DOI: 10.25728/mlsd.2023.0149

# МЯГКИЕ ЗАВИСИМОСТИ МЕЖДУ ПРОЕКТАМИ В УПРАВЛЕНИИ ПРОГРАММАМИ

**Бурков В.Н., Буркова И.В., Щепкин А.В.**
*Институт проблем управления им. В.А. Трапезникова РАН, Москва, Россия*
Vlab17@bk.ru, irbur27@gmail.com, av_shch@mail.ru

*Аннотация. Рассматриваются задачи управления программами при наличии мягких зависимостей между проектами. Реализация мягких зависимостей уменьшает время или затраты на выполнение непосредственно следующего проекта. Дается постановка и решение задачи, в которой требуется определить множество выполняемых мягких зависимостей при различных условиях.*

*Ключевые слова: программа, проект, жесткие зависимости, мягкие зависимости.*

## Введение

В задачах управления проектами и программами рассматриваются различного типа зависимости между работами или проектами (это зависимости типа старт-финиш, финиш-старт, финиш-финиш и старт-старт) [1-3].

При реализации программ часто возникает ситуация, когда результаты одного проекта можно использовать при выполнении других проектов, что позволяет уменьшать время выполнения или затраты на выполнение (или и то и другое) других проектов. Это тоже зависимости, получившие название мягких, поскольку их выполнение не обязательно, но может принести положительный эффект.

Граф программы с мягкими зависимостями может иметь контуры. Дается постановка и решение задачи, в которой требуется определить множество выполняемых мягких зависимостей, которые обеспечивают минимальную продолжительность выполнения программы.

Задачи с мягкими зависимостями рассматривались в [4,5]. Однако в этих работах эффект от учета нескольких мягких зависимостей на проект (и по времени, и по затратам) равнялся сумме эффектов от влияния отдельных зависимостей. В статье рассматривается общий случай, когда уменьшение продолжительностей $A_j(R_j)$ является произвольной функцией от множества $R_j$ мягких зависимостей.

## 1. Постановка задачи

Дадим основные определения.

**Определение 1.** Мягкие (рекомендательные) зависимости $(i, j)$ – зависимости, которые можно не выполнять, но реализация которых уменьшает время выполнения или затраты на выполнение (или то и другое) проекта $j$.

На рис.1 изображены мягкие зависимости, входящие в вершину $j$.

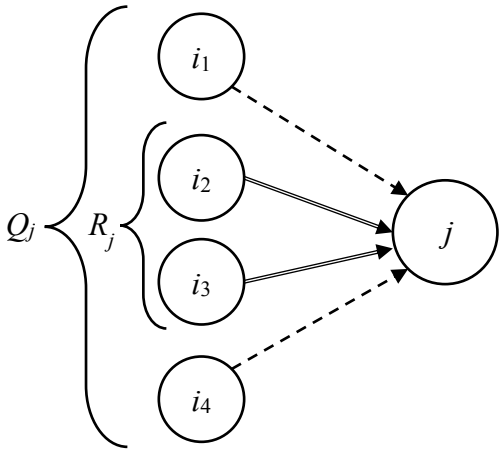



*Рис. 1. Мягкие зависимости, входящие в вершину j*

Здесь $Q_j$ – множество мягких зависимостей, заходящих в вершину $j$. $R_j$ – подмножество реализуемых мягких зависимостей.

Введем также следующие обозначения: $A_j(R_j)$ – уменьшение времени, $B_j(R_j)$ – уменьшение затрат, $\tau_i$ – продолжительность проекта, если мягкие зависимости не учитываются. Заметим, что $A_j(R_j) < \tau_j$, так как продолжительность проекта всегда больше нуля.

**Определение 2.** Жесткие зависимости – это классические зависимости типа «финиш-старт» (работа последующая не может начаться, пока не завершится работа предшествующая).

В дальнейшем будем считать, что $x_{ij} = 1$ если мягкая зависимость $(i, j)$ реализуется (выполняется), $x_{ij} = 0$ если не реализуется. Кроме того, имеющаяся, но не реализованная мягкая зависимость изображается как $- - \rightarrow$, соответственно, реализованная мягкая зависимость изображается в виде $\Rightarrow$ и, наконец, жесткая зависимость изображается как $\rightarrow$.

**Определение 3.** Граф зависимостей – граф, вершины которого соответствуют проектам, а дуги отражают зависимости между проектами – жесткие и мягкие. Граф зависимостей может иметь контуры.

Поскольку учет мягких зависимостей уменьшает время и затраты непосредственно следующего проекта, складывается впечатление, что целесообразно учитывать все мягкие зависимости. Однако это не так. Это следует из примера 1.

**Пример 1.**

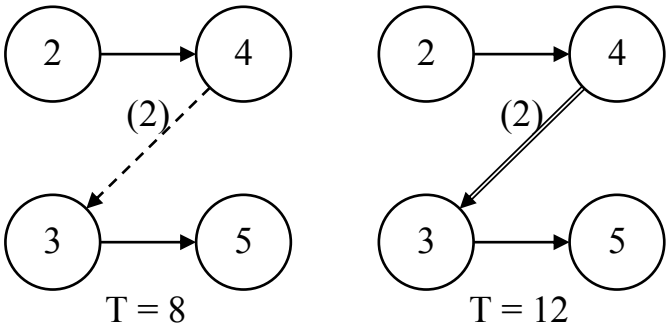



*Рис. 2. Графы зависимостей для примера 1*

В вершинах указаны продолжительности проектов. На дуге (4, 3) – уменьшение продолжительности проекта 3. Если не учитывать мягкую зависимость, проекты 2 и 4, а также 3 и 5 выполняются последовательно, а цепочки $2 \rightarrow 4$ и $3 \rightarrow 5$ – параллельно. Если учитывать (реализовывать) мягкую зависимость (4,3), то проект 3 выполняется после проектов 2 и 4, а продолжительность программы увеличивается на 4, несмотря на сокращение продолжительности проекта 3 на 2.

## 2. Определение множества выполняемых мягких зависимостей

**Задача.** Определить значения $x = \left(x_{ij}\right)$, при которых продолжительность программы минимальна.

### 2.1. Алгоритм решения задачи для графа без контуров

Пусть $P_i$ – множество жестких зависимостей, входящих в вершину $i$, нумерация проектов – правильная, т.е. для любой зависимости $(i, j)$ $j > i$. Рассматриваем проекты в порядке возрастания номеров.

1-*й шаг*. Положим

$$\lambda_1 = \tau_1. \tag{1}$$

*i-й шаг*. Вычисляем

$$\lambda_i = \min_{R_i} \left\lfloor \max_{k \in P_i \cup R_i} \lambda_k - A\left(R_i\right) + \tau_i \right\rfloor \tag{2}$$

*n-й шаг*. Определяем индекс вершины *n.*

$$\lambda_n = T_{min}. \tag{3}$$

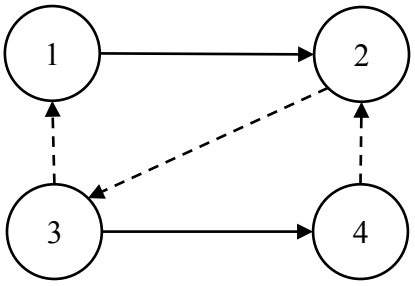
**Обоснование алгоритма.** Пусть все $R_i, i = \overline{1, n}$ определены. Тогда получаем обычную задачу с жесткими зависимостями и применяем алгоритм определения критического пути [6-9]. На каждом шаге алгоритма рассматриваем все варианты выбора $R_i$ и берем вариант, обеспечивающий минимальное время окончания соответствующей работы. Чтобы определить число рассматриваемых

вариантов для работы $i$, рассмотрим все предшествующие зависимости $k \in P_i \cup R_i$. Пронумеруем их по убыванию (невозрастанию) $\lambda_k$, т.е.

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m,$$

где $m$ – число зависимостей. Пусть $s$ – жесткая зависимость с максимальным значением $\lambda_s$ среди всех жестких зависимостей. Очевидно, что все мягкие зависимости, у которых $\lambda_t \leq \lambda_s$, $\lambda_t \in R_t$ выгодно выполнять. Поэтому рассматривать нужно мягкие зависимости, для которых $\lambda_t > \lambda_s$, $\lambda_t \in R_t$. Таким образом перебрать нужно $q+1$ зависимостей, где $q$ – число мягких зависимостей, для которых $\lambda_t > \lambda_s$. Если жестких зависимостей нет $(P_i = \varnothing)$, то рассматриваем вариант, когда ни одна мягкая зависимость не выполняется и $\lambda_i = \tau_i$.

**Пример 2.** Рассмотрим граф на Рис. 3.




$\tau_1$=3
$\tau_2$=2
$\tau_3$=7
$\tau_4$=5

*Рис. 3. Граф, иллюстрирующий алгоритм решения задачи*

В таблице 1 представлены продолжительности сокращения проектов.

*Таблица 1. Продолжительность сокращения проектов*

| $R_i$ | (1,3) | (2,3) | (2,4) | (1,3); (2,3) |
|---|---|---|---|---|
| $A(R_i)$ | 4 | 4 | 6 | 5 |

Вычисляем:

1 *шаг*.

$$\lambda_1=3$$

2 *шаг*.

$$\lambda_2=3+2=5$$

Для вершины 3 имеем две мягких зависимости – (1,3) и (2,3). Рассматриваем их в порядке убывания. Заметим, что если зависимость (2,3) выполняется, то выгодно выполнять и зависимость (1,3). Имеем в этом случае:

$$\lambda_3 = \max(\lambda_1,\lambda_2) + \tau_3 - A\big((1,3),(2,3)\big) = 5+7-5 = 7.$$

Рассматриваем зависимость (1,3). Имеем:

$$\lambda_3 = \lambda_1 + \tau_3 - A(1,3) = 3+7-4 = 6.$$

Наконец, если ни одна зависимость не выполняется, то $\lambda_3 = \tau_3 = 7$. Минимальное $\lambda_3 = 6$, т.е. выполняется зависимость (1,3).

3 *шаг*. Поскольку $\lambda_3 > \lambda_2$, зависимость (2,4) выгодно выполнять. Имеем:

$$\lambda_4 = \lambda_3 + \tau_4 - A(2,4) = 6+5-4 = 7.$$

То есть минимальная продолжительность программы равна 7, а проекты выполняются в следующей последовательности: сначала выполняется проект 1, затем параллельно проекты 2 и 3, а после завершения проекта 3 (к этому моменту проект 2 уже завершён) выполняется проект4.

## 2.2. Алгоритм решения задачи для графа зависимостей с контурами

**Утверждение 1.** Если вершина $i_1$ имеет минимальный индекс, то для этой вершины все входящие мягкие зависимости не выгодно выполнять.

*Доказательство.* Рассмотрим мягкую зависимость $(i, i_1)$. Имеем $\lambda_i \geq \lambda_{i_1}$. Более того, для любого графа $G_1$, включающего только реализуемые (жесткие) зависимости с индексами вершин $\lambda_i' \leq \lambda_i$ имеет

место $\lambda'_i \geq \lambda_1$. Действительно, пусть существует путь $\mu = (i_1, i_2, ..., i_k, i)$. Имеем $\lambda'_i > \lambda'_{i_k} > ... > \lambda'_{i_1} = \lambda_{i_1}$. Однако, для выполнения зависимости ($i$, $i_1$) необходимо, чтобы имело место $\lambda'_i < \lambda_{i_1}$.

**Описание алгоритма**

0 *шаг.* Определяем индексы без учета мягких зависимостей.

1 *шаг*. Пусть $i_1$ – вершина с минимальным индексом. В силу утверждения 1 исключаем все мягкие зависимости, входящие в неё. Определим все выполняемые (реализуемые) зависимости ($i_1$, $j$). Корректируем индексы всех вершин за исключением первой ($i_1$).

2 *шаг*. Определяем вершину с минимальным индексом (исключая вершину $i_1$). Обозначим ее номером $i_2$. В силу утверждения 1, все зависимости ($i$, $i_2$), $i \neq i_1$ не выгодно реализовывать. Определяем все реализуемые зависимости ($i_2$, $i$), $i \neq i_1$. Корректируем индексы вершин $i \neq i_1$, $i_2$.

*k-й шаг*. Определяем вершину $i_k$ с минимальным индексом (исключая вершины $i_1 - i_{k-1}$). В силу утверждения 1, все зависимости ($i$, $i_k$), $i \neq i_1$, $i_2$,…, $i_{k-1}$ не выгодно реализовывать. Определяем все реализуемые зависимости ($i_k$, $i$), $i \neq i_1$, $i_2$,…, $i_{k-1}$. Рассматривая зависимость ($i_k$, $i$), мы, естественно, добавляем все выполняемые зависимости ($j$, $i$), где $j < i_k$. Корректируем индексы вершин $i \neq i_1$, $i_2$,…, $i_k$.

*n-й шаг*. Определяем индекс вершины $i_n$.

**Пример 3**. Рассмотрим граф на рис. 4. Все зависимости мягкие.

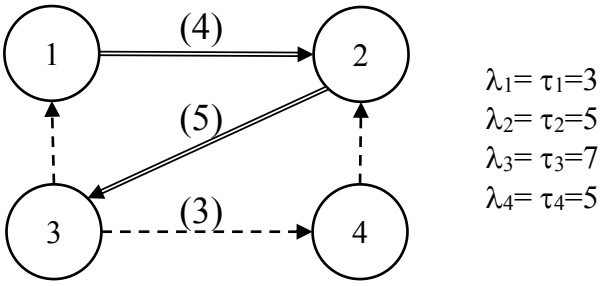



*Рис. 4. Граф с контурами, иллюстрирующий алгоритм решения задачи*

1 *шаг*. Берем вершину 1 с минимальным значением $\lambda_1$. Зависимость (3,1) не учитываем.

2 *шаг*. Полагаем $\lambda_2 = \lambda_1 + (\tau_2 - 4) = 3 + (5 - 4) - 4 = 4 < 5$. Берем вершину 2, как вершину с минимальным значением $\lambda$. Зависимость (4,2) не учитываем.

*3 шаг.* Полагаем $\lambda_3 = 4 + (7 - 5) = 6 < 7$.

*4 шаг.* Полагаем $\lambda_4 = 5$, поскольку при учете зависимости (3,4) имеем $\lambda_4 = \lambda_3 + (\tau_4 - 3) = 6 + (5 - 3) = 8 > 5$.

В итоге имеем последовательные проекты 1, 2 и 3 и выполняемый независимо от них проект 4. Время выполнения всей программы равно 6.

Рассмотрим более сложный пример – с наличием жестких зависимостей.

**Пример 4.** Сеть приведена на рис. 5.

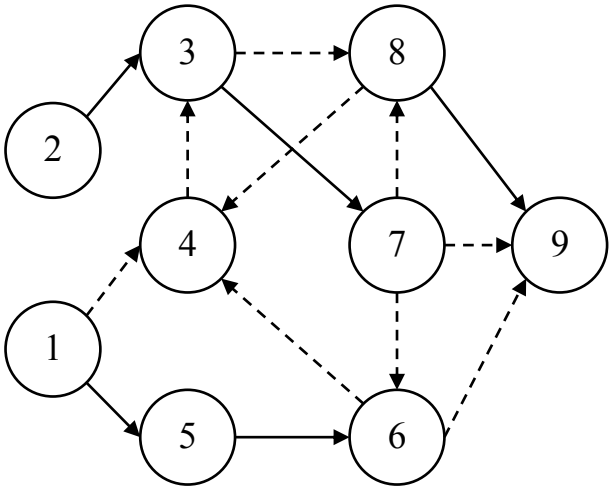



*Рис. 5. Граф примера 4*

Таблица 2 продолжительностей проектов приведена ниже.

0 *шаг.* Определяем индексы без учета мягких зависимостей. Заметим, что если не учитывать мягких зависимостей, то сеть на рис.5 имеет правильную нумерацию вершин.

$$\lambda_1 = \tau_1 = 3$$

*Таблица 2. Продолжительность проектов для примера 4*

| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| $\tau_i$ | 3 | 4 | 5 | 7 | 4 | 6 | 3 | 5 | 4 |

$$\lambda_2 = \tau_2 = 4$$
$$\lambda_3 = \lambda_2 + \tau_3 = 9$$
$$\lambda_4 = \tau_4 = 7$$
$$\lambda_5 = \lambda_1 + \tau_5 = 7$$
$$\lambda_6 = \lambda_5 + \tau_6 = 13$$
$$\lambda_7 = \lambda_3 + \tau_7 = 12$$
$$\lambda_8 = \tau_8 = 5$$
$$\lambda_9 = \tau_9 + \max(\lambda_7; \lambda_8) = 16$$

1 *шаг.* Определяем минимальный индекс $\lambda_1 = 3$. Пусть $A_4((1,4)) = 6$. Вычисляем:

$$\lambda_4 = \min(\tau_4; \lambda_1 + \tau_4 - A_4((1,4))) = 4.$$

Зависимость (1,4) учитываем.

2 *шаг.* $\lambda_2 = \tau_2 = 4$.

3 *шаг.* Следующая вершина с минимальным индексом – это вершина 4, $\lambda_4 = 7$. Зависимости (6,4) и (8,4) не учитываем. Корректируем индекс $\lambda_3$. Пусть $A_3((4,3)) = 3$. Вычисляем:

$$\lambda_3 = \min(9; 4 + 5 - 3) = 6.$$

Зависимость (4,3) учитываем.

4 *шаг.* Вершина с минимальным индексом – это вершина 8, $\lambda_8 = 5$. Зависимости (3,8) и (7,8) не учитываем. На рисунке 6 представлена сеть после четырех шагов алгоритма (зависимости, которые не учитываются, удалены).

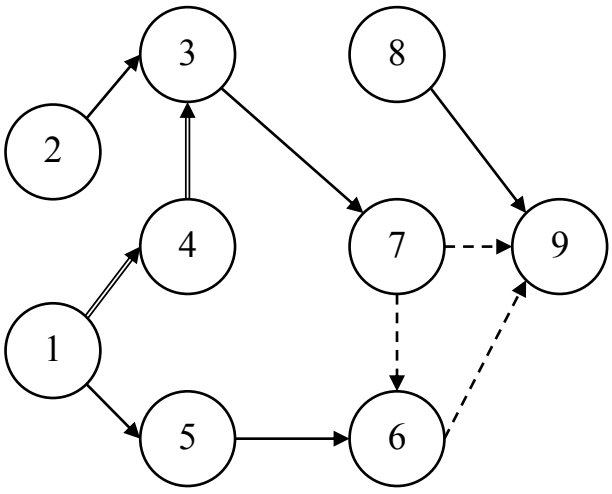



*Рис. 6. Промежуточный граф примера 4*

5 *шаг.* Рассматриваем вершину 9, $\lambda_9 = 9$. Зависимости (6,9) и (7,9) не учитываем.

6 *шаг.* Рассматриваем вершину 7, $\lambda_7 = 12$. Пусть $A_6((7,6)) = 3$. Зависимость (7,6) не учитываем.

Окончательный вид графа представлен на рисунке 7. Продолжительность программы T = 13.

## 5. Заключение

В работе рассмотрена задача определения минимальной продолжительности программы при наличии мягких зависимостей между проектами. Предложен алгоритм определения критического пути.

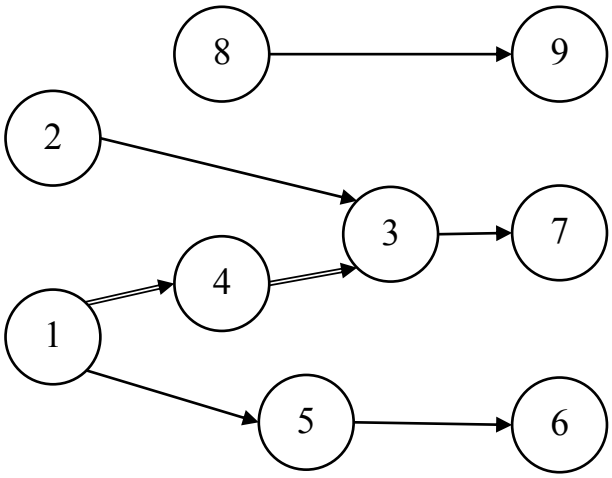



*Рис. 7. Окончательный граф примера 4*

Представляет интерес постановка задач, когда выполнение мягких зависимостей приводит к снижению затрат непосредственно следующего проекта. Эти задачи требуют дальнейших исследований.

Одна из задач связана с определением выполняемых мягких зависимостей, при которых может быть достигнута максимальная экономия затрат. Заметим, что если в графе зависимостей отсутствуют контуры, то решение очевидно, нужно выполнять все мягкие зависимости. В случае наличия контуров задачу следует рассматривать как сложную задачу дискретной оптимизации, не имеющую сегодня эффективных методов решения. В случае, если снижение затрат при наличие нескольких зависимостей, входящих в одну вершину, равно сумме снижений затрат от каждой зависимости, задачу можно свести к задаче о покрытии двудольного графа [6]. При этом для получения оценок можно применить метод сетевого программирования [10-11].

Ещё одна задача связана с учетом и снижения продолжительности проектов, и снижения затрат при наличии мягких зависимостей. Одну из постановок задач можно описать следующим образом: определить множество мягких зависимостей, при которых снижение затрат максимально при ограничении на время выполнения программы. Здесь предполагается поиск различных частных случаев, допускающих эффективное решение.

Все перечисленные задачи требуют дальнейших исследований.

## Литература


1. Управление проектами: основы профессиональных знаний, национальные требования к компетентности специалистов (NCB – SOVNET National Competence Baseline Version 3.1) – М.: ЗАО «Проектная ПРАКТИКА», 2014 – 259 с.
2. *Клиффорд Ф. Грей, Эрик У. Ларсон.* Управление проектами: Практическое руководство. – М.: «Дело и Сервис», 2003.
3. *Бурков В.Н., Новиков Д.А.* Как управлять проектами. – М.: СИНТЕГ-ГЕО, 1997.
4. *Баркалов С.А., Бурков В.Н.* Математические основы управления проектами: Учебн. пособие / Баркалов С.А., Бурков В.Н., Буркова И.В., Воропаев В.И., Секлетова Г.И. и др. Под ред. Буркова В.Н. – М.: Высш. шк., 2005. – 423 с.
5. *Баркалов С.А., Буркова И.В., Колпачев В.Н., Потапенко А.М.* Модели и методы распределения ресурсов в управлении проектами. М.: ИПУ РАН, 2004. - 85 с.
6. *Бурков В.Н., Заложнев А.Ю., Новиков Д.А.* Теория графов в управлении организационными системами. М.: Синтег, 2001. – 124 с.
7. *Емеличев В.А., Мельников О.И,, Сарванов В.И., Тышкевич Р.И.* Лекции по теории графов. М.: Наука, 1990. – 384 с.
8. *Гимади Э.Х., Рыков И.А.* Асимптотически точный подход к приближенному решению некоторых задач покрытия графа // Тр. Ин-та математики и механики УрО РАН. 2015. Vol. 21, № 3. С 89-99.
9. *Хачай М.Ю., Незнахина Е.Д.* Полиномиальная приближенная схема для евклидовой задачи о цикловом покрытии графа // Тр. Ин-та математики и механики УрО РАН.2014. Vol. 20, № 4. С. 297-311.
10. *Буркова И.В., Порядина В.Л., Кашенков А.Р.* Алгоритм реализации и решения двойственной задачи методом сетевого программирования // Научный вестник Воронежского государственного архитектурно-строительного университета. Серия: Управление строительством. 2013. № 1 (4). С. 17-22.
11. *Буркова И.В.* Метод сетевого программирования в задачах нелинейной оптимизации // Автоматика и телемеханика. 2009. № 10. С. 15-21.