$\Xi(M,N)=\left\{\begin{matrix}(M_0^0,N_0^0,M_0^0,N_0^0),(M_0^1,N_0^1,M_1,N_1),\\(M_0^2,N_0^2,M_0^2,N_0^2),(M_1^2,N_1^2,M_1^2,N_1^2)\end{matrix}\right\}.$$

Псевдокод той части процедуры, в которой выполняется завершающий этап, выглядит следующим образом:

FOR M:=1 to $M_{\max}$ *DO FOR N:=1 to* $N_{\max}$ *DO*
$\Xi(M,N)$*:=GetRefs(M, N);*

Здесь использована рекурсивная функция *GetReferencies*, псевдокод которой имеет вид:

FUNCTION GetRefs(M, N): Ξ *;*
// функция возвращает множество четверок
BEGIN // FUNCTION
result := $\varnothing$*;*
IF $\Xi(M,N)=\varnothing$ *THEN EXIT;*
S := $|\Xi(M,N)|$ *;*
FOR s := 0 TO (S-1) DO
BEGIN // анализирует $(M_s,N_s,\widehat{M}_s,\widehat{N}_s)$
IF $\Xi(M_s,N_s)=\varnothing$
THEN Result := Result $\cup(M_s,N_s,\widehat{M}_s,\widehat{N}_s)$
ELSE Result := Result $\cup$ *GetRefs*(M_s,N_s)*;*
END; // FOR S
END; // FUNCTION

В заключение настоящего раздела приведем описание функционирования построенного приведенного компетентного алгоритма.

Функционирование приведенного компетентного алгоритма

Входные данные о задаче $Z(\Im_0,\{x(n)\}_{n=0}^{N-1})$ *приведенного компетентного алгоритма*

- отсчеты входного сигнала $\{x(n)\}_{n=0}^{N-1}$,
- априорная информация $\Im_0=(\{h(n)\}_{n=0}^{M-1},\Im_x)$ о задаче Z, включающая:
  - отсчеты ИХ $\{h(m)\}_{m=0}^{M-1}\quad(M\le M_{\max})$,
  - априорная информация о входном сигнале $\Im_x=(N,\Im_{(x)})$ (не используется).

Реализация компетентного алгоритма

АЛГОРИТМ $\bar{A}^{\oplus}$ *- НАЧАЛО*
IF $\begin{pmatrix}(1\le M\le M_{\max})\wedge(1\le N\le N_{\max})\\\wedge(\Xi(M,N)\ne\Delta)\end{pmatrix}$
THEN
IF $\Xi(M,N)=\varnothing$
THEN $A^{\oplus}(Z)$ *// результат решения -> y(n)*
ELSE BEGIN
S := $|\Xi(M,N)|$ *;*
$M_0^{\Sigma}=0$*;*
FOR s:=1 TO (S-1) DO
$M_s^{\Sigma}=M_{s-1}^{\Sigma}+\widehat{M}_{s-1}$ *; // разбиение ИХ*
FOR s:= 0 TO (S-1) DO
BEGIN // имеем $(M_s,N_s,\widehat{M}_s,\widehat{N}_s)$
FOR m := 0 TO (M_s-1) *DO*

$$h_s(m)=\begin{cases}h(m+M_s^{\Sigma}), & m=\overline{0,\widehat{M}_s-1},\\0, & m=\overline{\widehat{M}_s,M_s-1}.\end{cases}$$

FOR n := 0 TO (N_s-1) *DO*

$$x_s(n)=\begin{cases}x(n+M_s^{\Sigma}), & n\in[0,\widehat{N}_s-1],\\0, & n=[\widehat{N}_s,N_s-1].\end{cases}$$

$\Im_0^s=(\{h_s(m)\}_{m=0}^{M_s-1},\Delta)$*;* $Z_s=Z(\{x_s(n)\}_{n=0}^{N_s-1},\Im_0^s)$*;*
Выполняем алгоритм $A^{\oplus}(Z_s)$*;*
// результат решения -> $y_s(n)$
END; // for s
FOR n := 0 TO (N – M - 1) DO

$$y(n) := \sum_{s=0}^{S-1} y_s(n);$$

END; // begin

ELSE «Задача не может быть решена»;

АЛГОРИТМ $\breve{A}^{\oplus}$ *- КОНЕЦ.*

7. Синтез алгоритма, индуцированного априорной информацией задачи Z

В настоящем разделе рассматриваются теоретические и алгоритмические основы построения процедуры синтеза алгоритма, индуцированного априорной информацией задачи *Z,* для одной из наиболее распространенных на практике ситуаций. А именно, для случая, когда априорная информация задачи *Z* не содержит данных о свойствах обрабатываемого сигнала: $\Im_{(x)} = \varnothing$. Таким образом, параметры задачи вычисления свертки имеют вид: $Z\left(\Im_0, \{x(n)\}_{n=0}^{N-1}\right)$, где $\Im_0 = \left(\{h(n)\}_{n=0}^{M-1}, (N, \varnothing)\right)$.

Задача построения синтезированного алгоритма в ситуации, когда $\Im_{(x)} \neq \varnothing$, является предметом отдельного исследования и отдельной работы.

7.1. Теоретические основы синтеза индуцированного алгоритма для задач с пустой априорной информацией о свойствах сигнала

Приведем несколько определений из теории конечных разностей, которые нам понадобятся ниже [2, 10, 22, 23, 28, 42, 48, 51].

Определение 38. Пусть K – натуральное число, а $\{a_k\}_{k=}^{K}$ - заданные элементы поля F. Последовательность $h(0), h(1), \ldots$ элементов поля F, удовлетворяющая соотношению

$$h(m) = \begin{cases} b_m, & m = \overline{0, K-1}, \\ \sum_{k=1}^{K} a_k h(m-k) + \varphi(m), & m = \overline{K, M-1}, \end{cases} \qquad (68)$$

называется линейной рекуррентной последовательностью (ЛРП) K-го порядка над полем F. Первые члены $\{b_k\}_{k=0}^{K-1}$ однозначно определяют всю последовательность и называются ее *начальными значениями*. Соотношение (68) называется *линейным рекуррентным соотношением (ЛРС) K-го порядка*.

Определение 39. ЛРС называется *однородным,* если $\varphi(m) \equiv 0$, в противном случае ЛРС называется неоднородным. Соответствующие рекуррентные последовательности (РП) называются *однородной* или *неоднородной ЛРП над полем F*.

Теорема 7. (необходимое условие строгой эффективности индуцированного алгоритма). Пусть приведенный компетентный алгоритм $\breve{A}^{\oplus}$ над множеством алгоритмов постоянной сложности $\{A_i\}_{i=0}^{I-1}$ задан на области определения $\aleph_{\breve{A}^{\oplus}(M,N)} = \{(M,N): 1 \le M \le M_{\max}, 1 \le N \le N_{\max}\}$.

Пусть решается задача $Z\left(\Im_0, \{x(n)\}_{n=0}^{N-1}\right)$ с априорной информацией $\Im_0 = \left(\{h(n)\}_{n=0}^{M-1}, (N, \varnothing)\right)$, где $(M,N) \in \aleph_{\breve{A}(M,N)}$ и $h(m) \neq 0 \quad (m = 0, M-1)$.

Индуцированный алгоритм $A_{\oplus}^{\Im}(Z) \in \left[\left\{\breve{A}^{\oplus}\right\}\right]$ задачи Z над множеством $\left\{\breve{A}^{\oplus}\right\}$ является строго эффективным, только если все отсчеты в КИХ $\{h(m)\}_{m=0}^{M-1}$ удовлетворяют неоднородному ЛРС порядка *K,* а M_0 отсчетов удовлетворяют соответствующему однородного ЛРС, и для пары K, M_0 справедливо:

$$(M_0 > K) \wedge \left(K + u_{\breve{A}^{\oplus}}(M - M_0 + K, N) < u_{\breve{A}^{\oplus}}(M, N)\right).$$

Доказательство: непосредственно следует из необходимых условий Теоремы 4 и свойства функции сложности для приведенного алгоритма постоянной сложности.

■

Теорема 7 формулирует необходимое условие, при котором может быть построен строго эффективный алгоритм. В дополнение к ней может быть сформулирован целый ряд условий, которые являются уже достаточными. Фактически, достаточные условия перечисляют различные частные решения исходной задачи. Одна из возможных подобных теорем приведена ниже

Теорема 8.

Пусть выполняются условия Теоремы 7. Индуцированный алгоритм $A_{\oplus}^{\Im}(Z) \in \left[\left\{\breve{A}^{\oplus}\right\}\right]$ задачи Z над множеством $\left\{\breve{A}^{\oplus}\right\}$ является строго эффективным тогда, когда отсчеты КИХ $\{h(n)\}_{n=0}^{M-1}$ формируют однородную ЛРП порядка *K,* удовлетворяющего неравенству:

$$K < u_{\breve{A}^{\oplus}}(M,N) - 2u_{\breve{A}^{\oplus}}(K,N) - \xi_{add}. \qquad (69)$$

Доказательство:

Индуцированный алгоритм является наилучшим алгоритмом из замыкания по модели CR множества $\left\{\breve{A}^{\oplus}\right\}$. Построим некоторый определенный алгоритм A_{CR} из замыкания по модели CR. Для него справедливо выражение (45) для сложности алгоритмов модели CR и соответствующее неравенство по отношению к сложности индуцированного алгоритма:

$$U\left(A^{\Im}_{\oplus}(Z)\right) \le U\left(A_{CR}(Z)\right) =$$
$$= U\left(A_{prep}\left(Z_{prep}\right)\right) + \left[\sum_{s=0}^{S-1} U\left(A_s\left(Z_s\right)\right) + \xi_{add}(S-1)\right] +$$
$$+ \left(K_h + K_x - 2\right).$$

Поскольку по условию задачи $\Im_{(x)} = \varnothing$, следовательно, $K_x = 1$ и выражение сложности принимает вид:

$$U\left(A^{\Im}_{\oplus}(Z)\right) \le U\left(A_{CR}(Z)\right) =$$
$$= \left[\sum_{s=0}^{S-1} U\left(A_s\left(Z_s\right)\right) + \xi_{add}(S-1)\right] + K_h - 1. \quad (70)$$

Пусть, далее отсчеты КИХ $\{h(n)\}_{n=0}^{M-1}$ формируют однородную рекуррентную последовательность некоторого порядка *K*. Тогда для всех элементов последовательности (кроме крайних) справедливо выражение:

$$h(n) = \sum_{k=1}^{K} \alpha_k h(n-k), \quad n = \overline{K, M-1}.$$

Зададим $K_h = K + 1$ и определим значения ИХ $\{g_h(k)\}_{k=0}^{K_h-1}$ следующим образом:

$$g_h(k) = \begin{cases} 1, & k = 0, \\ -\alpha_k, & k = \overline{1, K_h - 1}. \end{cases}$$

Тогда результат $\{\tilde{h}(n)\}_{n=0}^{M+K_h-2}$ свертки КИХ $\{h(n)\}_{n=0}^{M-1}$ и ИХ $\{g_h(k)\}_{k=0}^{K_h-1}$ имеет вид:

$$\tilde{h}(m) = \begin{cases} \sum_{k=0}^{K_h-1} g_h(k) h(m-k), & m = \overline{0, K_h - 2}, \\ 0, & m = \overline{K_h - 1, M-1}, \\ \sum_{k=0}^{K_h-1} g_h(k) h(m-k), & m = \overline{M, M + K_h - 2}. \end{cases}$$

Очевиден способ построения допустимого покрытия:

$$S = 2, \quad D_0 = [0, K_h - 2], D_0 = [M, M + K_h - 2].$$

Тогда соотношение (70) принимает вид

$$U\left(A^{\Im}_{\oplus}(Z)\right) \le U\left(A_{CR}(Z)\right) =$$
$$= U\left(\breve{A}^{\oplus}(Z_0)\right) + U\left(\breve{A}^{\oplus}(Z_1)\right) + \xi_{add} + K_h - 1.$$

Задачи Z_0, Z_1 отличаются только значениями отсчетов в соответствующих покрытиях D_0, D_1. Тогда

$$U\left(A^{\Im}_{\oplus}(Z)\right) \le U\left(A_{CR}(Z)\right) =$$
$$= 2u_{\breve{A}^{\oplus}}(K_h - 1, N) + \xi_{add} + K_h - 1.$$

По определению строгой эффективности, индуцированный алгоритм будет являться строго эффективным, если $U\left(A^{\Im}_{\oplus}(Z)\right) < u_{A^{\oplus}}(M, N)$. Потребуем, чтобы это неравенство выполнялось не только для индуцированного алгоритма, но и для построенного алгоритма модели CR. Тогда:

$$U\left(A^{\Im}_{\oplus}(Z)\right) \le U\left(A_{CR}(Z)\right) =$$
$$= 2u_{\breve{A}^{\oplus}}(K_h - 1, N) + \xi_{add} + K_h - 1 <$$
$$< u_{A^{\oplus}}(M, N).$$

Откуда, меня порядок слагаемых в двух правых выражениях, получаем:

$$K_h < u_{A^{\oplus}}(M, N) - 2u_{\breve{A}^{\oplus}}(K_h - 1, N) + 1 - \xi_{add}.$$

Учитывая взаимосвязь $K_h = K + 1$, получаем искомое неравенство (69).

■

Следствие 10. (Теоремы 8) Пусть выполняются условия Теоремы 7.1. И пусть $I = 1, A_0 = A_{DC}$. Тогда индуцированный алгоритм $A^{\Im}_{\oplus}(Z) \in \left[\left\{\breve{A}^{\oplus}\right\}\right]$ задачи Z над множеством $\left\{\breve{A}^{\oplus}\right\}$ является строго эффективным, если отсчеты импульсной характеристики формируют однородную ЛРП порядка *K,* удовлетворяющего неравенству

$$K < (M - \xi_{add})/3 \le M/3. \quad (71)$$

Теоремы 7 и 8 позволяют сформулировать конструктивный способ синтеза строго эффективного алгоритма, который оформлен ниже в виде следствия.

Следствие 11. (Теорем 7 и 8) Для построения строго эффективного алгоритма, индуцированного априорной информацией задачи Z, над множеством из приведенного компетентного алгоритма $\left\{\breve{A}^{\oplus}\right\}$ следует:

- перебрать возможные порядки ЛРП $K = \overline{1, M/2}$.
- для каждого возможного порядка ЛРП представить КИХ $\{h(n)\}_{n=0}^{M-1}$ в виде ЛРП всеми возможными способами, удовлетворяющими условию Теоремы 7.
- Для каждого способа представления в виде ЛРП рассчитать значение сложности полученного алгоритма CR.
- Из полученных решений выбрать решение с минимальной сложностью. Полученный алгоритм – индуцированный.
- Если сложность индуцированного алгоритма ниже сложности приведенного компетентного алгоритма $\breve{A}^{\oplus}$, следовательно полученный индуцированный алгоритм является строго эффективным.

Наиболее сложным этапом в приведенном способе синтеза строго эффективного алгоритма является этап представления КИХ $\{h(n)\}_{n=0}^{M-1}$ в виде однородного или неоднородного ЛРП. В литературе описан ряд способов получения такого представле-

ния [2, 10, 13, 22, 23, 28, 32, 34, 36, 38, 42, 48-51]: метод Паде, метод Z-преобразования, аппроксимация сплайнами, аппроксимация в частотной области, аппроксимация рекуррентными функциями, в частности, экспонентами и полиномами, прямая аппроксимация рекуррентным соотношением заданного порядка и др. Эти способы условно можно подразделить на два:

- (точное) представление цифрового сигнала в виде однородного или неоднородного ЛРП заданного порядка,
- аппроксимация цифрового сигнала в виде однородного или неоднородного ЛРП заданного порядка.

Способ аппроксимации КИХ в виде ЛРП ниже не рассматривается, поскольку связан не с исходной проблемой, а с ее измененной постановкой, в которой допускается получение не точного, а приближенного выходного сигнала. Один из возможных способов нахождения представления КИХ в виде ЛРП приведен ниже, как составная часть процедуры синтеза алгоритма, индуцированного априорной информацией.

7.2. Процедура синтеза индуцированного алгоритма

Результатом работы процедуры синтеза приведенного компетентного алгоритма является определенный своими (внутренними) параметрами приведенный компетентный алгоритм $\breve{A}^{\oplus}$, заданный на требуемой области определения $\aleph_{\breve{A}^{\oplus}(M,N)} = \{(M,N):\ \ 1 \le M \le M_{\max}, 1 \le N \le N_{\max}\}$.

Внутренними параметрами алгоритма, индуцированного априорной информацией задачи Z являются:

- величины $M_{\max}$, $N_{\max}$,
- числовые параметры алгоритма модели CR, которые включают в себя (см. п.3.1):

- величину K_h и отсчеты КИХ $\{g_h(k)\}_{k=0}^{K_h-1} \quad (g_h(0)=1)$;
- $K_x = 1, \quad g_x(0) = 1$ (см. ремарку начала п.7);
- величину S и параметры допустимого покрытия $\{[d_0^s, d_1^s]\}_{s=0}^{S-1}$.

Исходные данные процедуры синтеза

Исходными данными процедуры синтеза являются:

- внутренние параметры приведенного компетентного алгоритм $\breve{A}^{\oplus}$, включающие в себя:
  - величины $M_{\max}$, $N_{\max}$,
  - матрица сложности алгоритма $\left\| u_{\breve{A}^{\oplus}}(M,N) \right\|_{(M,N)\in[1,M_{\max}]\times[1,N_{\max}]}$,
  - матрица переадресаций алгоритма $\left\| \Xi(M,N) \right\|_{(M,N)\in[1,M_{\max}]\times[1,N_{\max}]}$ (не используется) ;
- информация о задаче Z, включающая в себя:
  - значения отсчетов КИХ $\{h(n)\}_{n=0}^{M-1}$
  - длину обрабатываемого сигнала N.

Требуемый результат процедуры

Внутренние параметры индуцированного алгоритма.

Реализация процедуры

Величины $M_{\max}$, $N_{\max}$ берутся из исходных данных процедуры.

$$U_{\min} := u_{\breve{A}^{\oplus}}(M,N);$$

$$\left(K_h, \{g_h(k)\}_{k=0}^{K-1}, S, \{[d_0^s, d_1^s]\}_{s=0}^{S-1}, \{\tilde{h}(m)\}_{m=0}^{M+K_h-2}\right)_{\min} := \left(1, \{1\}_{k=0}^{0}, 1, \{[0, M-1]\}_{s=0}^{0}, \{h(m)\}_{m=0}^{M-1}\right);$$

FOR K:=1 TO [M/2] DO BEGIN
FOR m:=0 TO M-K+1 DO BEGIN

$$\begin{pmatrix} a_1 \\ \vdots \\ a_K \end{pmatrix} := \begin{pmatrix} h(m) & \cdots & h(m+K-1) \\ \vdots & \ddots & \vdots \\ h(m+K-1) & \cdots & h(m+2K-2) \end{pmatrix}^{-1} \begin{pmatrix} h(m+K) \\ \vdots \\ h(m+2K) \end{pmatrix}$$

$$K_h := K+1;\ g_h(k) := \begin{cases} 1, & k=0, \\ -a_k, & \overline{k=1,K}. \end{cases}$$

$$h(n) * g_h(n) \Rightarrow \{\tilde{h}(n)\}_{n=0}^{M+K-1};$$

$$M_0 := \sum_{t=0}^{M+K-1} I(\tilde{h}(n)=0)$$

IF not $\left(\begin{pmatrix} (M_0 > K) \wedge \\ \left(K + u_{\breve{A}^{\oplus}}(M - M_0 + K, N) < \right) \\ < u_{\breve{A}^{\oplus}}(M,N) \end{pmatrix} \right)$

THEN CONTINUE; // GoTo FOR m

// ELSE – ищем возможное решение

S := 1;

WHILE (S <> 0)

// Перебор покрытий для заданного S

WHILE ($\exists \{[d_0^s, d_1^s]\}_{s=0}^{S-1}$ *)*

// расчет сложности для данного покр-я

$$U := \sum_{s=0}^{S-1} u_{\breve{A}^{\oplus}}(d_1^s - d_0^s + 1, N);$$

IF ($U < U_{\min}$ *) THEN BEGIN*

$$U_{\min} := U;$$

$$\left(K_h, \{g_h(k)\}_{k=0}^{K-1}, S, \{[d_0^s, d_1^s]\}_{s=0}^{S-1}, \{\tilde{h}(m)\}_{m=0}^{M+K_h-2}\right)_{\min} := \left(K_h, \{g_h(k)\}_{k=0}^{K-1}, S, \{[d_0^s, d_1^s]\}_{s=0}^{S-1}, \{\tilde{h}(m)\}_{m=0}^{M+K_h-2}\right);$$

END; // IF
END; // WHILE
S := S+1;
END; // WHILE S

END; // FOR m
END; // FOR K

// дополнительные присвоения

$$\tilde{g}(k) := \begin{cases} 1, & k = 0, \\ -g_h(k), & k = \overline{1, K_h - 1}. \end{cases}$$

$$\tilde{h}_s(m) = \begin{cases} \tilde{h}(m), & m \in D_s, \\ 0, & m \notin D_s. \end{cases} \quad \left(s = \overline{0, S-1}\right)$$

В заключение настоящего раздела приведем описание функционирования построенного индуцированного алгоритма.

Функционирование индуцированного алгоритма

Входные данные

Отсчеты входного сигнала $\{x(n)\}_{n=0}^{N-1}$,

Реализация индуцированного алгоритма

АЛГОРИТМ $A_{\oplus}^{\Im}$ *- НАЧАЛО*

IF not $(1 \le M \le M_{\max}) \wedge (1 \le N \le N_{\max})$

THEN «Задача не решается»;
ELSE
FOR s=0 TO S-1 DO

$\tilde{A}^{\oplus} : Z\left(\left(\{h_s(m)\}, \Im_x\right), \{x(n)\}_{n=0}^{N-1}\right) \Rightarrow \{y_s(n)\}_{n=0}^{N-1}$;

FOR n=0 TO N-1 DO

$$y(n) = \sum_{t=1}^{K_h - 1} \tilde{g}(t) y(n-t) + \sum_{s=0}^{S-1} y_s\left(n - d_0^s\right);$$

АЛГОРИТМ $A_{\oplus}^{\Im}$ *- КОНЕЦ.*

8 О частных решениях задачи синтеза эффективного алгоритма

Рассмотренный метод синтеза эффективного алгоритма учитывает все аспекты рассматриваемой проблемы: и ограничения задачи, и информацию об обрабатываемом сигнале, и эффективность работы алгоритмов исходного множества. Применение этого метода, как показывает теоретическая часть работы, дает гарантированный результат – эффективный алгоритм – для любой задачи. В то же время, как любой общий подход, предложенный метод дает ряд достаточно простых частных решений, которые были и/или могут быть получены независимо, эвристически или из других соображений. Естественно, алгоритмы, подобные алгоритмам основных классов, как одно из частных решений задачи синтеза эффективного алгоритма, интерес в данном случае не вызывают. Относительно существования других частных решений в виде уже известных алгоритмов некоторые замечания были даны в п.3.2, в котором рассматривались подклассы алгоритмов модели CR. Индуцированный алгоритм, как любой другой алгоритм модели CR, также принадлежит одному из указанных подклассов. И если в литературе рассматривались некоторые алгоритмы соответствующего подкласса, следовательно, и конкретный алгоритм, подобный индуцированному алгоритму, мог быть рассмотрен. Обзору именно таких результатов посвящен этот раздел.

Первой группой алгоритмов, которая относится к частным решениям задачи синтеза индуцированного алгоритма, и которая рассматривалась в литературе по ЦОС и ЦОИ ранее, являются рекурсивные и параллельно-рекурсивные алгоритмы с КИХ, задаваемой в виде рекуррентного соотношения [24-26, 32, 38, 40, 43-47, 52, 55-61, 63]. Одной из первых работ, которая указывала на существование КИХ с хорошей «рекурсивной» реализацией была работа Л.П. Ярославского [40]. В ней указанный автор рассматривал простой в вычислительном плане алгоритм рекурсивного расчета свертки с экспоненциальной и показательной КИХ. Следом за этой работой появилась работа В.В. Сергеева [32], где уже указывался целый класс КИХ-к, которые позволяют получить простой рекурсивный алгоритм расчета свертки: показательные, синусоидальные, косинусные, полиномиальные. В этой же работе впервые указан общий (показательно-степенной) вид «элементарной» КИХ, которая позволяет строить вычислительно простой рекурсивный алгоритм. Однако следует отметить, что автор, фактически, ограничился рассмотрением КИХ, которая удовлетворяет РС второго порядка. Аналогичные работы по рекурсивной реализации свертки для КИХ синусного и косинусного вида есть и у других авторов [45, 47, 52,]. Кроме того, подобные работы до сих пор появляются в печати, открывая заново уже известные результаты.

Несколько позднее появилось сразу несколько работ, среди которых следует выделить работы M. Hatamian [46], M.F. Zakaria [63] и Н.И. Глумова [43], по синтезу вычислительно эффективных алгоритмов с КИХ полиномиального вида. Фактически, все работы неявно использовали возможность представления полиномиальной функции *K*-го порядка в виде рекуррентного соотношения (*K*+1)-го порядка. Этот факт явно был указан в работе А.В. Чернова [38], в которой был получен рекурсивный алгоритм, аналогичный рассмотренным в [43, 46, 63]. Следует, однако, отметить еще один результат, который был получен Н.И. Глумовым, В.В. Сергеевым и автором настоящей работы. Он связан с построением полиномиальных функций (КИХ-фильтров), которые допусками предельно низкое число арифметических операций в рекурсивной реализации и позволяли организовывать обработку в параллельно-рекурсивном режиме. В работах [43, 44] были указаны условия построения таких функций, приведен их явный вид. Как, однако, было позднее показано в [58] автором настоящей работы, построенные в [43, 44] полиномы имеют богатую историю и хорошо известны в теории конечных разностей под термином «обобщенные степени» или «факториальные полиномы» [2, 10, 42, 48, 51]. Несмотря на это, нестандартное использование полиномиальных функций, связанных явным образом с их рекуррентным представле-

нием, позволило получить ряд численно простых алгоритмов вычисления сверток в одномерном и двумерном случаях [24-26, 56-60].

Наконец, следует еще раз вернуться к работе А.В. Чернова [38]. В ней впервые была явно указана связь между однородным ЛРС, которое описывает КИХ, и быстрым рекурсивным алгоритмом. Обобщение этого результата на случай синтеза параллельно-рекурсивного алгоритма было сделано автором настоящей работы в [57].

Все указанные алгоритмы этой (первой) группы оказываются частными решениями задачи синтеза эффективного алгоритма, при следующих исходных данных: $Z\left(\Im_0, \{x(n)\}_{n=0}^{N-1}\right)$, $\Im_0 = \left(\{h(n)\}_{n=0}^{M-1}, (N, \varnothing)\right)$, импульсная характеристика представима в виде однородного ЛРС некоторого порядка:

- экспоненциальная КИХ – 1-го порядка,
- синусная и косинусная КИХ – 2-го порядка,
- полиномиальная КИХ – (K+1)-го порядка,
- произвольная функция с РС K-го порядка.

В этом частном решении, которое задает индуцированный алгоритм, S=2. Кроме того, в качества базового множества алгоритмов постоянной сложности, над которым строится индуцированный алгоритм, выступает множество из единственного алгоритма прямого вычисления свертки: $\{A_{DC}\}$.

Второй группой алгоритмов, которая относится к частным решениям задачи синтеза индуцированного алгоритма, и которая рассматривалась в литературе по ЦОС и ЦОИ ранее, являются алгоритмы, использующие информационную избыточность сигнала. Работы по разработке таких алгоритмов велись совместно Л.П. Ярославским, И.А. Овсиевичем и В.И. Кобером [17, 39, 53, 54, 62]. В основном, авторы ограничивались использованием операций численного дифференцирования и/или простейшего (линейного) предсказания входного сигнала. Реализация свертки выполнялась при этом с помощью алгоритмов типа БПФ, которые корректировались под «разреженный» вид исходных данных.

Алгоритмы этой (второй) группы оказываются частными решениями задачи синтеза эффективного алгоритма, при следующих исходных данных:

- $\Im_{(x)}$ - содержит информацию о том, что сигнал является кусочно-постоянным и/или линейно-постоянным.
- $\{h(n)\}_{n=0}^{M-1}$ - содержит значения «подобные шуму», которые не могут быть представлены компактно посредством ОРП или НРП.

В качества базового множества алгоритмов постоянной сложности, над которым строится индуцированный алгоритм, выступает множество алгоритмов типа БПФ $\{A_{FC}\}$, включающее в себя алгоритмы БПФ, ориентированные на работу с разреженными данными [31, 39].

Следует особо отметить, что автору настоящей работы не известны примеры построения эффективных (быстрых) алгоритмов, в которых неоднородные ЛРП использовались бы как базовый математический аппарат, подходящий для синтеза соответствующих алгоритмов. В то же время, Теоремы 7 указывает на эту возможность как на одно из очевидных частных решений задачи синтеза эффективного алгоритма.

Благодарности

Работа выполнена при поддержке:

- Российского фонда фундаментальных исследований (РФФИ), проект № 06-01-00616-а;
- Фонда содействия отечественной науке;
- Министерства образования и науки РФ, Правительства Самарской области и Американского фонда гражданских исследований и развития (CRDF Project SA-014-02) в рамках российско-американской программы «Фундаментальные исследования и высшее образование» (BRHE).

Литература

1. Агаян С.С. Успехи и проблемы быстрых ортогональных преобразований // Распознавание, классификация, прогноз. – М.: Наука, 1990. В.3. С. 146-214.
2. Андерсон Дж.А. Дискретная математика и комбинаторика: Пер. с англ. // М.: ИД »Вильямс», 2004. - 960 с.
3. Ахмед Н., Рао К. Р. Ортогональные преобразования при обработке цифровых сигналов // М.: Связь, 1980. 248 с.
4. Баранов С. Н., Домарацкий А. Н., Ласточкин Н. К., Морозов В. П. Процесс разработки программных изделий // М.: Наука. Физматлит, 2000. 176 с.
5. Блейхут Р. Быстрые алгоритмы цифровой обработки сигналов // М.: Мир, 1989. - 448 с.
6. Быстрые алгоритмы в цифровой обработке изображений. Под ред. Т.С. Хуанга // М.: Радио и связь, 1984. 224 с.
7. Вариченко Л.В., Лабунец В.Г., Раков М.А. Абстрактные алгебраические системы и цифровая обработка сигналов // Киев: Наукова думка, 1986. 248 с.
8. Виттих В.А., Сергеев В.В., Сойфер В.А. Обработка изображений в автоматизированных системах научных исследований // М.: Наука, 1982. 214 с.
9. Власенко В.А., Лапа Ю.М., Ярославский Л.П. Методы синтеза быстрых алгоритмов свёртки и спектрального анализа сигналов // М.: Наука, 1990. 180 с.
10. Гельфонд А.О. Исчисление конечных разностей. Изд. 3-е, испр // М.: Наука, 1967. 375 с.
11. Голд Б., Рэйдер Ч. Цифровая обработка сигналов // М.: Советское Радио, 1973. 367 с.
12. Гольденберг Л.М., Матюшкин Б.Д., Поляк М.Н. Справочник. Цифровая обработка сигналов // М.: Радио и связь, 1985. 312 с.
13. Грэхем Р., Кнут Д., Паташник О. Конкретная математика. Основание информатики: Пер. с англ. // М.: Мир, 1998. 703 с.
14. Журавлев Ю.И. Непараметрические задачи распознавания образов // Кибернетика, №6, 1976.
15. Журавлев Ю. И. Корректные алгебры над множествами некорректных (эвристических) алгоритмов. Часть I // Кибернетика, 1977. No. 4. С. 5–17.

16. Журавлев Ю.И. Об алгебраическом подходе к решению задач распознавания или классификации // Проблемы кибернетики. М.: Наука, 1978. № 33. С. 5–68.
17. Кобер В.И., Овсеевич И.А. Использование информационной избыточности сигналов для снижения вычислительных затрат на их обработку // Радиотехника, 1999. № 5. С. 13-16.
18. Лабунец В.Г. Алгебраическая теория сигналов и систем: Быстрое многомерное преобразование Фурье. - Свердловск: Изд-во Урал. ун-та, 1989. 196 с.
19. Лабунец В.Г. Единый подход к алгоритмам быстрых преобразований // Применение ортогональных методов при обработке сигналов и анализе систем. Свердловск: УПИ, 1980. С. 4-14.
20. Марков А.А., Нагорный Н.М. Теория алгоритмов. – М.: Наука, 1984. 432 с.
21. Математическая энциклопедия: под редакцией Виноградова И.М. // Т. 1-5, 1977.
22. Миролюбов А.А., Солдатов М.А. Линейные однородные разностные уравнения // М.: Наука, 1981. 208 с.
23. Миролюбов А.А., Солдатов М.А. Линейные неоднородные разностные уравнения // М.: Наука, 1986. 127 с.
24. Мясников В.В. Четные полиномиальные базисы для обработки изображений фильтрами с осесимметричными импульсными характеристиками // Автометрия, No 1, 1996. С. 80-87.
25. Мясников В.В. Рекурсивный алгоритм вычисления свертки изображения с неразделимым двумерным полиномиальным КИХ-фильтром // Компьютерная оптика, В.26, 2004. С. 80-82.
26. Мясников В.В. О рекурсивном вычислении свертки изображения и двумерного неразделимого КИХ-фильтра // Компьютерная оптика, 2005. Выпуск 27. С. 117-122.
27. Никольский С.М. Курс математического анализа. Т.1. 3-е изд., перераб. и доп. // М.: Наука, 1989. 464 с.
28. Нуссбаумер Г. Быстрое преобразование Фурье и алгоритмы вычисления свёрток // М.: Радио и связь, 1985. 248 с.
29. Оппенгеймер А.В., Шафер Р.В. Цифровая обработка сигналов: Пер. с англ. // М.: Связь, 1979. 416 с.
30. Рабинер Л, Гоулд Б. Теория и применение цифровой обработки сигналов: Пер. с англ. // М.: Мир, 1978. 848 стр.
31. Раудин П.В. Быстрые алгоритмы ДПФ для разреженных входных или выходных данных // 3-я ИКсийская конф. по распознаванию образов и анализу изображений (РОАИ-97). Н.Новгород, 1997. Ч.1. С. 243-247.
32. Сергеев В.В. Параллельно-рекурсивные КИХ-фильтры для обработки изображений // Компьютерная оптика. 1992. No.10-11. С.186–201.
33. Сойфер В.А., Храмов А.В. Класс спектрально-рекуррентных алгоритмов оценивания полей // Тезисы доклада 5-го Международного симпозиума по теории информации. Москва; Тбилиси, 1979. Ч.2. С. 136-138.
34. Трахтман А.М., Трахтман В.А. Основы теории дискретных сигналов на конечных интервалах // М.: Советское радио, 1975. 208 с.
35. Хемминг Р.В. Цифровые фильтры: Пер. с англ. // М.: Советское радио, 1980. 224 с.
36. Холл М. Комбинаторика: Пер. с англ. // М.: Мир, 1970, 424 с.
37. Чернов В.М. Арифметические методы синтеза быстрых алгоритмов дискретных преобразований // Диссертация на соискание ученой степени доктора физико-математических наук по специальности 05.13.17 «Теоретические основы информатики». Самара: Институт систем обработки изображений РАН, 1998. 277 с.
38. Чернов А.В. Быстрое рекурсивное вычисление значений одномерных и двумерных конечных сверток // Компьютерная оптика. 2002. № 25. С.190-197.
39. Ярославский Л.П. Введение в цифровую обработку изображений // М.: Сов. радио, 1979. 312 с.
40. Ярославский Л.П. О возможности параллельной и рекурсивной организации цифровых фильтров // Радиотехника, 1984. N 3. С.87-91.
41. Ярославский Л.П. Цифровая обработка сигналов в оптике и голографии. Введение в цифровую оптику // М.: Радио и связь, 1987. 296 с.
42. Agarwal R.P. Difference Equations and Inequality: Theory, Methods, and Applications, 2nd ed., rev. exp. // New York: Marcel Dekker, 2000. 998 p.
43. Glumov N.I., Myasnikov V.V., Sergeyev V.V. Application of polynomial bases for image processing using sliding window // SPIE. Image Processing and Computer Optics, 1994. Vol.2363. P. 40-49.
44. Glumov N.I., Myasnikov V.V., Sergeyev V.V. Parallel-Recursive Local Image Processing and Polynomial Bases // Proceedings of the Third IEEE Inrernational Conference on Electronics, Circuits, and Systems ICECS'96. Rodos, Greece, 1996. Vol.2. P.696-699.
45. Gupta A., Rao K.R. A fast recursive algorithm for the discrete sine transform // IEEE Transactions on Acoustic, Speech and Signal Processing, 1990. Vol. ASSP-38, No.3. P. 553-557.
46. Hatamian M. A real-time two-dimensional moment generating algorithm and its single chip implementation // IEEE Trans. on Acoustic, Speech and Signal Processing, 1986. Vol. ASSP-34. No.3. P.546-553.
47. Hou H.S. A fast recursive algorithm for computing the discrete cosine transform // IEEE Transactions on Acoustic, Speech and Signal Processing, 1987. Vol. ASSP-35. No.10. pp.1455-1461.
48. Jagerman D. Difference Equations with Applications to Queues // New York: Marcel Dekker, 2000. 246 p.
49. Jain A.K., Jain J.R. Partial differential equations and finite difference methods in image processing. Part II - Image restoration // IEEE Transactions on Automatic Control. Vol.AC-23, 1978. P.817-834.
50. Jain A.K. Partial differential equations and finite difference methods in image processing. Part I - Image representation // J. Optimiz. Theory and Appl. 1977. Vol. 23. P. 65-91.
51. Jordan Ch. Calculus of finite differences, 2nd ed. // New York: Chelsea Publ. Co., 1950. 652 p.
52. Kober V. A fast recursive algorithm for sliding sine transform // Electronics Letter, 2002. Vol. 38. No.25. P. 1747-1748.
53. Kober V., Yaroslavsky L.P. Use of signal redundancy for reduction of signal processing computational time // Proceedings of Seminar on Digital Image Processing in Medicine, Remote Sensing and Visualization of Information. Riga, 1992. P. 7-9.
54. Kober V., Ovseyevich I.A., Yaroslavsky L.P. Information redundancy of signals: a way to save processing time // Pattern Recognition and Image Analysis, 1993. Vol.3, No. 1. P.15-18.
55. Li B. High-Order Moment Computation of Grey-Level Images // IEEE Trans. on Image Processing, 1995. Vol. 4. No. 4. P.502-505.

56. Myasnikov V.V. A Recursive Algorithm for Computing the Convolution of an Image with a Two-Dimensional Indecomposable Polynomial FIR Filter // Pattern Recognition and Image Analysis, 2005. Vol. 15, No. 1. P. 260-263.
57. Myasnikov V.V. Methods for Designing Recursive FIR Filters // Proceedings of International Conference "Computer Vision and Graphics" (ICCVG 2004). Warsaw, Poland, 2004. Springer. P. 845-850.
58. Myasnikov V.V. Construction of Integer-Value Polynomials for Recursive Calculation of the Convolution with FIR-Filter // Тезисы 7-й Международной конференции «International Conference on Pattern Recognition and Image Analysis». Санкт-Петербург, 2004. P. 331-334.
59. Myasnikov V.V. Recursive Algorithm of Calculation the Convolution of Image and Inseparable 2-D Polynomial FIR-Filter // Тезисы 7-й Международной конференции «International Conference on Pattern Recognition and Image Analysis», Санкт-Петербург, 2004. P. 327-330.
60. Myasnikov V.V. On Recursive Computation of the Convolution of Image and 2-D Inseparable FIR-Filter // The 9th World Multi-Conference on Systemics, Cybernetics and Informatics. Orlando, Florida (USA), 2005. P.268-272.
61. Vitkus R.Y., Yaroslavsky L.P. Recursive algorithms for local adaptive linear filtration // in: Mathematical Research., Eds.: Yaroslavsky L.P., Rosenfeld A. and Wilhelmi W. Academy Verlag, Berlin, 1987. P.34-39.
62. Yaroslavsky L.P., Kober V. Redundancy of signals and transformations and computational complexity of signal processing // Proceedings of IEEE Conference on Pattern Recognition. Jerusalem, 1994. P.164-166.
63. Zakaria M.F. et al. Fast algorithm for the computation of moment invariants // Pattern Recognition, 1987. Vol.20, No.6, P. 634-643.