Е.Р. Гафаров, Москва, ВЦ РАН УДК 519.854.2

Задача построения расписания проекта в программном продукте 1С:Управление строительной организацией

Аннотация

В данной статье приводится метаэвристический алгоритм "Муравьиные колонии" решения классической задачи теории расписаний RCPSP (построение расписания проекта с учетом ограничения на ресурсы). Описаны особенности реализации алгоритма в рамках программного продукта 1С:Управление строительной организацией.

Введение

В декабре 2006 года готовится к выпуску программный продукт 1С:Управление строительной организацией (1С:УСО). Цель создания программы – комплексная автоматизация основных подразделений строительной компании. В 1С:УСО в единой системе объединены модули: управление финансами, бухгалтерским учетом, закупками, запасами и т.д. Ядро системы – модуль "управление строительным производством", в котором реализованы базовые принципы управления проектами с учетом специфики строительного производства.

В данной статье приводятся основные принципы управления проектами, а так же рассмотрена задача построение расписания проекта с учетом ограничений на ресурсы.

1 Управление проектами

Достаточно широкое описание проблемы "управления проектами" приведено в работе [1]. В этом параграфе в сжатом виде приводятся

особенности планирования проектов с учетом специфики строительства.

Проект – совокупность взаимосвязанных действий, направленных на достижение конкретных целей. К примеру, проект "строительство дома" может состоять из работ "выемка грунта", "укладка фундамента", "возведение стен первого этажа"и т.д. Очевидно, что по технологии "укладка фундамента" производится после "выемки грунта". В этом случае говорят, что между работами существует отношение предшествования.

Зачастую детальный состав работ нам неизвестен вначале описания проекта. Как следствие возникает необходимость планирования "поэтапно". Более того, каждый этап может быть своего рода отдельным проектом.

В системе 1С:УСО консолидируется информация о проектах как самого холдинга, так и его подразделений. К примеру, проект "Строительство микрорайона" ведется в проектном отделе "головной организации"и состоит из работ "Строительство дома №1", "Строительство дома №2", "Строительство магазина", "Прокладка дорог". В свою очередь "Строительство дома №1" является подпроектом, который планируется и выполняется обособленным подразделением компании.

Подробнее опишем характеристики работ и взаимосвязи между ними.

Характеристики работы (операции).

Тип работы:

Продолжительность. Это классический тип работы. Важная ее характеристика – продолжительность выполнения;

Контрольное событие. К примеру – составление отчетности, разовый контроль. Такая работа имеет нулевую продолжительность;

Гамак. Работа выполняется строго между моментами времени окончания работ-предшественников и началом работ-

последователей;

Продолжительность. Задается в днях или часах. На практике редко планирование ведется в меньших единицах времени. Очевидно, что продолжительность может зависеть от интенсивности работы (т.е. от количества и качества привлеченных ресурсов). Для составления различных вариантов выполнения проекта в 1С:УСО предусмотрено ведение нескольких сценариев проекта;

Невозобновимые ресурсы. К примеру, материалы необходимые для выполнения работы;

Возобновимые ресурсы. Техника, персонал. Ресурсы могут объединятся в "бригады", которые выступают неделимым ресурсом. Зачастую у разных ресурсов, задействованных на работе, бывает разный график работы. Это следует учитывать, при составлении календарного плана проекта;

График работы. Разные работы могут выполнятся с разным графиком. К примеру, основные монтажные работы в 2 смены 7 дней в неделю, а вспомогательные работы - только в 1 смену и только в первую половину недели;

Ограничение на время выполнения. Параметр принимает значения "Как можно раньше", "Как можно позже", "Фиксированная дата", "Начало не позже", "Начало не раньше".

Прочие характеристики. Объем работ, Стоимость, Подразделение/контрагент – исполнитель работы.

Характеристики взаимосвязей между работами.

Обыденной представляется взаимосвязь типа "Окончание-Начало", то есть работа-последователь начнется не раньше, чем закончится "работа-предшественник". Выделяют связи "Начало-Начало"(когда работа-последователь не может начаться раньше, чем начнется работа-предшественник), "Окончание-Окончание", "Начало-Окончание".

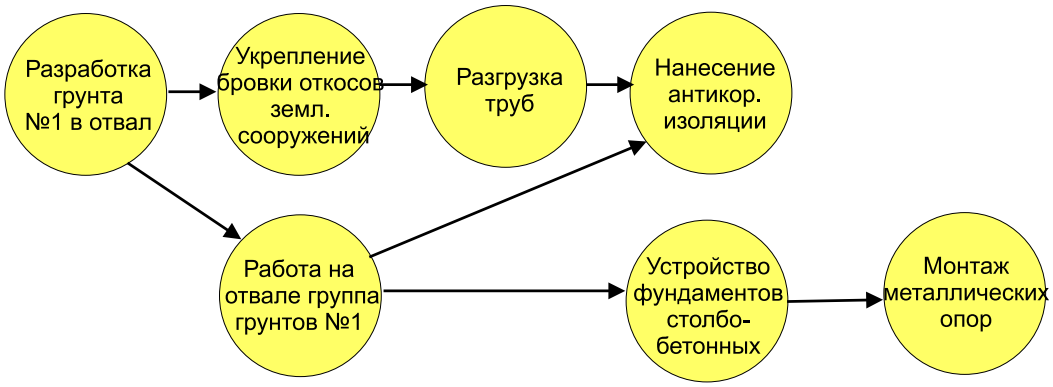

Рис. 1: Сетевой график. Пример из 1С:УСО.

Более того, возможно по технологии между окончанием работы-предшественника и началом работы-последователя должна быть задержка по времени (или по объему, если связь между работами "Начало-Начало").

Интуитивно понятным кажется представление проекта в виде сетевого графика, где узлы – работы, а дуги между узлами – взаимосвязи (см. рис. 1).

2 Составление расписания проекта

В 1С:УСО автоматически решается важная задача составления расписания проекта (расписания выполнения работ проекта). Расписания для проектов, состоящих из небольшого количества работ, можно составить и "вручную". Для реальных же проектов, в которых работы исчисляются сотнями, автоматизация составления расписания незаменима.

Можно составлять расписание без учета доступных ресурсов, но при учете количества доступных ресурсов задача составления расписания усложняется. Более того, здесь возникает проблема построения оптимального расписания проекта с учетом ограничения на ресурсы (Resource Constrained Project Scheduling Problem). В данной статье рассматривается задача с целевой функцией C_{max} – минимизация общего времени выполнения проекта. При этом будем учитывать только

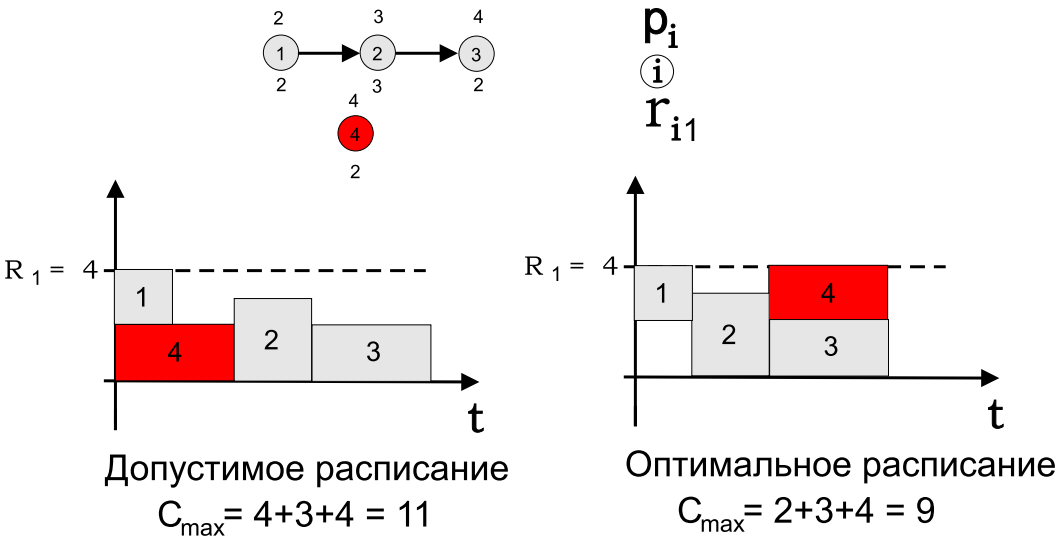

Рис. 2: Пример задачи оптимизации расписания.

возобновимые ресурсы.

Пример. Рассмотрим пример, представленный на рис. 2. Проект состоит из 4-х работ. На сетевом графике сверху над узлами указана продолжительность работ p_i, снизу – необходимое для выполнения работы количество r_{i1} возобновимого ресурса 1. Всего доступно 4 единицы ресурса 1 (к примеру, доступно 4 монтажника).

На рисунке представлено 2 допустимых расписания (при которых соблюдены отношения предшествования, ограничения на ресурсы и продолжительность требований).

Алгоритм List Scheduling (LS) составления допустимого расписания проекта можно описать следующим образом:

Обозначим через EL (Eligible List) – список "доступных" работ не включенных в расписание (для которых не определен период выполнения, но все предшественники расставлены).

Алгоритм LS.

Шаг 1. Поместим в EL те работы, для которых нет предшественников.

Шаг 2. **ПОКА** $EL \neq \emptyset$ **ЦИКЛ**

- Выберем работу $j \in EL$.

- Рассчитаем для работы j самый ранний период выполнения $[t_1, t_2)$. При этом учитываем продолжительность работы ($t_2 - t_1$ больше или равно продолжительности), указанный график работы, доступность ресурсов. То есть в каждый момент времени $t \in [t_1, t_2)$ работа ресурсами обеспечена полностью.
- Резервируем ресурсы на интервале $[t_1, t_2)$ для работы j, то есть уменьшаем количество доступных ресурсов.
- Убираем работу j из списка "доступных" работ. $EL = EL \setminus \{j\}$.
- В EL добавляем последователей работы j, для которых все предшественники расставлены.

КОНЕЦ ЦИКЛА.

Продемонстрируем работу алгоритма на примере рис. 2.
Шаг 1. $EL = \{1, 4\}$
Шаг 2. Пусть $j = 1$. Самое раннее время выполнения работы 1: $[0, 2)$. $EL = \{4, 2\}$;
Шаг 2. Пусть $j = 4$. Самое раннее время выполнения работы 4: $[0, 4)$. $EL = \{2\}$;
Шаг 2. $j = 2$. Самое раннее время выполнения работы 3: $[4, 7)$, так как на интервале $[2, 4)$ для работы 2 уже не хватает ресурсов (зарезервированы под работу 4). $EL = \{3\}$;
Шаг 2. $j = 3$. Самое раннее время выполнения работы 3: $[7, 11)$, так как работа 2 заканчивается в момент времени 7. $EL = \emptyset$

В итоге мы получили расписание при котором время выполнения проекта $C_{max} = 11$. Мы можем представить последовательность постановки работ в расписание как перестановку $(1, 4, 2, 3)$.

Если бы мы ставили работы в порядке $(1, 2, 4, 3)$ или $(1, 2, 3, 4)$, то получили бы расписание, при котором $C_{max} = 9$, то есть расписание, соответствующее перестановке $(1, 2, 4, 3)$ лучше с точки зрения целевой функции.

Определение. *Активным назовем то допустимое расписание, при котором каждая работа выполняется как можно раньше.*

В результате работы алгоритма LS строятся только активные расписания. Вполне очевидна следующая теорема.

Теорема 1 [2] *Оптимальное решение задачи построения расписания проекта с учетом ограничения на ресурсы следует искать среди активных расписаний.*

Каждому активному расписанию соответствует некая перестановка $(j_1, j_2, \ldots, j_n)$ – последовательность постановки работ в расписание. То есть задача построения оптимального расписания сводится к задаче нахождения оптимальной перестановки.

3 Математическая постановка проблемы RCPSP

Дано множество работ $N = \{i = 1, \ldots, n\}$ и r возобновимых ресурсов $k = 1, \ldots, r$. В каждый момент времени t доступно R_k единиц ресурса k. Заданы продолжительности обслуживания p_i для каждого требования $i = 1, \ldots, n$. Во время обслуживания требования i требуется r_{ik} единиц ресурса k.

Между некоторыми парами требований заданы ограничения предшествования: $i \rightarrow j$ означает, что обслуживание требования j начинается не раньше окончания требования i. Обозначим множество отношений предшествования $C = \{i \rightarrow j | i, j \in N\}$.

Обслуживание требований начинается в момент времени $t = 0$. Прерывание требований запрещены.

Необходимо определить моменты времени начала обслуживания требований $S_i,\ i = 1, \ldots, n$ так, чтобы:

1. в каждый момент времени t "занятое" количество ресурса k было меньше или равно R_k

2. не нарушались отношения предшествования, то есть $S_i + p_i \leq S_j$, если $i \to j$

3. минимизировать время выполнения проекта

$$C_{max} = \max_{i=1,\ldots,n} \{C_i\}, C_i = S_i + p_i.$$

Задача в такой постановке является упрощенным вариантом практической задачи. В частности, есть только один тип взаимосвязей "Окончание-Начало", не учитывается график выполнения работы, задержки между работами и т.д.

Показано, что взаимосвязи типа "Начало-Начало", "Начало-Окончание", "Окончание-Окончание" можно преобразовать к связи типа "Окончание-Начало" введением фиктивных требований. Существует постановка задачи, при которой учитывается задержка между выполнением работ d_{ij}.

На практике количество выделенных на проект ресурсов может меняться со временем (к примеру, в апреле у нас доступно 5 экскаваторов, в июне – только 3, а с 12 августа – 7 экскаваторов). В соответствии с практической задачей будем считать, что величины R_k зависят от времени t, т.е. $R_k(t)$.

Несмотря на некоторое упрощение модели, приведенные алгоритмы и свойства задачи RCPSP актуальны и для исходной практической задачи.

Решение задачи RCPSP представляется в виде набора $S = (S_1, \ldots, S_n)$. В следующей теореме доказано, что допустимому решению $(S_1, \ldots, S_n)$ соответствует некоторая перестановка $\pi = (j_1, \ldots, j_n)$.

Теорема 2 *Решение задачи однозначно представляется в виде перестановки* $\pi = (j_1, \ldots, j_n)$ *из* n *требований.*

Для каждого требования i определим временное окно $[r_i, d_i]$, в котором требование i должно быть обслужено при любом допустимом расписании S, $C_{max}(S) \leq UB$, где UB – некоторая верхняя граница для C_{max}:

$$r_0 = 0;\ r_i = \max_{j|j\to i}\{r_j + p_j\},\ i = 1, \ldots, n+1,$$

$$d_{n+1} = UB;\ d_i = \min_{j|i\to j} d_j - p_j,\ i = n, n-1, \ldots, 0.$$

Мы можем представить структуру проекта как *требования-в-вершинах* ориентированного графа $G = (V, A)$, где список вершин $V = \{1, \ldots, n\}$ соответствует списку требований, а множество дуг $A = \{(i, j)|i, j \in V; i \to j\}$ соответствует ограничениям предшествования.

Зачастую, чтобы граф проекта был связным, добавляют 2 "пустых" требования 0 и $n+1$, соответствующие началу и окончанию проекта. $p_0 = p_{n+1} = 0$. Добавляются ограничения предшествования $0 \to i, i \to n+1,\ i = 1, \ldots, n$.

4 Алгоритмы решения проблемы RCPSP

Далее приводится алгоритм диспетчеризации с учетом введенных обозначений.

Алгоритм List Scheduling

1: Пусть EL – список всех работ без предшественников;
2: **while** $EL \neq \emptyset$ **do**
3: Выберем работу $j \in EL$;
4: $t := max_{i\to j \in C}\{S_i + p_i\}$;
5: **while** Существует ресурс k, что $r_{jk} > R_k(\tau)$ для некоторого $\tau \in [t, t + p_j)$ **do**
6: Вычислим минимальное значение $t_k > t$, что работа j может быть выполнена на интервале $[t_k, t_k + p_j)$, если рассматривается только ресурс k;
7: $t := t_k$;
8: **end while**
9: Назначим выполнение работы j на интервал $[S_j, C_j) = [t, t + p_j)$;

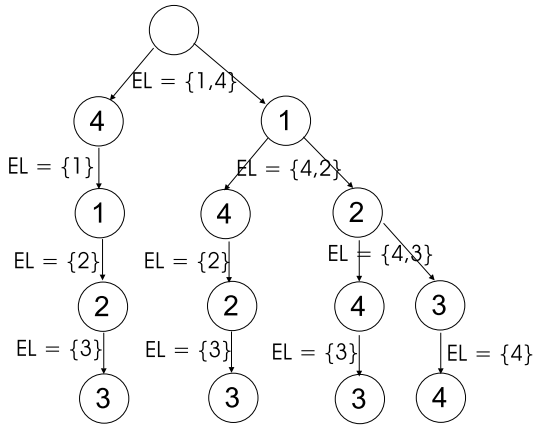

Рис. 3: Дерево поиска оптимального расписания.

10: Резервируем ресурсы под работу $R_k(\tau) = R_k(\tau) - r_{jk},\ k = 1, \ldots, r,\ \tau \in [t, t + p_j)$;

11: $EL = EL \setminus \{j\}$. Добавляем в EL последователей требования j, для которых все предшественники расставлены;

12: **end while**.

Полученное решение зависит от выбора работы j на шаге 3 алгоритма.

На основании данного алгоритма диспетчеризации можно построить точный алгоритм ветвей и границ. Ветвление в котором происходит при выборе работы j. Схема ветвления данного алгоритма для примера рис.2 представлена на рис.3.

Задача RCPSP NP-трудна в сильном смысле [2]. Стоит отметить, что к настоящему моменту ни один точный алгоритм для задачи RCPSP за приемлемое время не может решить любой пример даже когда $n = 60$.

Актуальным становится построение эффективных нижних и верхних оценок, разработка гибридных алгоритмов с применением эвристик.

В продукте 1С:УСО реализован алгоритм решения задачи Ant Colony Optimization (ACO) [3]. Этот итерационный алгоритм основан на идее последовательного приближения к оптимальному решению.

Каждая итерация – "запуск искусственного муравья", который "пытается" по некоторому правилу выбрать наилучший маршрут к "пище" (к оптимуму функции), используя метки своих предшественников.

Каждый муравей выполняет алгоритм List Scheduling, выбирая требование $j \in EL$ исходя из значений параметров:

а) η_{ij} – эвристическая информация о том, насколько хорошим кажется постановка работы j на место i в перестановке $(j_1, j_2, \ldots, j_n)$ (напомним, что каждому активному расписанию соответствует некая перстановка). То есть насколько хорошим кажется выбор работы $j \in EL$ на i-ом срабатывания цикла. Этот параметр можно вычислить следующим образом:

1. По правилу d: $\eta_{ij} = \frac{1}{d_j},\ i = 1, \ldots, n$;
2. По правилу $CriticalPatch$ (критический путь): $\eta_{ij} = d_j - r_j,\ i = 1, \ldots, n$;

б) τ_{ij} – "след" (в природе: след феромона). После каждой итерации этот параметр корректируется. Параметр показывает насколько "хорошим" для работы j оказалась позиция i в перестановке $(j_1, j_2, \ldots, j_n)$. То есть это накопленная статистическая информация о качестве выбора для позиции i работы j, в то время как η_{ij} характеризует предполагаемую выгоду такой постановки при недостатке накопленной информации.

Под эвристической процедурой понимается процедура, не имеющая формального обоснования, а опирающаяся лишь на анализ специфики структуры задачи и на связанные с ней содержательные соображения [4].

Параметры η_{ij} рассчитываются один раз перед первой итерацией.

На каждом шаге вычисляется **Матрица вероятностей перехода**:

$$\rho_{ij} = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{h \in EL} [\tau_{ih}]^\alpha [\eta_{ih}]^\beta}, & j \in EL, \\ 0, & j \notin EL. \end{cases}$$

Правило, по которому на позицию i выбирается требование j определяется следующим образом:

$$\begin{cases} j = argmax_{h \in EL}[\tau_{ih}]^{\alpha}[\eta_{ih}]^{\beta}, & q < q_0, \\ j \text{ определяется случайным образом,} & \\ \text{согласно распределению вероятностей } \rho_{ij}, & q \geq q_0, \end{cases}$$

где $0 \leq q_0 \leq 1$ – параметр алгоритма, а значение q вычисляется случайным образом на каждом шаге.

После того, как работа j было поставлено на позицию i в перестановке, пересчитывается "локальный след":

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho\tau_0,$$

где $\tau_0, \rho \in [0, 1]$ – некие параметры алгоритма.

После каждой итерации "глобальный след" τ_{ij} корректируется по правилу:

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho/C^*_{max},$$

если в "лучшем" найденном расписании на позиции i перестановки находится работа j. Иначе

$$\tau_{ij} = (1 - \rho)\tau_{ij},$$

Значение C^*_{max} – время выполнения проекта для "лучшего" найденного расписания.

Алгоритм ACO дает хорошие результаты при интеграции в него локального поиска, к примеру, попарной перестановки требований (если при этом не нарушаются отношения предшествования). Локальный поиск можно запускать после каждой итерации.

5 Алгоритм ACO в рамках 1С:УСО

В программном продукте 1С:УСО реализован алгоритм ACO. У пользователя есть возможность настроить параметры алгоритма, выбрать эвристики для расчета η_{ij} определить влияние каждой эвристики на

вероятностей перехода ρ_{ij}. В алгоритме предлагается на выбор 6 эвристик $\eta_{ij}^1,\ \eta_{ij}^2,\ \eta_{ij}^3,\ \eta_{ij}^4,\ \eta_{ij}^5,\ \eta_{ij}^6$. Параметры ρ_{ij} вычисляются следующим образом:

$$\rho_{ij} = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}^1]^{\beta^1} [\eta_{ij}^2]^{\beta^2} [\eta_{ij}^3]^{\beta^3} [\eta_{ij}^4]^{\beta^4} [\eta_{ij}^5]^{\beta^5} [\eta_{ij}^6]^{\beta^6}}{\sum_{h \in EL} [\tau_{ih}]^\alpha [\eta_{ih}^1]^{\beta^1} [\eta_{ih}^2]^{\beta^2} [\eta_{ih}^3]^{\beta^3} [\eta_{ih}^4]^{\beta^4} [\eta_{ih}^5]^{\beta^5} [\eta_{ih}^6]^{\beta^6}}, & j \in EL, \\ 0, & j \notin EL, \end{cases}$$

У пользователя есть возможность настроить параметры $\alpha, \beta^1, \beta^2, \beta^3, \beta^4, \beta^5, \beta^6, \rho, m, q_o$. То есть можно настроить количество муравьев и количество запусков алгоритма ACO. В процессе работы алгоритма выводятся сообщения о построенных расписаниях и найденных значениях C_{max}. В любой момент пользователь может прерваться и сохранить полученное расписание или перенастроить параметры алгоритма.

На демонстрационном примере алгоритм ACO нашел расписание, при котором $C_{max} = 4{,}5$ месяца, в то время, как алгоритм диспетчеризации строит расписание, при котором $C_{max} = 6$ месяцев.

Стоит отметить, что в основе алгоритма ACO лежит алгоритм диспетчеризации List Scheduling, в котором можно учесть специфику практической задачи (графики выполнения работ, ограничение на время выполнения, задержки между работами). Этим алгоритм ACO выгодно отличается от точных алгоритмов решения релаксированной задачи RCP-SP.

Тем не менее алгоритм не оправдывает ожидания на некоторых классических тестовых примерах. Планируется построить гибридный алгоритм, объединив идеи алгоритма ACO и алгоритм ветвей и границ [5].

Список литературы

[1] **Бурков В.Н., Буркова И.В., Горгидзе И.А., Джавахадзе Г.С., Хуродзе Р.А., Щепкин А.В.**, *Задачи управления в социальных и экономических системах*// Синтег, Москва, 2005, 256 стр.

[2] **Brucker P., Knust S.**, *Complex scheduling*// Springer, 2006.

[3] **Merkel D., Middendorf M., Schmeck H.**, *Ant Colony Optimization for Resource-Constrainted Project Scheduling*// IEEE Transactions on Evolutionary Computation, vol.6, NO 4, 2002, pp. 333-346

[4] **Корбут А.А., Финкельштейн Ю.Ю.**,*Приближенные методы дискретного программирования* // Известия АН СССР. Технич. кибернетика, 1. 1983. P. 165–176.

[5] **Гафаров Е.Р.**,*Гибридный алгоритм решения задачи минимизации суммарного запаздывания для одного прибора* // Информационные технологии, №1, 2007, стр. 30-37.