УДК 681.142.2(338.9)

Сравнительная эффективность различных способов разработки алгоритмов и программ

Н.Л. Малинина

Реферат

Методика сравнительной оценки эффективности различных способов создания математического и программного обеспечения (МПО) основана на структурных свойствах сетевых моделей и граф-схем алгоритмов, то есть анализе числа взаимно-независимых путей в графе. Стоимость разработки МПО рассматривается как сумма стоимостей отдельных этапов общей технологической схемы. Анализ работ по каждому этапу этой схемы позволяет вывести приближенные формулы стоимостей разработки по каждому этапу в отдельности для различных технологий программирования. На основании полученных формул выделяются области целесообразного применения различных способов создания МПО в зависимости от размеров алгоритма. Достоверность аналитических зависимостей разработанной методики доказывается совпадением их с экспериментальными данными М.Х. Холстеда и других авторов.

Введение

Далеко не секрет, что к настоящему времени стоимость разработки математического и программного обеспечения (МПО) существенно возросла и составляет до 85% от общей стоимости комплекса ЭВМ и МПО. Тенденция роста стоимости МПО не снижается, и с этим фактором разработчики МПО во всех странах сталкиваются повседневно. В большинстве случаев эти трудности пытаются обойти, приспосабливая уже разработанные крупные программные комплексы к новым задачам. Хотя гораздо более серьезной проблемой представляется разработка логической структуры для объединения этих модулей, а также процедур проверки их работоспособности.

Целью настоящей работы является разработка методики, позволяющей достаточно корректно оценивать трудозатраты на построение МПО различными способами.

1. Структура процесса разработки МПО

Проектирование сложных технических систем и комплексов в областях автоматизированного проектирования и конструирования, автоматизированного управления связано с решением очень сложных задач программирования. Все существующие способы разработки и/или построения математического и программного обеспечения (МПО) в принципе можно разбить на ряд последовательных этапов и свести к единой технологической схеме (рис.1).

Очевидно, что первый и второй этапы в принципе не могут быть автоматизированы, поскольку опираются на опыт, эрудицию, знания и интуицию разработчика, требуют

непосредственного труда исследователя и нередко граничат с искусством. В работах по третьему этапу при разработке расчетных модулей (поиск методов расчета и оптимизации, вывод расчетных формул, выбор расчетных или граничных условий) возможно, могут быть автоматизированы лишь способы поиска информации. Разработка блок-схемы или произвольного вычислительного алгоритма в графическом, матричном или табличном виде пока еще повсеместно выполняется вручную, хотя так же могут быть в достаточной степени автоматизированы [1,2].

Рис. 1. Единая технологическая схема разработки МПО

На пятом этапе возможность использования автоматизации частично реализуется за счет применения стандартных программ и отдельных технологий программирования [3,4,5], однако тоже только частично.

Два последних этапа являются самыми сложными и трудоемкими. Причиной этого является их совместное выполнение в практике разработки алгоритмов и программ, когда приходится одновременно решать три группы вопросов: упорядочение всех операторов алгоритма, их логическую и информационную увязку между собой, а также информационное снабжение программы из внешних источников и баз данных. Раздельное выполнение этих этапов и применение методов синтеза сетевых моделей и вычислительных алгоритмов [1,2] даст возможность автоматизировать процесс разработки нормальных алгоритмов, разделить вопросы упорядочения, а так же логической и информационной увязки.

Могут быть рассмотрены четыре типовых случая разработки МПО, приведенные в таблице I:

Таблица I

Степень автоматизации / Этапы разработки МПО	1	2	3	4	5	6	7
Неавтоматизированные способы разработки	-	-	-	-	-	-	-
Применение стандартных программ в отдельных стандартных модулях (малая автоматизиция)	-	-	-	-	+	-	-
Применение автоматизированного синтеза алгоритмов и программ при разработке МПО в целом, но только в интересах первоначального синтеза	-	-	-	-	-	+	+
Применение автоматизированного синтеза алгоритмов и программ, стандартных программ при разработке, как отдельных модулей, так и МПО в целом не только в интересах первоначального синтеза, но и для создания адаптивного МПО, допускающего реконфигурацию в диалоговом режиме.	-	-	-	+	+	++	++

Примечание: «-» - автоматизация не применяется; «+» - автоматизация применяется частично; «+ +» - автоматизация применяется в максимальном объеме.

Следуя единой технологической схеме, можно также рассмотреть четыре наиболее часто встречающихся случая разработки МПО (рис.2).

Рис. 2. Принципиальные схемы разработки МПО

2. Критерий эффективности некоторого способа разработки МПО

Эффективность любого (*i*-го) способа разработки МПО может быть оценена его трудоемкостью или стоимостью разработки. Очевидно, что трудозатраты поэтапной реализации всех семи этапов разработки - суть сумма трудозатрат реализаций этих семи этапов:

$$T_i = \sum_{l=1}^{l=7} T_{li} = T_{1i} + T_{2i} + T_{3i} + T_{4i} + T_{5i} + T_{6i} + T_{7i} \qquad (2.1)$$

$$C_i = = \sum_{l=1}^{l=7} C_{li} = C_{1i} + C_{2i} + C_{3i} + C_{4i} + C_{5i} + C_{6i} + C_{7i} \qquad (2.2)$$

где: $T_{1i}(C_{1i})$ – трудозатраты (стоимость) при изучении объекта проектирования, моделирования, управления;

$T_{2i}(C_{2i})$ – трудозатраты (стоимость) на декомпозицию объекта;

$T_{3i}(C_{3i})$ – трудозатраты (стоимость) на разработку расчетных модулей, блоков модели, операторов;

$T_{4i}(C_{4i})$ – трудозатраты (стоимость) на построение блок-схемы произвольного алгоритма;

$T_{5i}(C_{5i})$ – трудозатраты (стоимость) на программирование расчетных модулей, блоков модели, операторов;

$T_{6i}(C_{6i})$ – трудозатраты (стоимость) на разработку граф-схемы нормального алгоритма;

$T_{7i}(C_{7i})$ – трудозатраты (стоимость) на разработку управляющей программы.

В дальнейшем будут применяться в основном только формулы для определения трудозатрат.

3. Формулы для оценки трудозатрат и стоимости при разработке МПО неавтоматизированным способом

3.1.Для первого этапа следует принять:

$$T_1 = \mu_1 * n \qquad (3.1)$$

$$C_1 = s_1 * \mu_1 * n$$

где: n – длина итогового алгоритма в модулях или блоках;

μ_1 – коэффициент пропорциональности, характеризующий удельные трудозатраты, приходящиеся на первом этапе в среднем на один модуль или блок итогового алгоритма.

s_1 – стоимость одного часа трудозатрат на первом этапе.

Условимся здесь и далее считать, что стоимость одного часа трудозатрат или удельная стоимость трудозатрат определяется с учетом расходов на амортизацию оборудования, всех накладных расходов и т.д.

3.2.Для второго этапа по аналогии:

$$T_2 = \mu_2 * n \qquad (3.2)$$

$$C_2 = s_2 * \mu_2 * n$$

где: μ_2 – средние удельные трудозатраты на анализ и декомпозицию объекта, приходящиеся на один модуль или блок итогового алгоритма.

s_2 – здесь и далее – удельная стоимость трудозатрат на соответствующем этапе.

3.3.На третьем этапе правильнее перейти к другой схеме оценки трудозатрат, т.е.:

$$T_3 = \sum_1^n \tau_{3k} \qquad k = 1,2,...,n \qquad (3.3)$$

$$C_3 = s_3 * \sum_1^n \tau_{3k}$$

где: τ_{3k} – трудозатраты на разработку математического описания k-ого модуля.

3.4.Четвертый этап

Трудозатраты на разработку блок-схемы модели или блок-схемы произвольного алгоритма, состоящего из n модулей, эквивалентны трудозатратам на разработку матрицы смежности модулей произвольного алгоритма или таблицы отношений предшествования или следования [1].

По аналогии запишем:

$$T_4 = \mu_4 * n \qquad (3.4)$$

$$C_4 = s_4 * \mu_4 * n$$

где: μ_4 – коэффициент пропорциональности, характеризующий трудозатраты, приходящиеся на один модуль или блок на четвертом этапе.

Таким образом, получим, что: $\sum_1^4 T_l = (\mu_1 + \mu_2 + \mu_3) * n + \sum_1^n \tau_{3k}$ (3.5)

или: $\sum_1^4 C_l = (s_1 * \mu_1 + s_2 * \mu_2 + s_4 * \mu_4) * n + s_3 * \sum_1^n \tau_{3k}$

Поскольку работа по изучению объекта и его декомпозиции, а также работа по разработке блок-схемы являются наиболее сложными и наиболее высокооплачиваемыми, можно полагать, что: $s_1 = s_2 = s_4 = s$

Тогда: $\sum_1^4 C_l = s * (\mu_1 + \mu_2 + \mu_3) * n + s_3 * \sum_1^n \tau_{3k}$

3.5.Трудозатраты на пятом этапе можно рассматривать по аналогии с третьим этапом:

$$T_5 = \sum_1^n \tau_{5k} \qquad k = 1,2,...,n \qquad (3.6)$$

$$C_5 = s_5 * \sum_{1}^{n} \tau_{5k}$$

где: τ_{5k} – трудозатраты на разработку программы *k*-ого модуля.

3.6.Шестой и седьмой этапы

Они являются наиболее сложными, но на этих этапах возможность реализации автоматизированного построения алгоритмов и программ становится максимальной, поскольку именно на этих этапах появляется вероятность формализованного описания такого процесса. Поэтому для анализа трудозатрат и стоимости работ на этих этапах необходимо провести анализ технологии работы при построении алгоритмов и программ.

Блок-схема произвольного алгоритма представляет собой сопряженную сетевую модель (или ориентированный вершинный граф), как правило, с циклами, а в общем случае и с контурами. Преобразование блок-схемы произвольного алгоритма в граф-схему нормального алгоритма является необходимым условием разработки программы (придание свойства эффективной рекурсивности) и равносильно преобразованию сопряженной сетевой модели в обыкновенную сетевую модель [1] тем или иным способом, но чаще всего вручную. При выполнении этой работы любым нерегулярным способом такое преобразование сводится к перебору, просмотру всех взаимно-независимых путей в графе, ведущих от начальной вершины к конечной, их упорядочению и информационно-логической увязке между собой. Поэтому для оценки трудоемкости такой работы необходимо иметь оценку числа путей в графе и оценку числа пересечений таких путей друг с другом. Оценки эти достаточно сложны и имеют точное решение только в конкретных случаях. Однако, опираясь на некоторые общие свойства графов, можно получить приближенные оценки этих величин.

Одной из характеристик сложности структуры связного графа является его цикломатическое число:

$$\nu = m - n + 1 \qquad (3.7)$$

где: m – число дуг ориентированного графа;

n – число вершин ориентированного графа или число модулей блок-схемы.

Нижняя граница числа взаимно-независимых путей в графе определяется формулой В.П. Куляпина: $\sigma_{\min} = \nu + 1$ (3.8)

При этом действительное число взаимно-независимых путей, как правило, больше.

Рассмотрим пример на рис.3, где представлен граф H_q, который может рассматриваться как блок-схема.

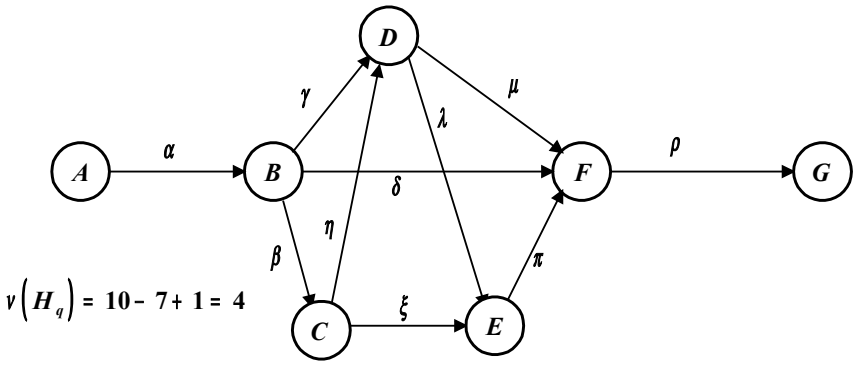

Рис. 3. Граф H_q.

Матрица смежности вершин этого графа F_{q1} представлена на рис.4.

Цикломатическое число графа H_q: $\nu\left(H_q\right) = 10 - 7 + 1 = 4$.

Нижняя граница числа взаимно-независимых путей: $\sigma_{\min} = 4 + 1 = 5$.

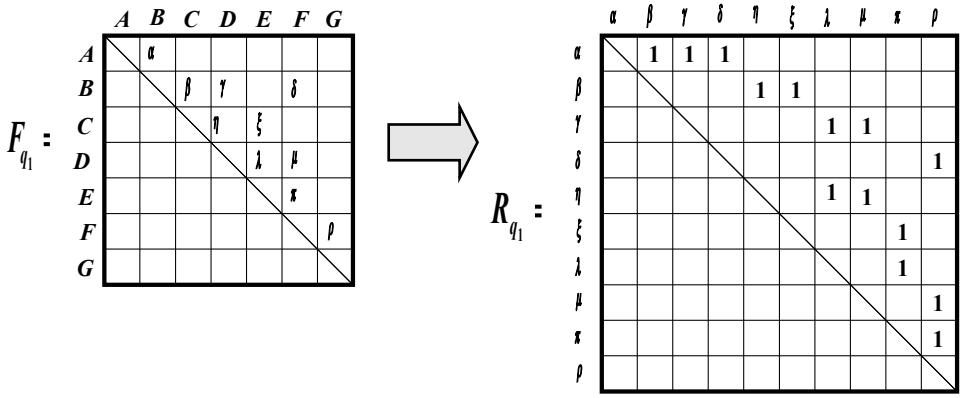

Рис. 4. Матрица смежности вершин графа F_{q1}.

Однако действительное число взаимно-независимых путей больше. Выяснить это можно путем несложных преобразований графа (прямого конвертирования). Построим матрицу R_{q1} – смежности дуг графа H_q (рис.4). Ряды этой матрицы соответствуют дугам графа H_q, а элементы $r_{ij} = 1$ – бинарным отношениям следования: $q_i < q_j$.

Будем рассматривать новую матрицу как матрицу смежности вершин нового графа H_{k2} (рис.5). В этом графе каждая вершина соответствует дуге и паре вершин исходного графа H_q. При таком преобразовании графа система бинарных отношений следования на исходном множестве элементов не изменится. Соответственно не могло измениться и число взаимно-независимых путей.

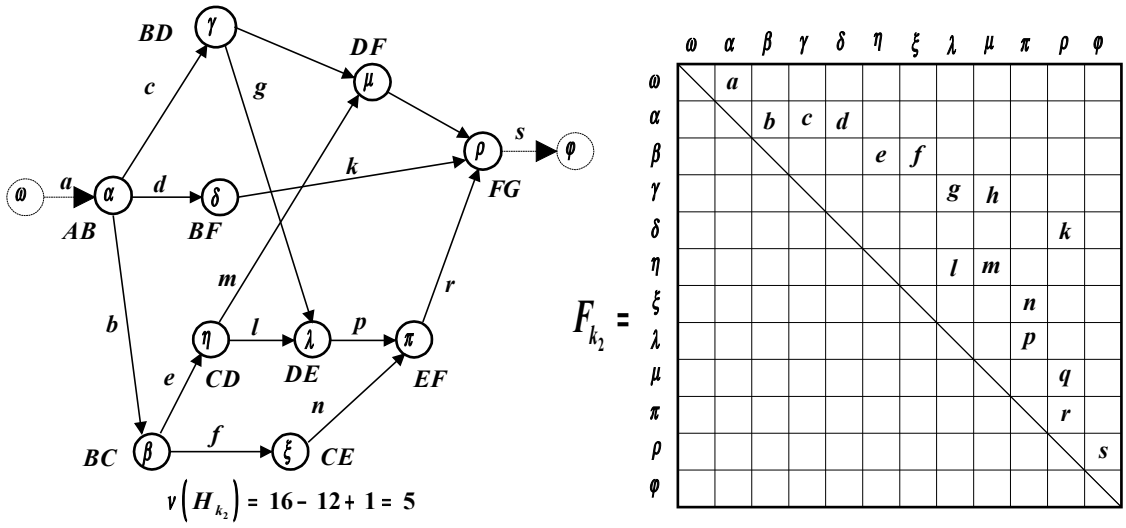

Рис. 5. Матрица смежности вершин графа H_{k2}.

Для того чтобы новый граф имел бы, как и исходный граф, начальную и конечную дуги, добавим к нему дуги a и s, а также вершины ω и φ, что не изменит его цикломатического числа. Матрица смежности вершин F_{k2} графа H_{k2} получит вид, представленный на рис.5.

Цикломатическое число нового графа H_{k2}: $\nu\left(H_{k2}\right) = 16 - 12 + 1 = 5$. Следовательно, число взаимно-независимых путей в исходном графе будет не менее шести. Действительно, исходный граф имеет следующие взаимно-независимые пути, совпадающие с взаимно-независимыми путями в конвертированном графе.

Таблица 2

№ п/п	Исходный граф Путь по дугам		Конвертированный граф Путь по вершинам	
1	$\alpha\gamma\mu\rho$	ω	$\alpha\gamma\mu\rho$	φ
2	$\alpha\gamma\lambda\pi\rho$	ω	$\alpha\gamma\lambda\pi\rho$	φ
3	$\alpha\delta\rho$	ω	$\alpha\delta\rho$	φ
4	$\alpha\beta\eta\mu\rho$	ω	$\alpha\beta\eta\mu\rho$	φ
5	$\alpha\beta\eta\chi\pi\rho$	ω	$\alpha\beta\eta\chi\pi\rho$	φ
6	$\alpha\beta\xi\pi\rho$	ω	$\alpha\beta\xi\pi\rho$	φ

Этот простейший пример показывает, что выявление действительного числа взаимно-независимых путей в графе без его конвертирования бывает весьма затруднительным. В сложных блок-схемах число взаимно-независимых путей может оказаться существенно больше цикломатического числа.

Итак, нижняя граница числа взаимно-независимых путей в блок-схеме определяется цикломатическим числом. Действительное число взаимно-независимых путей может быть определено путем последовательного многократного конвертирования блок-схемы. Введем функцию:

$$P_{i_\varphi}(\vartheta_\varphi) = \sum_{i}^{i_\varphi} P_{\chi_\varphi}(\vartheta_\varphi)$$

где: φ - номер пути в графе от начальной вершины к конечной,

$i_\varphi = 1,2,3\ldots,\xi_\varphi$ - порядковые номера вершины $\vartheta_\varphi \in \nu_\varphi$,

ν_φ - множество всех вершин, входящих в путь φ,

$\chi_\varphi = 1,2,\ldots,i_\varphi$ - порядковые номера вершин φ-го пути в графе от начальной вершины до вершины с номером i_φ,

$P_{\chi_\varphi}(\vartheta_\varphi)$ - сумма полустепеней захода и исхода вершины ϑ_φ.

Конвертирование исходного графа для определения действительного числа путей в нем считается законченным, когда функция $P_{i_\varphi}(\vartheta_\varphi)$ вдоль любого φ-го пути на интервале $(1 \div i_\varphi) \to (1 \div \xi_\varphi)$ становится по мере приближения $(\xi_\varphi - i_\varphi)$ к нулю вначале неубывающей, а затем убывающей. Граф в этом случае называется голономным, а действительное число путей в исходном графе будет равно:

$$\sigma_{исх.} = \nu_{голон.} + 1$$

где: $\nu_{голон.}$ - цикломатическое число голономного графа

Для целей настоящей работы достаточно ограничиться знанием нижней границы цикломатического числа. Так как преобразование блок-схемы в нормальный алгоритм и программу включает в себя трудозатраты на упорядочение блок-схемы, просмотр всех взаимно-независимых путей и их алгоритмическую и информационно логическую увязку, то общие трудозатраты на такую работу можно записать как:

$$T_{рс} = T_{упор} + \Delta T_{рс} \qquad (3.9)$$

где: $T_{упор}$ – трудозатраты на упорядочение блок-схемы;

$\Delta T_{рс}$ – трудозатраты на просмотр всех взаимно-независимых путей и их алгоритмическую и информационно-логическую увязку.

Рассмотрим по порядку.

$T_{упор}$ – трудозатраты на упорядочение. Существующие на практике методы упорядочения блок-схем, например, метод Де-Мукрона, не дают однозначного упорядочения, в результате чего исследователь оказывается перед необходимостью выбора того или иного варианта упорядочения, причем число таких вариантов пропорционально числу модулей в блок-схеме. Окончательный порядок достигается обычно путем ряда итераций. Обозначим число вариантов, которые надо

просмотреть и сравнить при упорядочении блок-схемы: $\chi_{упор}$. В идеальном случае $\chi_{упор} = 1$. В случае глобального перебора $\chi_{упор} = n!$. Практика показывает, что в большинстве реальных случаев имеет место примерно линейная зависимость:

$$\chi_{упор} \cong n * \alpha \qquad (3.10)$$

где: α – коэффициент пропорциональности.

Тогда: $T_{упор} = \chi_{упор} * \mu_{упор}$ (3.11)

где: $\mu_{упор}$ – средние трудозатраты на упорядочение одного варианта вручную.

Вполне очевидно, что $\mu_{упор}$ зависит от n. Из практики известно, что в качестве нижней границы можно принять следующую зависимость: $\mu_{упор} = \beta * n$ (3.12)

Тогда: $T_{упор} \cong \alpha * \beta * n^2$ (3.13)

Оценить значения коэффициентов α и β в отдельности невозможно из-за отсутствия статистических данных, однако некоторые сведения, касающиеся произведения $\alpha * \beta$ можно узнать из практики. Так, для упорядочения блок-схемы из 10 модулей с учетом выбора варианта упорядочения может потребоваться 1 или 2 часа, а для упорядочения блок-схемы из 50 модулей с учетом выбора варианта упорядочения может потребоваться несколько дней, т.е. до 40 или 50 рабочих часов и даже больше. Эти цифры позволяют дать следующую ориентировочную оценку величине $\alpha * \beta$:

$$\alpha * \beta = \frac{T_{упор}}{n^2}$$

или: $\alpha * \beta \cong 0,01 \div 0,02$ (3.14)

Поэтому в качестве ориентировочной оценки для $T_{упор}$. можно принять:

$$T_{упор} \cong (0,01 \div 0,02) * n^2 \qquad (3.15)$$

Рассмотрим величину $\Delta T_{рс}$. Трудозатраты на просмотр всех взаимно-независимых путей в блок-схеме и их алгоритмическую и информационно-логическую увязку можно рассматривать как сумму трудозатрат по всем взаимно-независимым путям на их просмотр и увязку по всем пересечениям каждого из путей с другими. Строго говоря, функция $\Delta T_{рс}$ является неаддитивной. По мере просмотра все чаще встречаются уже проверенные связи и уже просмотренные участки. И, хотя любая связь, сколько бы раз она не встречалась, должна быть проверена, но после ряда проверок она перестает проверяться. Дефицит времени неминуемо приводит к тому, что часть проверок не осуществляется. Часть эта становится тем больше, чем сложнее блок-схема. Поэтому при суммировании трудозатрат следует ввести поправку на реальную неаддитивность функции:

$$\Delta T_{pc} = \beta_{pc} * \sum_{1}^{\sigma} \Theta_{\varphi} \qquad \varphi = 1,2,...,\sigma \tag{3.16}$$

где: Θ_{φ} – трудозатраты на поиск, просмотр, анализ φ -ого пути и его алгоритмическую и информационно-логическую увязку с другими путями;

β_{pc} – средний коэффициент пропорциональности.

Строго говоря, выражение (3.15) следует записать в виде:

$$\Delta T_{pc} = \sum_{1}^{\sigma} \beta_{pc}(\varphi) * \theta_{\varphi} \tag{3.17}$$

Причем $\beta_{pc}(\varphi)$ может в зависимости от φ меняться от 1 до величины достаточно малой (но не равной нулю). Однако в целях упрощения задачи будем рассматривать ΔT_{pc} в виде (3.16), а позже определим функцию $\beta_{pc}(\varphi)$.

Трудозатраты Θ_{φ} определяются сложностью φ -ого пути, которая, в свою очередь, определяется числом связей типа "вершина-дуга" и "дуга-вершина", встречающихся вдоль каждого пути с учетом всех боковых ответвлений от пути и включений от других путей. Общее число связей равно удвоенному числу дуг в блок-схеме, т.е. $2m$.

Каждая из них должна быть подвергнута многократной проверке, поскольку на выходе из любой вершины в общем случае реализуются переходы к различным последующим вершинам (модулям, блокам), условия перехода при этом, как правило, различны. Вследствие этого каждый выход из модуля (вершины) блок-схемы при проверке любого из путей должен перепроверяться и согласовываться со всеми остальными выходами. Аналогичное правило действует и для входов. В реальной жизни ради экономии времени такой многократный просмотр не делается, хотя потом на выявление и устранение ошибок при отладке и сопровождении приходится тратить не меньшее, если не большее время.

Оценим трудозатраты на анализ φ -ого пути следующим образом:

$$\Theta_{\varphi} = \rho_{\varphi} * \mu^{(1)}_{\rho_{cp}} \tag{3.18}$$

где: ρ_{φ} – характеристика сложности φ -ого пути, равная числу связей типа "вершина-дуга" и "дуга-вершина", встречающихся вдоль φ -ого пути с учетом боковых связей.

$\mu^{(1)}_{\rho_{cp}}$ – средние трудозатраты на однократную проверку любой связи из числа ρ_{φ} на φ -ом пути с учетом трудозатрат на ее алгоритмическое и программное включение.

По определению:

$$\rho_{\varphi} = \sum_{1}^{l_{\varphi}+1} \left(|P_{-}| + P_{+} \right)_{q_{\varphi}}, \qquad q_{\varphi} = 1,2,...,\left(l_{\varphi+1} \right) \tag{3.19}$$

Здесь: $\left(\left|P_{-}\right|+P_{+}\right)$ – сумма абсолютных значений полустепеней захода и исхода в вершине q_{φ}, принадлежащей φ -му пути длиной l_{φ} дуг ($\left(l_{\varphi+1}\right)$ вершин).

Введем допущение, что: $\mu^{(1)}_{\rho_{\varphi}} = \mu^{(1)}_{\rho_{cp}}$ по всем путям в блок-схеме

Тогда:

$$\Delta T_{pc} = \beta_{cp} * \mu^{(1)}_{\rho_{cp}} * \sum_{1}^{\sigma} \sum_{1}^{l\varphi+1} \left(\left|P_{-}\right|+P_{+}\right)_{q_{\varphi}} \tag{3.20}$$

Введем понятие средней сложности пути: $$\rho_{cp} = \frac{\sum_{1}^{\sigma} \sum_{1}^{l_{\varphi}+1} \left(\left|P_{-}\right|+P_{+}\right)}{\sigma} \tag{3.21}$$

Тогда: $\Delta T_{p.c.} = \beta_{cp} * \mu^{(1)}_{\rho_{cp}} * \rho_{cp} * \sigma$ (3.22)

Оценим значения σ и ρ_{cp}. Выше было отмечено, что: $\sigma_{\min} = \nu + 1$

В свою очередь: $\nu = m - n + 1$

где: m – число дуг ориентированного графа или число ненулевых элементов матрицы смежности блок-схемы.

Вполне очевидно, что при $\nu = 0$ и $m_{\min} = n - 1$ граф представляет собой простую цепь.

Для матрицы без контуров и петель: $m_{\max} = \frac{n^2 - n}{2}$

Введем коэффициент плотности заполнения матрицы смежности:

$$r_n = \frac{m_r}{m_{\max}} \tag{3.23}$$

Очевидно, что в реальных случаях будем иметь некоторое промежуточное значение m_r:

$$m_{\min} \le m_r \le m_{\max} \tag{3.24}$$

Тогда для произвольного графа без контуров и петель:

$$m_r = r * m_{\max} \tag{3.25}$$

$$r_{\max} = 1 \quad \text{и} \quad r_{\min} = \frac{m_{\min}}{m_{\max}} = \frac{2}{n} \tag{3.26}$$

То есть: $r = \frac{2}{n}, \ldots, 1$

Цикломатическое число блок-схемы в зависимости от r будет:

$$\nu_r = r * \frac{n^2 - n}{2} - n + 1 \tag{3.27}$$

или: $\nu_r = (n-1) * \left(\frac{r * n}{2} - 1\right)$

Тогда: $\sigma_{\min} = (n-1) * \left(\frac{r * n}{2} - 1 \right) + 1$ (3.28)

Представим $\sigma_{\min}$ в виде функции: $\sigma_{\min} = n^{\omega(n,r)_{\min}}$ (3.29)

Тогда: $\omega(r,n)_{\min} = \frac{\ln(\sigma_{\min})}{\ln(n)}$ (3.30)

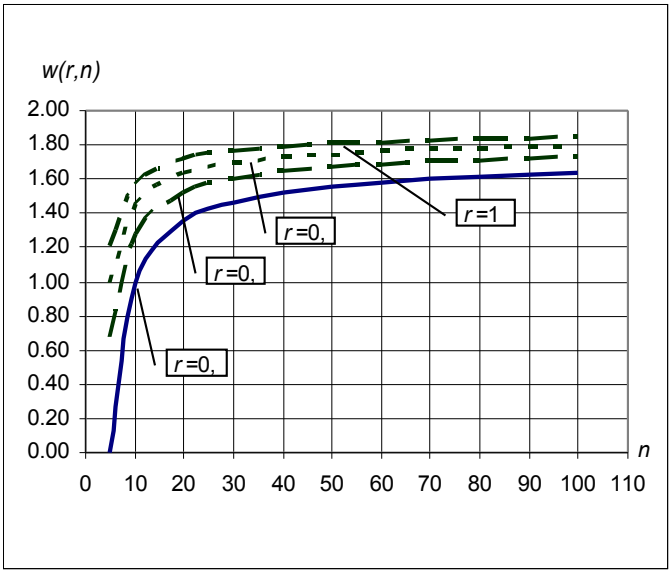

Рис. 6. Зависимость $\omega(r,n)_{\min}$

Функция $\omega(r,n)_{\min}$ для ряда значений n и r приведена на рис.6. Итак:

$\Delta T_{p.c.} \; \beta_{ср} * \mu^{(1)}_{\rho_{ср}} * \rho_{ср} * n^{\omega(r,n)_{\min}}$ (3.31)

Оценим величину $\rho_{ср}$ для некоторых типичных случаев.

Начнем со случая, когда $\rho_{ср}$ заведомо близко к минимальному значению. Для этого рассмотрим граф с большим числом параллельных путей, в каждом из которых начальная и конечная вершины простые, а остальные – простейшие. Граф такого вида, а также матрица смежности вершин представлены на рис.7 и 8.

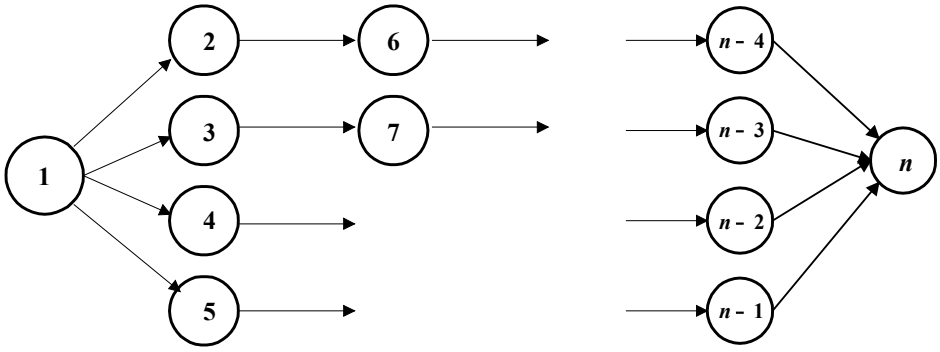

Рис.7. Граф с большим числом параллельных путей

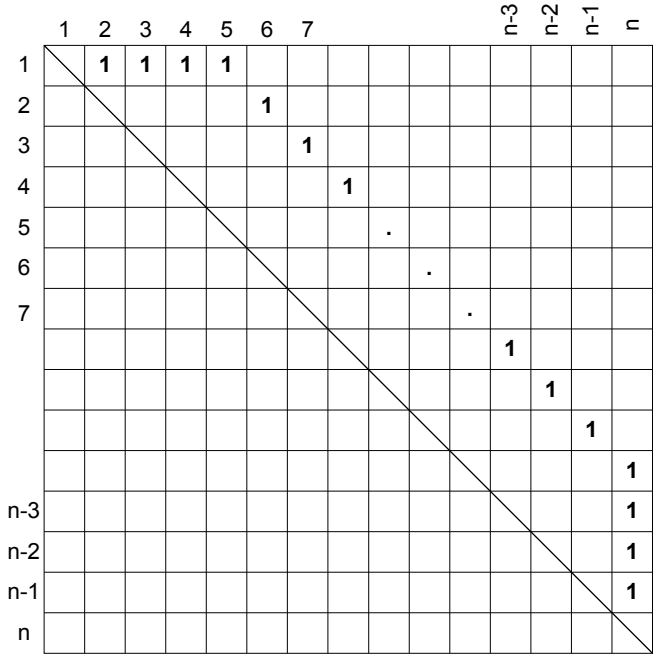

Рис 8. Матрица смежности вершин для графа с большим числом параллельных путей

Для каждого из путей в таком графе: $\rho_{ср} = 2 * \left(\sigma + \frac{n-2}{\sigma} \right)$ (3.32)

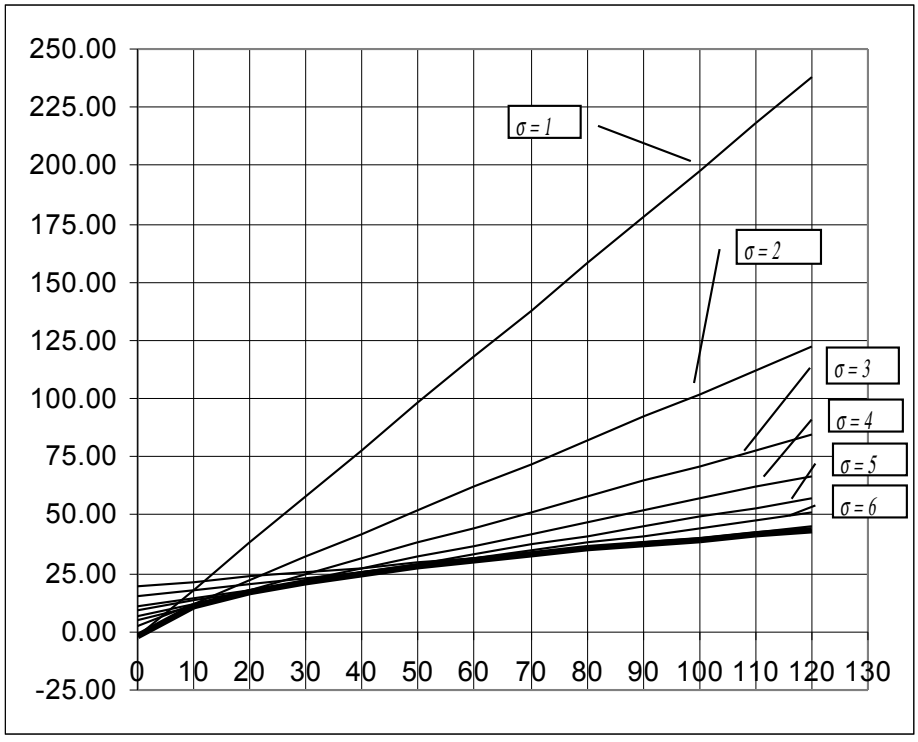

Рис. 9. Результаты расчетов $\rho_{ср}$ при различных σ и n для графов с большим числом параллельных путей.

Можно показать, что для графов такого вида при $10 < n < 50$: $\rho_{ср} \geq (n-3)^{0,77} + 8$ (3.33)

или: $\rho_{ср} * \sigma_{\min} \geq \left((n-3)^{0,77} + 8 \right) * \sigma_{\min}$ (3.34)

Но для графов такого вида: $\sigma_{\min} = \sigma = \nu + 1 = m^* - n + 2$ (3.35)

В свою очередь: $m^* = m_{\max} * r$ (3.36)

где: $r = r_{\min} \div r_{\max}; \quad r_{\min} = \frac{2}{n}; \quad r_{\max} = \frac{4(n-2)}{n(n-1)} \cong \frac{4}{n}$ (3.37)

Тогда для графов рассматриваемого вида:

$$m^*_{\min} = m_{\max} * r_{\min} = \frac{n^2 - n}{2} * \frac{2}{n} = n - 1$$

Или: $m^*_{\max} = m_{\max} * r_{\max} = \frac{n^2 - n}{2} * \frac{4(n-2)}{n(n-1)} = 2 * (n-2);$ (3.38)

Следовательно: $\sigma_{\min} = 1; \quad \sigma_{\max} = n - 2$ (3.39)

И любое промежуточное значение может быть выражено следующей формулой:

$$\sigma = m_{\max} * r - n + 2 = \frac{n^2 - n}{2} * r - n + 2$$ (3.40)

или:

$$\sigma = n^2 * \frac{r}{2} - n * \left(\frac{r}{2} + 1\right) + 2$$

Таким образом, имеем:

При: $r_{\min} = \frac{2}{n}$ $\rho_{ср} * \sigma_{\min} \geq (n-3)^{0,77} + 8$

При: $r = \frac{3}{n}$ $\rho_{ср} * \sigma_{\min} \geq (n-3)^{0,77} + 8 * \frac{n+1}{2}$ (3.41)

При: $r_{\max} = \frac{4}{n}$ $\rho_{ср} * \sigma_{\min} \geq (n-3)^{0,77} + 8 * (n-2)$

Можно показать, что для графов такого вида величина $\rho_{ср} * \sigma_{\min}$ может быть оценена по формуле:

$$\rho_{ср} * \sigma_{\min} \geq (n-3)^{0,77} + 8 * \left(1 + \frac{n-1}{2} * (r * n - 2)\right)$$ (3.42)

где: $r = \frac{2}{n}, ..., \frac{4}{n};$

Можно представить ρ_{σ} в более простом виде функциональной зависимости: $\rho_{ср} = n^{\omega}$ (3.43)

Тогда:

При: $r_{\min} = \frac{2}{n}$ $\omega \geq \frac{\ln[(n-3)^{0,77} + 8]}{\ln(n)}$

При: $r = \frac{3}{n}$ $\omega \approx \frac{\ln\{[(n-3)^{0,77}+8]*(\frac{n+1}{2})\}}{\ln(n)}$ (3.44)

При: $r_{\max} = \frac{4}{n}$ $\omega \approx \frac{\ln\{[(n-3)^{0,77}+8]*(n-2)\}}{\ln(n)}$

Функция $\omega(n,r)$ приведена на рис. 10.

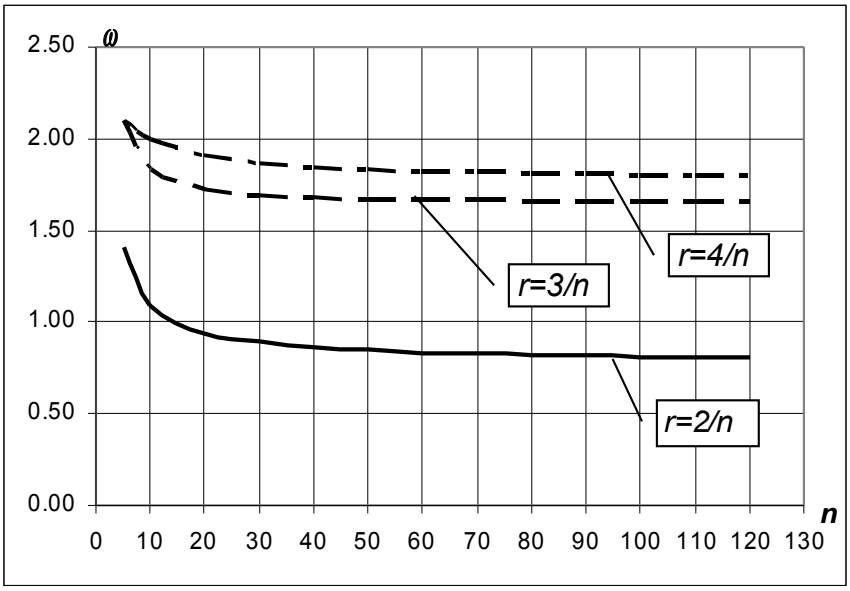

Рис. 10. Функция $\omega(n,r)$ для графов с большим числом параллельных путей.

Ясно видно, что даже для такого простого графа зависимость $\rho * \sigma$ от n близка к квадратичной при $r \approx \frac{3}{n}$. В качестве нижней границы для оценочных расчетов для графов такого вида можно принять: $\rho_{cp} * \sigma \cong n^{1,7}$ (3.45)

Таким образом, для графов, подобных, представленным на рис.5, нижняя граница трудозатрат может быть выражена формулой: $\Delta T_{pc} \approx \beta_{pc} * \mu^{(1)}_{\rho_{cp}} * n^{1,7}$ (3.46)

Для еще более сложных графов получить простую аналитическую зависимость для величины: $\sum_{\varphi=1}^{\varphi=\sigma} \sum_{1}^{l_\varphi+1} \left(|P_-| + P_+\right)_{q_\varphi}$ по-видимому, нельзя. Ограничимся лишь гарантированной оценкой нижней границы этой величины.

Рассмотрим граф на рис.3. Для него: $\sum_{1}^{\sigma} \sum_{1}^{l\varphi} \left(|P_-| + P_+\right)_{q_\varphi} = 94$ (3.47)

Число вершин, определяющих число путей в графе, равно пяти.

Тогда: $\rho_{cp} * \sigma = n^{2,82}$ (3.48)

Получим: $\rho_{ср} = \dfrac{n^{2,82}}{n^{\omega(r,n)_{\min}}}$ (3.49)

В этом случае: $\Delta T_{рс} = \beta_{рс} * \mu^{(1)}_{\rho ср} * n^{2,82}$ (3.50)

Таким образом, для не очень сложного графа мы получили почти кубическую зависимость трудозатрат от числа вершин в графе (или числа модулей алгоритма).

Рассмотрим граф, приведенный на рис.11.

	1	2	3	4	5	6
1		1	1	1	1	1
2			1	1	1	1
3				1	1	1
4					1	1
5						1
6						

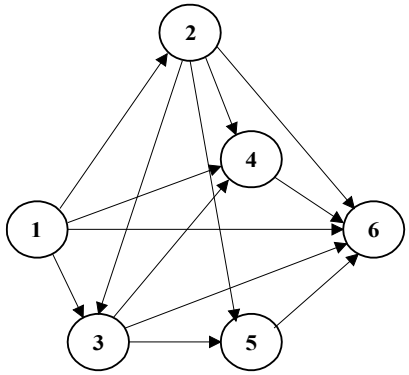

Рис. 11.

В этом графе:

Таблица 2.

φ	Пути в графе	ρ_{φ}	φ	Пути в графе	ρ_{φ}
1	1 2 3 4 5 6	30	9	1 3 4 5 6	25
2	1 2 3 4 6	25	10	1 3 4 6	20
3	1 2 3 5 6	25	11	1 3 5 6	20
4	1 2 3 6	20	12	1 3 6	15
5	1 2 4 5 6	25	13	1 4 5 6	20
6	1 2 4 6	20	14	1 4 6	15
7	1 2 5 6	20	15	1 5 6	15
8	1 2 6	15	16	1 6	10
			e		320

$m = 15$; $n = 6$; $\nu = 10$; $\sigma = 16$; $\sum_{1}^{\sigma=16} \sum_{1}^{l_{\varphi+1}} \left(|P_-| + P_+\right)_{q_\varphi} = 320$ $\rho_{ср} = 20$; $\rho_{ср} * \sigma = n^{3,22}$

Рассмотрим полный граф с числом вершин $n = 7$:

	1	2	3	4	5	6	7
1		1	1	1	1	1	1
2			1	1	1	1	1
3				1	1	1	1
4					1	1	1
5						1	1
6							1
7							

$v = 21 - 7 + 1 = 15$

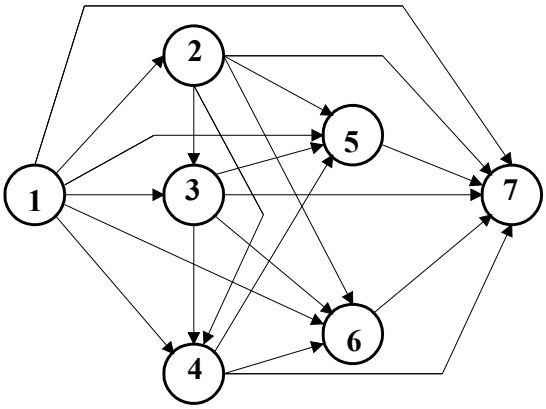

Рис.12. Полный граф с числом вершин $n = 7$.

Сделаем некоторые выводы из полученных оценок:

1. Для простейших графов (рис.5):

$$\Delta T_{PC} \mid \beta_{PC} * \mu^{(1)}_{\rho_{CP}} * n^{1,7} \tag{3.51}$$

2. Для произвольного графа с малым числом вершин (рис.3):

$$\Delta T_{PC} \mid \beta_{PC} * \mu^{(1)}_{\rho_{CP}} * n^{2,82} \tag{3.52}$$

3. Для полного графа с небольшим числом вершин (рис.11) $n = 6$:

$$\Delta T_{PC} \mid \beta_{PC} * \mu^{(1)}_{\rho_{CP}} * n^{3,22} \tag{3.53}$$

4. Для полного графа с $n = 7$ (рис.12):

$$\Delta T_{p.c.} = \beta_{pc} * \mu^{(1)}_{\rho_{cp}} * n^{2,88} \tag{3.54}$$

Полученные оценки позволяют принять в качестве гарантированной нижней границы для оценки трудозатрат следующую формулу:

$$\Delta T_{pc} \mid \beta_{pc} * \mu^{(1)}_{\rho_{cp}} * n^{2} \tag{3.55}$$

имея в виду при этом, что трудозатраты в реальных случаях могут быть намного больше.

Рассмотрим величину β_{cp} и дадим ей определение. Общее число связей, которые должны быть просмотрены при алгоритмической и информационно-логической увязке блок-схемы, включая и многократные повторные просмотры, равно:

$$\sum_{1}^{\sigma} \rho_{\varphi} \tag{3.56}$$

В действительности же по разным причинам на каждом φ-ом пути просматривается такое же или меньшее число связей:

$$\rho_{\varphi_g} \leq \rho_{\varphi}$$

Общее число просмотренных связей на σ путях будет:

$$\sum_{1}^{\sigma} \rho_{\varphi_g} \geq \sum_{1}^{\sigma} \rho_{\varphi} \qquad (3.57)$$

Поскольку известно не все число путей σ , то часть путей может оказаться не просмотренной. С учетом этого обстоятельства:

$$\sum_{1}^{\sigma_g} \rho_{\varphi_g} \geq \sum_{1}^{\sigma} \rho_{\varphi_g} \geq \sum_{1}^{\sigma} \rho_{\varphi} \qquad (3.58)$$

где:

$\sum_{1}^{\sigma_g} \rho_{\varphi_g}$ – действительное число проверенных связей с учетом повторных просмотров;

$\sum_{1}^{\sigma} \rho_{\varphi}$ – действительное число всех связей, подлежащих проверке с учетом повторных просмотров.

Можно определить β_{cp} как соотношение:

$$\beta_{cp} = \frac{\sum_{1}^{\sigma_g} \rho_{\varphi_g}}{\sum_{1}^{\sigma} \rho_{\varphi}} \geq 1 \qquad (3.59)$$

где:

$\varphi_g = 1,2,...,\sigma_g$; $\varphi = 1,2,...,\sigma$; $\sigma_g \geq \sigma$

Итак, необходимо определить ту долю от общего числа всех связей, подлежащих просмотру, которая действительно будет просмотрена. Для этого примем такую дисциплину просмотра, которая не противоречит здравому смыслу и, с учетом дефицита времени, которым всегда озабочен разработчик алгоритмов и программ, даст ему уверенность в том, что им будет просмотрено достаточное число связей. Будем считать, что разработчик, проводя просмотр первого пути, отмечает все просматриваемые связи и осуществляет их проверку и информационно-логическую и алгоритмическую увязку. При просмотре любого последующего пути он просматривает и отмечает все связи, а проверяет информационно-логическую увязку лишь тех связей, которые отмечены не более чем λ раз. Как только какая-либо связь оказывается отмеченной λ раз, она при всех последующих просмотрах просматривается, отмечается, но не проверяется.

В этом случае будем иметь следующую диаграмму проверок. При просмотре первого пути проверяются, увязываются и отмечаются все ρ_1 связей. При просмотре λ -го пути проверяются, увязываются и отмечаются все ρ_1 связей. При просмотре $(\lambda + 1)$ -го пути отмечаются все $\rho_{\lambda+1}$ связей, а проверяются и увязываются лишь $\rho^{1}_{\lambda+1}$ связей, которые помечены менее чем λ раз.

Те связи, которые помечены λ раз, считаются окончательно проверенными и более не проверяются, но продолжают отмечаться, если встречаются при дальнейших проверках. Число таких связей при просмотре $(\lambda + 1)$-го пути равно $\rho^{2}_{\lambda+1}$.

$$\rho^{1}_{\lambda+1} = \rho_{\lambda} * \psi_{\lambda+1} \tag{3.60}$$

$$\rho^{2}_{\lambda+1} = \rho_{\lambda+1} * (1 - \psi_{\lambda+1}) \tag{3.61}$$

где: $\psi_{\lambda+1}$ - доля связей, просмотренных и отмеченных при просмотре $(\lambda + 1)$-го пути, оказались отмечены не более λ раз.

Тогда при просмотре $(\lambda + 2)$-го пути:

$$\rho^{1}_{\lambda+2} = \rho_{\lambda+2} * \psi_{\lambda+2} \tag{3.62}$$

$$\rho^{2}_{\lambda+2} = \rho_{\lambda+2} * \left(1 - \psi_{\lambda+2}\right) \tag{3.63}$$

При просмотре φ^{*}_{1}-го пути $(\varphi^{*} > \lambda)$:

$$\rho^{1}_{\varphi^{*}} = \rho_{\varphi^{*}} * \psi_{\varphi^{*}} \tag{3.64}$$

$$\rho^{2}_{\varphi^{*}} = \rho_{\varphi^{*}} * \left(1 - \psi_{\varphi^{*}}\right) \tag{3.65}$$

При просмотре φ^{*} путей будет просмотрено, увязано и проверено связей:

$$\mathop{e}_{1}^{\varphi^{*}} \rho^{1}_{\varphi} = \rho_1 + \rho_2 + \rho_3 + ... + \rho_{\lambda} + \rho^{1}_{\lambda+1} + \rho^{1}_{\lambda+2} + ... + \rho^{1}_{\varphi^{*}} \tag{3.66}$$

Далее:

$$\mathop{e}_{1}^{\varphi^{*}} \rho^{1}_{\varphi} = \rho_1 + \rho_2 + \rho_3 + ... + \rho_{\lambda} + \rho_{\lambda+1} * \psi_{\lambda+1} + ... + \rho_{\varphi^{*}} * \psi_{\varphi^{*}} \tag{3.67}$$

или:

$$\mathop{e}_{1}^{\varphi^{*}} \rho^{1}_{\varphi} = \mathop{e}_{1}^{\lambda} \rho_{\varphi} + \mathop{e}_{\lambda}^{\varphi^{*}} \rho_{\varphi} * \psi_{\varphi}$$

При просмотре в блок-схеме всех σ путей:

$$\mathop{e}_{1}^{\sigma} \rho^{1}_{\varphi} = \mathop{e}_{1}^{\lambda} \rho_{\varphi} + \mathop{e}_{\lambda}^{\varphi^{*}} \rho_{\varphi} * \psi_{\varphi} \tag{3.68}$$

Однако все пути, как правило, неизвестны. Минимальное число путей может быть определено, как было показано выше, по формуле: $\sigma_{\min} = \nu + 1$. Тогда все пути сверх этого числа, если не проводить конвертирование графа (блок-схемы), просматриваются с некоторой вероятностью Υ_{φ}, следовательно:

$$\sum_{1}^{\sigma} \rho_{\varphi}^{1} = \sum_{1}^{\lambda} \rho_{\varphi} + \sum_{1}^{\sigma_{min}} \rho_{\varphi} * \psi_{\varphi} + \sum_{\sigma_{min}}^{\sigma} \Upsilon_{\varphi} * \rho_{\varphi} * \psi_{\varphi} \quad (3.69)$$

Упростим это выражение. В действительности мы не знаем ρ_{φ}, так как никогда заранее не известен порядок просмотра путей в графе. Поэтому, не совершая существенной ошибки, можно принять:

$$\rho_{\varphi} = \rho_{cp} = \frac{\sum_{1}^{\sigma} \rho_{\varphi}}{\sigma} \quad (3.70)$$

Тогда: $$\sum_{1}^{\sigma} \rho_{\varphi}^{1} = \frac{\sum_{1}^{\sigma} \rho_{\varphi}}{\sigma} * \left(\lambda + \sum_{\lambda}^{\sigma_{min}} \psi_{\varphi} + \sum_{\sigma_{min}}^{\sigma} \Upsilon_{\varphi} * \psi_{\varphi} \right) \quad (3.71)$$

Далее: $$\frac{\sum_{1}^{\sigma} \rho_{\varphi}^{1}}{\sum_{1}^{\sigma} \rho_{\varphi}} * \sigma = \lambda + \sum_{1}^{\sigma_{min}} \psi_{\varphi} + \sum_{\sigma_{min}}^{\sigma} \Upsilon_{\varphi} * \psi_{\varphi}$$

(3.72)

Имея в виду, что $\rho_{\varphi}^{1} = \rho_{\varphi_g}$, получим: $\beta_{pc} * \sigma = \lambda + \sum_{\lambda}^{\sigma_{min}} \psi_{\varphi} + \sum_{\sigma_{min}}^{\sigma} \Upsilon_{\varphi} * \psi_{\varphi}$ (3.73)

Величина ψ_{φ} заранее не известна и не известна ее зависимость от φ. Поэтому будем считать ее постоянной и некоторой средней. Тогда:

$\psi_{\lambda+1} = \psi$

$\psi_{\lambda+2} = \psi_{\lambda+1} * \psi = \psi^{2}$

.....

$\psi_{\varphi} = \psi^{\varphi-\lambda}$

.....

$\psi_{\sigma_{min}} = \psi^{\sigma_{min}-\lambda}$

Следовательно:

$$\sum_{\lambda}^{\sigma_{min}} \psi_{\varphi} = \sum_{\lambda}^{\sigma_{min}} \psi^{\varphi-\lambda} \quad (3.74)$$

где:

$\varphi = \lambda + 1; \lambda + 2; ....; \sigma_{min}$

$\varphi - \lambda = 1, 2, ..., \sigma_{min} - \lambda$

То есть:

$$\sum_{\lambda}^{\sigma_{min}} \psi^{\varphi-\lambda} = \psi + \psi^{2} + \psi^{3} + ... + \psi^{\sigma_{min}-\lambda} \quad (3.75)$$

Перейдем к последнему слагаемому: $\sum_{\sigma_{min}}^{\sigma} \gamma_\varphi * \psi_\varphi$. Распространяя правила просмотра и проверки связей, принятые для интервала от $\lambda + 1$ до σ_{min} на интервал от $\sigma_{min} + 1$ до σ. Напишем для данного слагаемого: $\varphi = \sigma_{min} + 1; \sigma_{min} + 2; ...; \sigma$

$\psi_{\sigma_{min}} = \psi^{\sigma_{min} - \lambda + 1}$

$\psi_{\sigma_{min}+2} = \psi^{\sigma_{min} - \lambda + 2}$

....

$\psi_{\sigma} = \psi^{\sigma - \lambda}$

Определим вероятность γ_φ. Будем понимать эту вероятность как вероятность события, состоящего в том, что любой из путей, номер которого находится в интервале от $\sigma_{min} + 1$ до σ, будет просмотрен. Очевидно, что эта вероятность равна доле просмотренных путей в интервале от $\sigma_{min} + 1$ до σ. Обозначим: $\sigma_{зад}$ - заданное, выбранное, либо предполагаемое для просмотра число путей, причем такое, что: $\sigma_{зад} < \sigma$.

Тогда: $$\gamma_\varphi = \frac{\sigma_{зад} - \sigma_{min}}{\sigma - \sigma_{min}} \qquad (3.76)$$

и выражение (3.72) получит вид:

$$\beta_{рс} * \sigma = \lambda + \sum_{\lambda}^{\sigma_{min}} \psi_\varphi + \frac{\sigma_{зад} - \sigma_{min}}{\sigma - \sigma_{min}} * \sum_{\sigma_{min}}^{\sigma} \psi_\varphi \qquad (3.77)$$

Далее:

$$\beta_{.с.} = \frac{\lambda}{\sigma} + \frac{1}{\sigma} * \sum_{\lambda}^{\sigma_{min}} \psi_\varphi + \frac{\sigma_{зад} - \sigma_{min}}{\sigma\left(\sigma - \sigma_{min}\right)} * \sum_{\sigma_{min}}^{\sigma} \psi_\varphi \qquad (3.78)$$

Введем обозначения:

$$\bar{\sigma} = \frac{\sigma}{\sigma_{min}}; \quad \bar{\sigma}_{зад} = \frac{\sigma_{зад}}{\sigma_{min}}$$

и получим: $$\beta_{рс} = \frac{1}{\sigma_{min}} * \left(\frac{\lambda}{\bar{\sigma}} + \frac{1}{\bar{\sigma}} * \sum_{\lambda}^{\sigma_{min}} \psi_\varphi + \frac{\bar{\sigma}_{зад} - 1}{\bar{\sigma}\left(\bar{\sigma} - 1\right)} * \sum_{\sigma_{min}}^{\bar{\sigma} * \sigma_{min}} \psi_\varphi\right) \qquad (3.79)$$

Очевидно, что: $\beta_{рс} \to 1$.

Следовательно, величина $\beta_{рс}$ характеризует полноту проработки алгоритма и программы в целом.

Рассмотрим некоторые числовые примеры.

1. Пусть:

$\sigma_{\min} = 10;$ $\quad \bar{\sigma}_{\text{зад}} = 1,5;$

$\lambda = 2;$ $\quad \sigma_{\text{зад}} = 15;$

$\sigma = 20;$ $\quad \psi = 0,8;$

$\bar{\sigma} = 2;$

Найдем слагаемые в скобках в выражении (3.79):

$$\lambda / \bar{\sigma} = 2/2 = 1;$$

$$e_{\lambda}^{\sigma_{\min}} \psi_{\varphi} = e_{3}^{10} \psi_{\varphi} = 3,328911;$$

$$e_{\sigma_{\min}}^{\bar{\sigma}\cdot\sigma_{\min}} \psi_{\varphi} = e_{10}^{20} \psi_{\varphi} = 0,4511865;$$

и, наконец:

$$\beta_{ср} = \frac{1}{10} * \left(1 + \frac{3,328911}{2} + \frac{1,5-1}{2(2-1)} * 0,5991607\right) = 0,2814$$

В рассмотренном примере в случае сравнительно простого графа (блок-схемы) при обязательной двукратной проверке каждой связи оказывается, что программист выполняет чуть больше четверти необходимой работы по взаимной информационно-логической увязке всех связей в блок-схеме.

2. Рассмотрим более сложный пример:

$\sigma_{\min} = 20;$ $\quad \bar{\sigma}_{\text{зад}} = 1,5;$

$\lambda = 2;$ $\quad \sigma_{\text{зад}} = 30;$

$\bar{\sigma} = 2;$ $\quad \psi = 0,8;$

$\sigma = 40;$

В этом примере:

$$\lambda / \bar{\sigma} = 1;$$

$$e_{\lambda}^{\sigma_{\min}} \psi_{\varphi} = e_{3}^{20} \psi_{\varphi} = 3,9279423;$$

$$e_{\sigma_{\min}}^{\sigma} \psi_{\varphi} = e_{21}^{40} \psi_{\varphi} = 0,0712685;$$

$$\beta_{ср} = \frac{1}{20} * \left(1 + \frac{1}{2} * 3,9279423 + \frac{1}{4} * 0,0712685\right) = 0,149;$$

Можно сделать некоторый, хотя и не вполне очевидный, вывод: при сохранении тех же характеристик процесса разработки алгоритма: $\lambda = 2; \psi = 0,8$ при усложнении блок-схемы вдвое: $\sigma_1 = 20; \sigma_2 = 40$ полнота проверки алгоритма и программы при разработке уменьшается почти вдвое, хотя, казалось бы, она не должна была бы измениться столь существенно.

Таким образом, величина β_{pc} определяет ту долю от общего числа всех связей, подлежащих просмотру, которая действительно будет просмотрена. Величина β_{pc} характеризует полноту проработки алгоритма и программы в целом. Введение такого понятия как полнота проверки алгоритма и программы при их создании и анализ величины β_{pc} дает возможность в дальнейшем подойти к оценке вероятности появления ошибок в алгоритме или программе или математического ожидания числа ошибок. В задачу данного исследования подобная работа пока не входит. Поэтому вновь вернемся к выражению (3.22): $\Delta T_{pc} = \beta_{cp} * \mu^{(1)}_{\rho_{cp}} * \rho_{cp} * \sigma$

Выше было показано, что можно принять: $\Delta T_{pc} \approx \beta_{cp} * \mu^{(1)}_{\rho_{cp}} * n^2$

Имея в виду (3.78) получим:

$$\Delta T_{pc} \approx \mu_{\rho_{cp}} * \frac{1}{\sigma_{\min}} \left(\frac{\lambda}{\bar{\sigma}} + \frac{1}{\bar{\sigma}} \sum_{\lambda}^{\sigma_{\min}} \psi_{\varphi} + \frac{\overline{\sigma_{зад}} - 1}{\bar{\sigma}(\bar{\sigma} - 1)} * \sum_{\sigma_{\min}}^{\bar{\sigma} * \sigma_{\min}} \psi_{\varphi} \right) * n^2$$

Пусть:

$\psi_{\varphi} = 1$, $\lambda = 0$, $\sigma_{зад} = \sigma$

Тогда:

$$\Delta T_{pc} \approx \mu_{\rho_{cp}} * \frac{1}{\sigma_{\min}} \left(0 + \frac{\sigma_{\min}}{\bar{\sigma}} + \frac{\sigma - \sigma_{\min} - 1}{\bar{\sigma}} \right) * n^2 = \mu^{(1)}_{\rho_{cp}} * \frac{\sigma}{\sigma_{\min} * \bar{\sigma}} * n^2 = \mu^{(1)}_{\rho_{cp}} * n^2$$

Итак, в идеальном случае полной проверки алгоритма и программы по всем путям имеем:

$$\Delta T_{pc} \approx \mu^{(1)}_{\rho_{cp}} * n^2 \qquad (3.80)$$

3.7. Полная стоимость разработки программного продукта

Общие трудозатраты на разработку алгоритма и программы из n модулей или блоков неавтоматизированным способом выразятся следующей формулой:

$$C = s_1 \mu_1 n + s_2 \mu_2 n + s_3 \sum_{1}^{n} \tau_{3k} + s_4 \mu_4 n + s_5 \sum_{1}^{n} \tau_{5k} + s_6 \alpha \beta n^2 + s_6 \mu^{(1)}_{\rho_{cp}} n^2$$

$$T = \mu_1 n + \mu_2 n + \sum_{1}^{n} \tau_{3k} + \mu_4 n + \sum_{1}^{n} \tau_{5k} + \alpha \beta n^2 + \mu^{(1)}_{\rho_{cp}} n^2 \qquad (3.81)$$

где:

$C(T)$ – полная стоимость (или трудозатраты в часах) разработки алгоритма и программы из n модулей;

s – стоимость одного часа трудозатрат на соответствующем этапе разработки.

Работа по изучению объекта и его декомпозиции, а также по разработке блок-схемы являются наиболее сложными и высокооплачиваемыми, то есть: $s_1 = s_2 = s_4$. Кроме того, следует полагать, что: $s_5 < s_3 < s_1$

Поэтому:

$$C \approx s_1 * (\mu_1 + \mu_2 + \mu_3) * n + s_3 \sum_1^n \tau_{3k} + s_5 \sum_1^n \tau_{5k} + s_6 * \alpha * \beta * n^2 + s_6 * \mu_{\rho_{cp}}^{(1)} * n^2$$

Или:

$$C \approx A * n + B * n^2 \qquad (3.82)$$

Последние формулы, хотя и приближенно, но выражают определенную закономерность зависимости трудозатрат или стоимости создания МПО вручную от числа модулей или операторов (блоков) программы. Значения коэффициентов A и B зависят от трудоемкости работ по созданию МПО на различных этапах. Характер полученных функциональных зависимостей совпадает с экспериментальными данными [2], полученными статистическим путем, что может служить косвенным подтверждением достоверности полученных формул. Выражение (3.82) может быть написано не для определения стоимости разработки МПО, а для определения трудозатрат, то есть:

$$T_{pc} = A_t * n + B_t * n^2 \qquad (3.83)$$

Исходя из имеющейся статистики, в качестве первого приближения для коэффициентов A и B могут быть приняты следующие значения: $A_T = 1$ и $B_T = 0,02$. Эти значения соответствуют случаю составления алгоритмов и программ простейшего вида, как например, рассмотренный в работе [6] алгоритм Евклида. Значения функции $T_{pc}(n)$ представлены на рис.13.

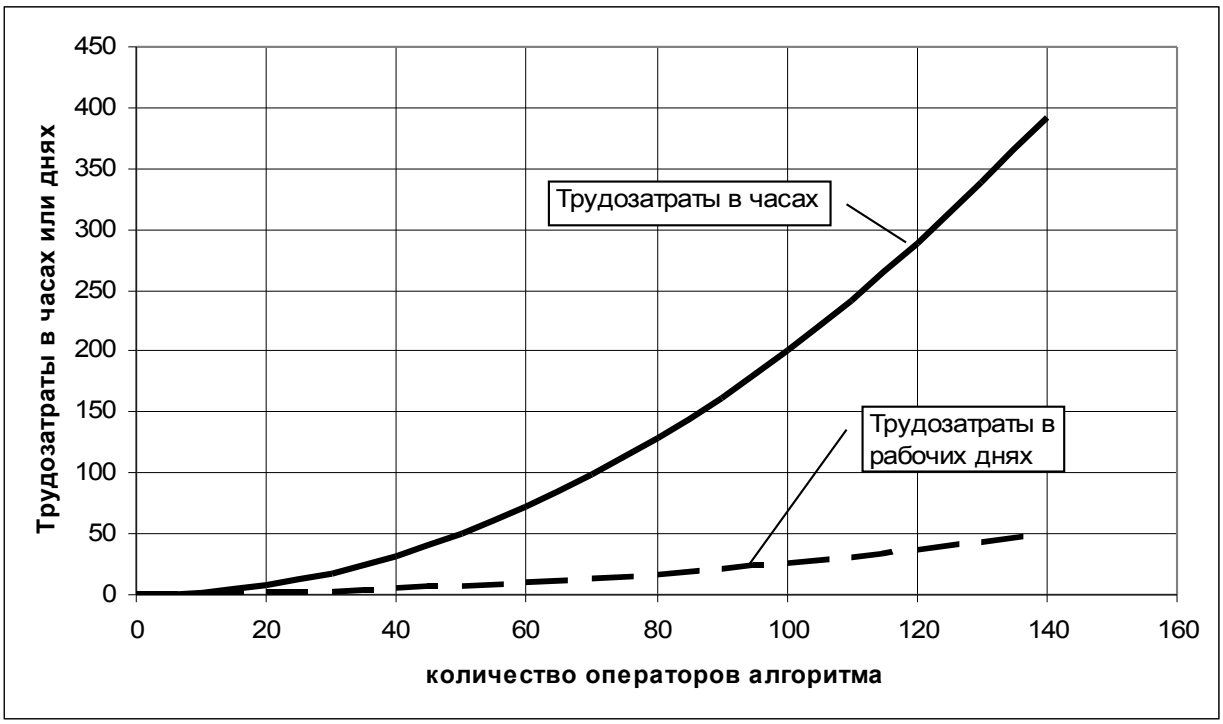

Рис. 13. Значения функции $T_{pc}(n)$

Из этого следует вполне очевидный вывод о необходимости снижения трудозатрат на этапе синтеза алгоритма, что может привести к существенному снижению трудозатрат на разработку МПО в целом.

4. Формулы для оценки трудозатрат на разработку МПО способом автоматизированного синтеза алгоритмов и программ

Не вдаваясь в детали автоматизации синтеза алгоритмов и программ, попробуем оценить стоимость или трудозатраты на такую автоматизацию. При этом будем исходить из того, что автоматизация осуществляется программно-алгоритмическими средствами.

Автоматизированный синтез алгоритмов и программ может быть осуществлен на 6 и 7 этапах разработки МПО, а способы работы на 1-5 этапах практически не меняются. Небольшие необходимые изменения могут состоять в том, что создание отдельных модулей или блоков должно проводится единообразно, имеется в виду, что заранее должен быть разработан стандарт на модули и операторы, который должен строго выдерживаться. Это не должно привести к резкому увеличению трудозатрат на 1-5 этапах. Кроме того, есть возможность разрабатывать отдельные блоки или модули также автоматизированным способом, что позволит еще больше снизить трудозатраты и сделать автоматизированный синтез еще более выгодным.

Автоматизированный синтез как алгоритмов и программ, так и отдельных модулей, может осуществляться в один или два этапа. При более простых методах синтеза синтез осуществляется в два этапа. На первом этапе осуществляется автоматизированный синтез алгоритма по блок-схеме, а на втором - автоматизированный синтез программы по полученному алгоритму. Второй этап в

настоящее время в той или иной степени присутствует, тогда как первый – нет. Таким образом, даже наличие разработанных методов синтеза программ по алгоритмам пока не позволяет решить проблему автоматизированного синтеза МПО в необходимой степени. Разработанные к настоящему времени методы автоматизированного синтеза, и их объединение с методами автоматизированного построения программ позволят осуществлять автоматизированный синтез алгоритмов и программ в один этап. В этом случае трудоемкость непосредственно автоматизированного этапа синтеза МПО будет практически равна нулю. Трудозатраты пойдут лишь на разработку метода и программы самого синтеза. И в этом случае необходимо будет учитывать лишь трудозатраты на доработку полученных программ синтеза применительно к тому или иному классу задач.

Итак, если метод автоматизированного синтеза МПО разработан [7] и реализован для некоторого числа M классов задач, достаточно близких между собой по характеру исходных данных, языкам, типам алгоритмов и т.д., то для всех задач подобных классов оценка стоимости разработки МПО должна выполняться с учетом этого обстоятельства.
Поэтому в случае автоматизированного синтеза МПО можно записать:

$$C_{i_{AC}} = \sum_{l=1}^{l=5} C_{li} + \delta_{ACm} \tag{4.1}$$

где: δ_{ACm} – дополнительная стоимость разработки МПО, приходящаяся на каждую из задач, входящих в M классов.

Оценим эту величину. Общее число задач, входящих в M классов, будет:

$$\sum_{1}^{M} N_m \text{, где } n = 1,2,3,...,M \tag{4.2}$$

Пусть общие расходы на разработку методов автоматизированного синтеза (МАС) МПО составят C_{AC}, а дополнительные расходы на доработку МАС МПО применительно к m-ому классу задач $-\Delta C_{ACm}$. Тогда дополнительные расходы на разработку МПО, приходящиеся в среднем на одну задачу m-ого класса за счет разработки метода автоматизированного синтеза, составят:

$$\delta_{ACm} = \frac{C_{AC}}{\sum_{1}^{M} N_m} + \frac{\Delta C_{ACm}}{N_m} \tag{4.3}$$

Здесь:

$$C_{AC} = s_6 * T_{MAC} \tag{4.4}$$

$$\Delta C_{ACm} = s_6 * \Delta T_{MAC_m} \tag{4.5}$$

где:

T_{MAC} – трудозатраты на разработку и реализацию общего для всех задач M классов метода автоматизированного синтеза МПО;

ΔT_{MAC_m} – трудозатраты на доработку МАС МПО применительно к задачам m -ого класса.

Таким образом:

$$\delta_{ACm} = \frac{s_6 * T_{MAC}}{\sum_{1}^{m} N_m} + \frac{s_6 * \Delta T_{MACm}}{N_m}$$

Если предположить, что в каждом классе некоторое среднее число задач или $N = N_{cp} = const$, то:

$$\delta_{ACm} = \frac{s_6 * T_{MAC}}{M * N_{cp}} + \frac{s_6 * \Delta T_{MACm}}{N_{cp}} \quad (4.6)$$

или:

$$\delta_{ACm} = \frac{s_6 * T_{MAC}}{N_{cp}} * \left(\frac{1}{M} + \frac{\Delta T_{MACm}}{T_{MAC}} \right) \quad (4.7)$$

Очевидно, что δ_{ACm} зависит не от сложности конкретной задачи, а только от свойств класса задач, и, таким образом, является для каждого класса задач величиной постоянной. Если МАС МПО создается только для одного класса задач, то:

$$\delta_{ACm} = \frac{s_6 * T_{MAC}}{N_{cp}} \quad (4.8)$$

Из этого следует, что δC_{ACm} не зависит от сложности МПО конкретной задачи, а только от свойств класса задач и, таким образом, является величиной постоянной для каждого класса.
Если метод автоматизированного синтеза создается для одного класса задач, то:

$$\delta_{AC} = \frac{s_6 * T_{AC}}{N_{cp}} \quad (4.9)$$

5. Теоретические области целесообразного применения различных методов построения МПО

5.1 Неавтоматизированная разработка МПО

$$C_{pc} = s_1(\mu_1 + \mu_2 + \mu_3) * n + s_3 * \sum_{1}^{n} \tau_{3k} + s_5 * \sum_{1}^{n} \tau_{5k} + s_6 * \alpha * \beta * n^2 + s_6 * \beta_{pc} * \mu_{\rho_{cp}}^{(1)} * n^2$$

Здесь:

s_1 - стоимость одного часа трудозатрат на 1, 2 и 4 этапах разработки МПО,

s_3 - стоимость одного часа трудозатрат на третьем этапе разработки МПО,

s_5 - стоимость одного часа трудозатрат на пятом этапе разработки МПО,

s_6 - стоимость одного часа трудозатрат на 6 и 7 этапах разработки МПО,

μ_1 - удельные трудозатраты на первом этапе разработки МПО в среднем приходящиеся на один модуль алгоритма (модели),

μ_2 - аналогично на втором этапе,

μ_4 - аналогично на четвертом этапе,

τ_{3k} - удельные трудозатраты, приходящиеся на третьем этапе разработки МПО на k-ый модуль,

τ_{5k} - аналогично для пятого этапа,

$\alpha * \beta \approx 0,01 \div 0,02$

$\mu_{\rho_{cp}}^{(1)}$ - удельные трудозатраты, приходящиеся на проверку и информационно-логическую увязку каждой связи, встречающейся при просмотре очередного пути в алгоритме,

β_{pc} - показатель полноты проверки всех связей в алгоритме (необходимые расчетные формулы для определения β_{pc} приведены выше),

n - число модулей в алгоритме.

5. 2. Автоматизированная разработка МПО

$$C_{AC} = s_1 * (\mu_1 + \mu_2 + \mu_4) * n + s_5 * \sum_1^n \tau_{5k} + \delta C_{ACm}$$

где δC_{ACm} может определяться по следующим формулам:

$$\delta_{ACm} = \frac{s_6 * T_{MAC}}{\sum_1^m N_m} + \frac{s_6 * \Delta T_{MACm}}{N_m}$$

где:

T_{MAC}, ΔT_{MAC} - определены выше,

$m = 1,...,M$

M - число классов задач,

N_m - число задач в классе.

T_{MAC} может, вообще говоря, определяться по формуле приведенной к расчету трудоемкости, либо может определяться из практики. Имеющийся на сегодняшний день опыт разработки МПО позволяет дать следующую оценку T_{MAC}: $T_{MAC} \approx 2000 \div 5000$ - .

Для величины ΔT_{MAC} никакой статистики нет, хотя предположительно можно дать следующую оценку: $\Delta T_{MAC} \approx 1000 \div 2000$ - . Если в каждом из M классов задач число задач составляет N_{cp}, то: $\delta C_{ACm} = \frac{s_6}{N_{cp}}(\frac{T_{MAC}}{M} + \Delta T_{MACm})$

или: $\delta C_{ACm} = \frac{s_6 * T_{MAC}}{N_{cp}} (\frac{1}{M} + \frac{\Delta T_{MACm}}{T_{MAC}})$

При разработке МПО только для одного класса задач: $\delta_{AC} = \frac{s_6 * T_{AC}}{N_{cp}}$

5.3. Области целесообразного применения метода автоматизированного синтеза

Оценки областей целесообразного применения носят качественный характер, поэтому воспользуемся упрощенными формулами. Для сравнения рассмотрим следующие случаи разработки МПО.

5.3.1. Сравнительная оценка эффективности автоматизированного и неавтоматизированного способов построения МПО

Для иллюстрации сравнительной оценки используем упрощенные выражения для расчета C_{AC} и C_{PC}.

Для прямого, ручного программирования:

$C_{pc} \approx A * n + B * n^2$

Для автоматизированного построения МПО получим следующее упрощенное выражение:

$C_{AC} = A * n + D_1$

Оценим значения величин A, B и D_1.

Для упрощения расчетов примем:

1. $s_1 = s_3 = s_5 = s_6 = 1\$/час$,
2. $\mu_1 = 10 ч/час$,
3. $\mu_2 = 5 ч/час$,
4. $\mu_4 = 5 ч/час$,
5. $\tau_{3k} = 50 час$,
6. $\tau_{5k} = 500 час$,
7. $\alpha * \beta = 0,02$,
8. $\mu_{p_{cp}}^{(1)} = 5$,
9. $T_{AC} = 5000 час/год$,
10. $N_{cp} = 1,4,25$,
11. $\gamma = 1$
12. $D_1 = 2000, 5000, 10000, 20000$

При этих исходных данных: $A = 570$, $B \approx 5$.

Тогда:

$C_{pc} \approx 570 * n + 5 * n^2$

$C_{AC} = 570 * n + D_1$

Результаты расчетов, приведенные на рис.14, позволяют сделать вывод о том, что даже при весьма невыгодных исходных данных для автоматизированного построения МПО, его внедрение становится выгодным, начиная с весьма небольших размеров программ (примерно 50-70

операторов). При исходных данных, более близких к реальности, выигрыш от применения автоматизации становится существенно ощутимее.

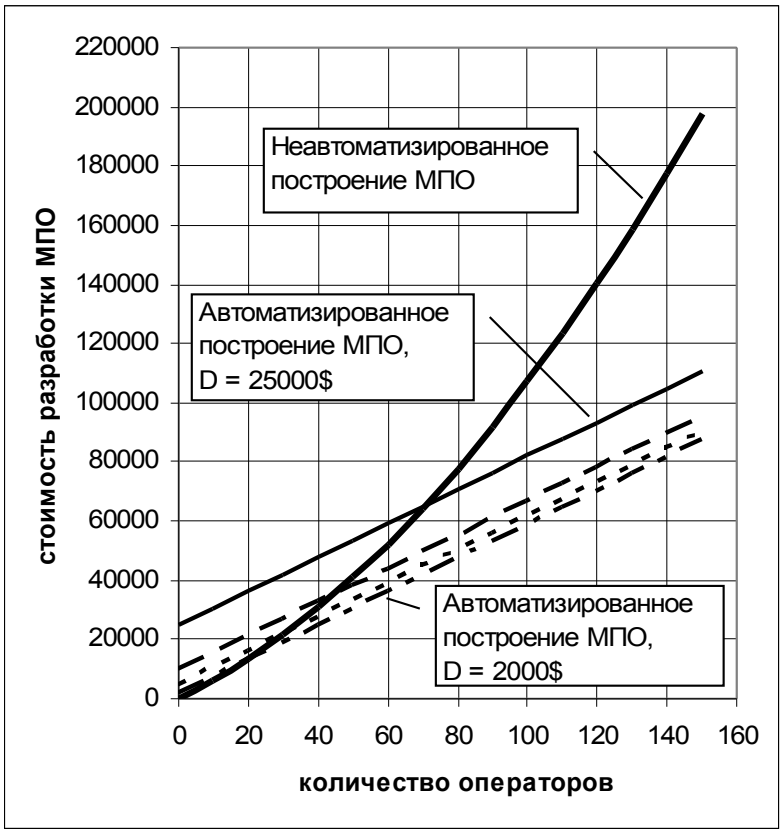

Рис. 14. Зависимость стоимости разработки МПО в случае неавтоматизированного построения программы и в случае применения автоматизированного синтеза.

5.3.2. Области целесообразного применения методов автоматизированного синтеза МПО

Оценка областей целесообразного применения методов автоматизированного синтеза носит в первую очередь качественный характер, поскольку весьма сложно найти достоверные исходные данные. Для оценки областей целесообразного применения методов автоматизированного синтеза можно ограничиться формулами для трудоемкости разработки МПО. Для сравнения рассмотрим следующие случаи.

Непосредственное прямое программирование

$$T_{pc0} = A_t * n + B_t * \left(p * n \right)^2 \qquad (5.1)$$

где:

p - средняя длина программного модуля;

n - количество модулей;

$p * n$ - средняя длина всей программы.

A_t, B_t - коэффициенты, определяемые, как было показано выше, но без учета стоимости одного часа трудозатрат.

Модульное программирование

В этом случае производится отдельное программирование модулей вручную, а затем их объединение в общую программу также вручную.

$$T_{pc} = A_t * n + B_t * p^2 * n + B_t * n^2 \qquad (5.2)$$

Частично автоматизированная разработка МПО

Программирование отдельных модулей или блоков проводится общепринятыми методами вручную, а в единую систему или комплекс они собираются с помощью МАС МПО.

$$T_{AC} = A_t * n + B_t * p^2 * n + \gamma * D_1 \qquad (5.3)$$

где: D_1 – трудозатраты на разработку МАС МПО;

γ – доля рассматриваемой задачи во всем классе задач, для которого были разработаны МАС МПО.

Полностью автоматизированная разработка МПО

В этом случае, как отдельные модули, так и все МПО строится автоматизировано. В этом случае: $T_{AC_0} = A_{T_0} * n + \gamma * D_1$ (5.4)

Сравнивая различные варианты между собой, получим приближенные формулы, описывающие в функциональном виде границы раздела между областями целесообразного применения того или иного способа разработки МПО:

AC PC	3. T_{AC}	4. T_{AC_0}
1. T_{pc_0}	1/3 - T_{pc_0} / T_{AC}	1/4 - T_{pc_0} / T_{AC_0}
2. T_{pc}	2/3 - T_{pc} / T_{AC}	2/4 - T_{pc} / T_{AC_0}

Вариант 1-3. Сравнение непосредственной прямой разработки программ вручную с частично автоматизированным синтезом МПО: $T_{pc_0} = T_{AC}$

Из условия равенства трудозатрат, получим формулу для определения средней длины программного модуля:

$$p_{1,3} = \sqrt{\frac{\gamma * D_1}{5\left(n^2 - n\right)}} \qquad (5.5)$$

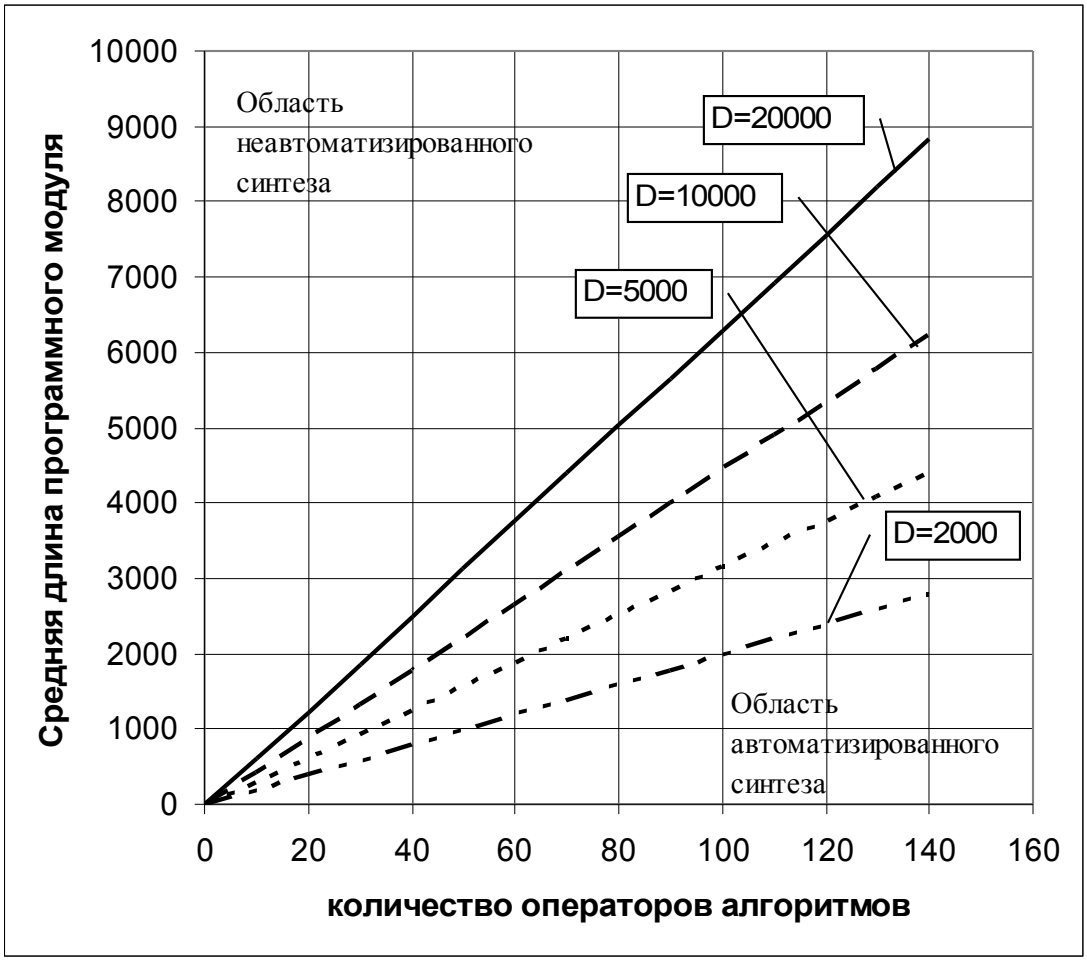

Рис.15.

Вариант 1-4. Сравнение непосредственной прямой разработки программ вручную с полностью автоматизированным синтезом: $T_{pc_0} = T_{AC_0}$. Слово "вручную" нельзя понимать буквально. В данной статье не учитывается возможное повышение производительности труда от появившихся в последние годы достаточно мощных средств машинного интерфейса, которые, с одной стороны, весьма облегчают работу создателей МПО, а с другой стороны, все-таки не решают проблему автоматизации синтеза структуры алгоритма или программы в полном объеме.

Из условия равенства трудозатрат, получим формулу для определения средней длины программного модуля: $p_{1,4} = \sqrt{\frac{\gamma * D_1 - 430n}{5n^2}}$ (5.6)

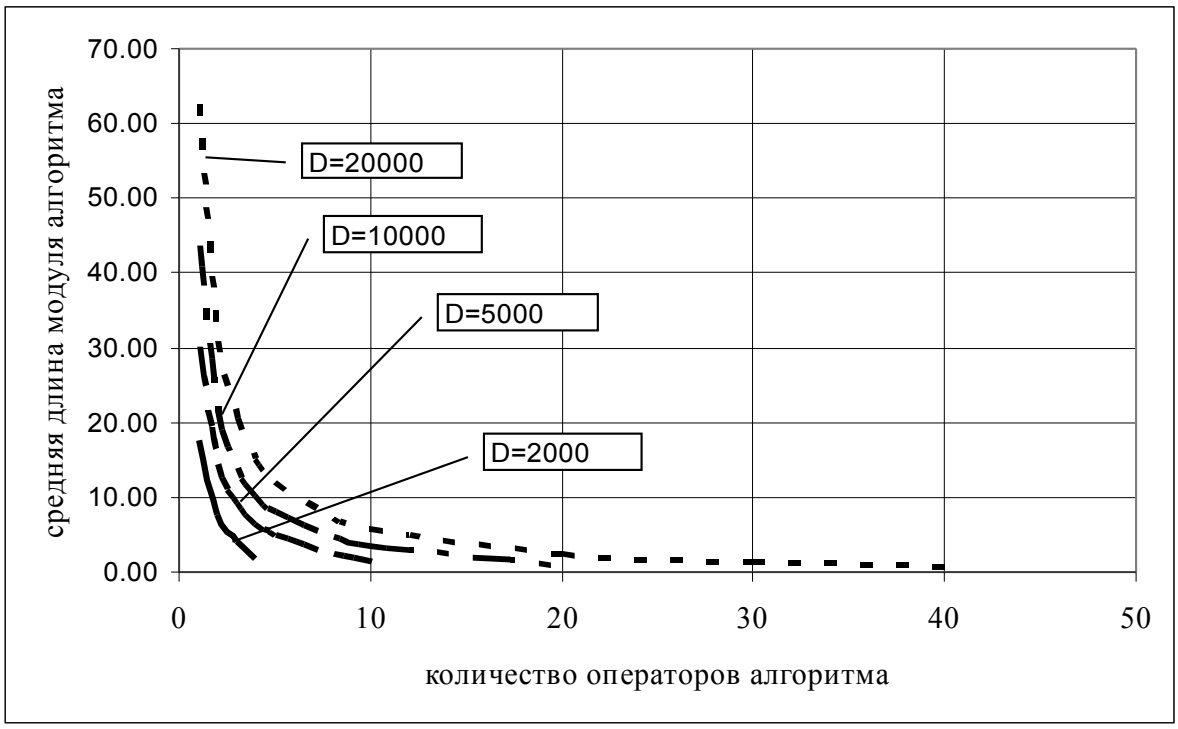

Рис. 16

Вариант 2-3. Сравнение модульной разработки МПО при ручном синтезе всей программы с частично автоматизированным синтезом МПО: $T_{pc} = T_{AC}$

Тогда: $D_1^* = \dfrac{B_T * n^2}{\gamma}$ (5.7)

В данном случае размеры модулей не влияют на оценку применимости МАС МПО. Важно лишь число модулей. При $\gamma = 1$ имеем следующие результаты:

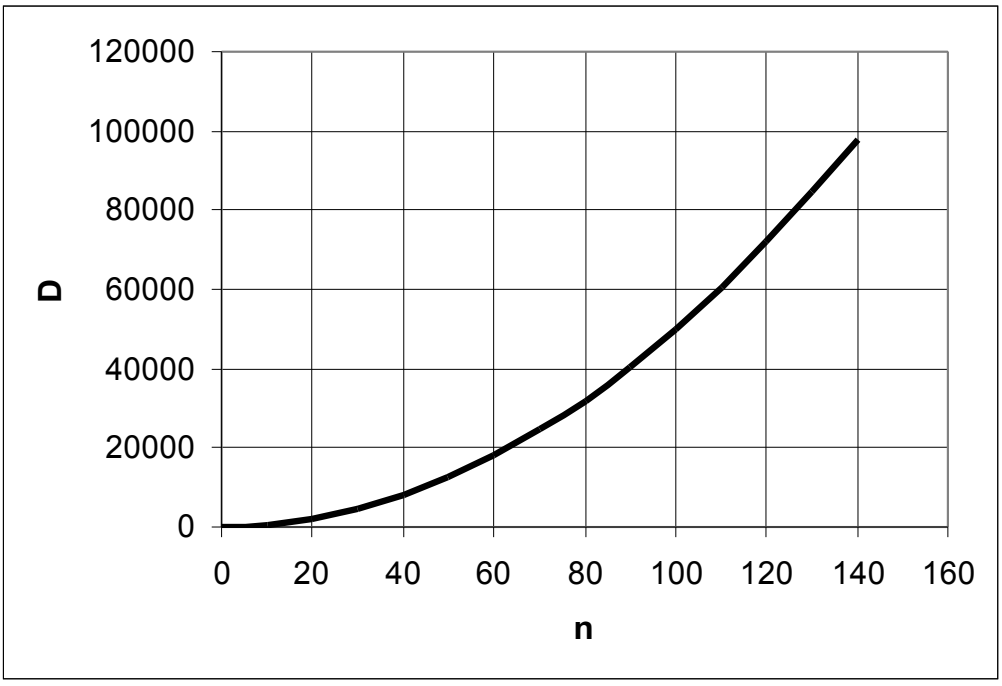

Рис. 17.

Вариант 2-4. Сравнение модульной разработки МПО при ручном синтезе программы с полностью автоматизированным синтезом МПО: $T_{pc} = T_{AC_0}$

$$p_{2,4} = \sqrt{\frac{\gamma * D_1 - 5n^2}{5n}} \qquad (5.8)$$

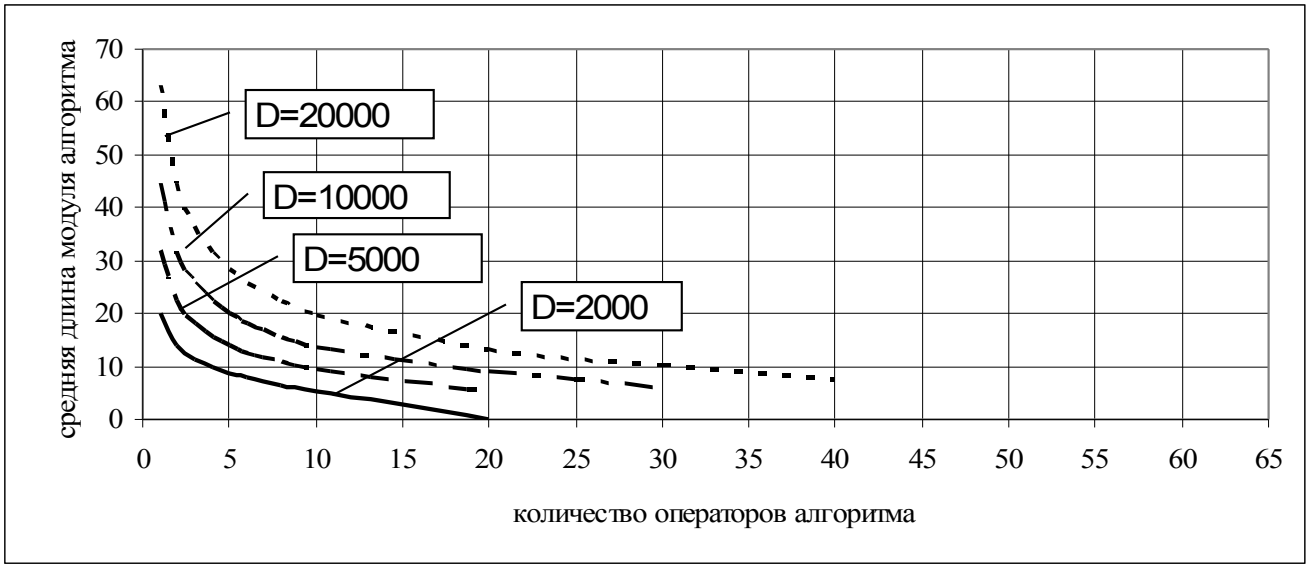

Рис. 18.

Для $p = 1$ и $\gamma = 1$ получим:

$D_1 = 5*n + 5*n^2$

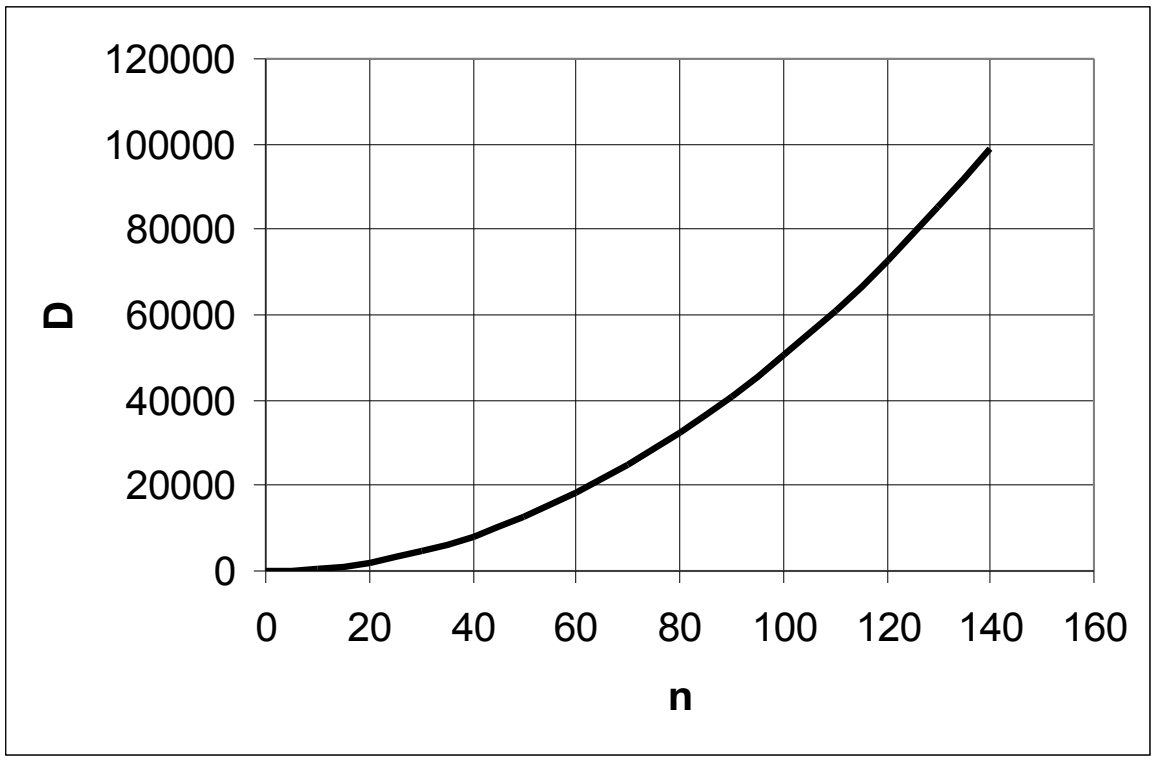

Рис. 19

Коэффициенты и постоянные величины приняты в тех значениях, которые встречаются в практической работе (экспертным путем). Важно было не столько установление точных границ, но показать, что они в принципе существуют.

Выводы:

1. Проведенный анализ технологии разработки математических моделей, алгоритмов и программ опирается на экспертные оценки квалифицированных специалистов в этой области, что позволило выявить структуру этого процесса и разработать основные математические зависимости для оценки трудозатрат на разработку МПО и его стоимости при применении различных методов.
2. Предлагаемая методика оценки трудозатрат и стоимости МПО как при его построении вручную, так и при автоматизированном построении опирается на структурные свойства моделей, алгоритмов и программ, рассматриваемых в качестве обыкновенных сетевых моделей.
3. Достоверность выявленных зависимостей от размеров алгоритмов и программ подтверждается их совпадением с зависимостями, выявленными на основе иных подходов другими авторами.
4. Проведенные с применением предлагаемой методики ориентировочные оценки эффективности методов автоматизированного синтеза МПО позволили получить следующие предварительные результаты:
5. Использование МАС МПО в полном объеме даже в случае уникального применения (1 задача) позволяет существенно снижать стоимость разработки МПО (особенно крупных программ). В сравнении с прямым программированием выгода появляется, уже начиная с программ размером в 100 операторов (1 задача), иногда даже меньше.

Литература:

1. Ермилов Л.И. Синтез сетевых моделей сложных процессов и систем. - М.: МО СССР, 1970. - 45с.
2. Малинин Л.И. Восемь дополнительных теорем из теории графов. В кн. Нечипоренко В.И. Структурный анализ сложных систем. - М.: Сов. Радио, 1977, - с.198-212.
3. Вельбицкий И.В., Ходаковский В.Н., Шолмов Л.И. Технологический комплекс производства программ на машинах ЕС-ЭВМ и БЭСМ-6. - М.: Статистика, 1980. - 263 с.
4. Ефимов Е.И. Решатели интеллектуальных задач. - М.: Наука, 1982. -316 с.
5. Тыугу Э.Х. Концептуальное программирование. - М.: Наука, 1984. -255 с.
6. Холстед М.Х. Начала науки о программах. - М.: Финансы и Статистика, 1981, 128с.
7. Эванчук С. Как снизить трудоемкость программирования, - М: Сов. Радио, 1983, 97 с.

8. Малинина Н.Л. Автоматизированный синтез вычислительных алгоритмов для проектирования сложных технических систем. Дис. к. ф.-м. н.// МАИ.- М.: 1993. - 200с.

СВЕДЕНИЯ ОБ АВТОРЕ

Малинина Наталия Леонидовна, доцент кафедры вычислительной математики и программирования Московского авиационного института (государственного технического университета), к.ф.-м.н.