DOI: 10.25728/mlsd.2022.0451

ОРГАНИЗАЦИЯ ПРОИЗВОДСТВА НА ОСНОВЕ АУКЦИОНОВ

Вахранев А.В., Гасанов И.И.
ФИЦ ИУ РАН,
Россия, г.Москва, ул. Вавилова, д. 40
anton22255@ya.ru, gasanov48@bmail.ru

Аннотация: Рассматриваются алгоритмы сетевой организации взаимодействия ассоциации предприятий, занимающихся производством. Ранее была предложена схема такого взаимодействия по принципу множественных аукционов, посредством которых определяется распределение функций исполнения заказа на производство. Приводятся принципиальные схемы в форме кодов для реализации предложенной схемы имитационных процедур.

Ключевые слова: планирование производства, сетевое взаимодействие, коды блоков, аукцион.

Введение

Прогресс в информационных технологиях ставит перед управлением предприятиями новые задачи, реализация которых невозможна без четко сформулированных целей и применения инновационных методов управления. В работе [1] сформулирован оригинальный подход для решения задачи организации взаимодействий группы предприятий в рамках производственных процессов в сетевой постановке при условии заданного технологического графа и экономической свободы предприятий. Предложена аукционная схема. Настоящая работа носит подготовительный характер: сформулирована исходная оптимизационная модель, описаны принципиальные блоки программного комплекса и приводятся их реализация в форме кодов.

1 Пример задачи

Задачи поиска параметров оптимального производственного процесса всегда привлекают большой интерес исследователей, что подтверждают работы [2–5]. Рассмотрим, в частности, модель производственной линии из работы [3], которая выполняет задачи по сборке изделий. Производство складывается из отдельных задач, которые выполняют обособленные производители, именуемые в изложении станциями. Связь между задачами задается при помощи производственного графа:

$$PG = \{V, E, f_{spatial}, f_{incompatible}\}, \quad (1)$$

где V и E представляют наборы вершин и ребер соответственно. Если существует направленное ребро $e =< a, b >$, то задача a является непосредственным предшественником задачи b. $f_{spatial}(V_s)$ это бинарный признак: если $f_{spatial}(V_s) = 1$, то существует пространственное ограничение для задач сборки в наборе вершин V_s; в противном случае его нет. $f_{incompatible}$ представляет собой несовместимые ограничения: если $f_{incompatible}(V_c) = 1$, то задача удовлетворяет несовместимому ограничению; в противном случае такого ограничения нет. $V_s, V_c \in V$.

Запишем задачу в терминах модели [3]:
индексы:

- i, j – индексы задач;
- k – индекс станции;
- n – количество задач;
- p – уровень компетентности оператора (1- низкий, 2 – средний, 3- высокий).

параметры:

- C – время цикла;
- m – количество станций;
- $C_{\max}$ – максимальное время цикла;
- $C_{\min}$ – минимальное время цикла;
- t_{ip} – время, необходимое для выполнения задачи i, когда уровень квалификации оператора равен p;

- P_{ij} – ограничения приоритета между задачами i и j. $P_{ij}=1$ указывает, что есть ограничения приоритета; в противном случае ограничения приоритета нет;
- SC_{ij}^{+} – пространственное ограничение между задачами i и j. $SC_{ij}^{+}=1$ указывает, что задачи должны быть назначены одной и той же станции; в противном случае это не требуется;
- SC_{ij}^{-} – несовместимое ограничение между задачами i и j. $SC_{ij}^{-}=1$ указывает, что задачи не могут быть назначены одной и той же станции; в противном случае могут;
- R_p – количество операторов с уровнем квалификации p;
- $O_{\max}$ – максимальное количество задач, которые могут быть размещены на станции

переменные:

- $x_{ik}\in\{0,1\}$, $x_{ik}=1$ указывает, что задача i назначена станции k;
- $r_{ip}\in\{0,1\}$, $r_{ip}=1$ указывает, что операторы с уровнем квалификации p назначены на задачу i;
- $y_k\in\{0,1\}$, $y_k=1$ означает, что станция k включена;

Постановка задачи оптимизации:
найти:

$$\min\sum_{k=1}^{m} y_k\,, \tag{2}$$

при условиях:

$$\sum_{i=1}^{n} x_{ik}r_{ip}t_{ip}\le Cy_k\,,\forall k=1,2,..m,\forall p=1,2,3 \tag{3}$$

$$\sum_{k=1}^{m} x_{ik}=1,\forall i=1,2,..n, \tag{4}$$

$$\sum_{k=1}^{m} k\times x_{ik}\le\sum_{k}^{m} k\times x_{jk},\forall P_{ij}=1,\forall i=1,2,..n,\forall j=1,2,..n \tag{5}$$

$$y_{k+1}\le y_k,\forall k=1,2,..m \tag{6}$$

$$x_{ik}=x_{jk},\forall(i,j)\in SC_{ij}^{+},\forall k=1,2,..m \tag{7}$$

$$x_{ik}+x_{jk}\le 1,\forall(i,j)\in SC_{ij}^{-},\forall k=1,2,..m \tag{8}$$

$$\sum_{i=1}^{n} x_{ik}\le O_{\max},\forall k=1,2,..m, \tag{9}$$

$$\sum_{i=1}^{n} r_{ip}\le R_p,\forall p=1,2,3 \tag{10}$$

2 Сетевое производство

Отличие предлагаемой в работе [1] задачи состоит в учёте интересов отдельных производителей и организации аукционов, проводимых на этапе планирования. В данной работе рассматривается сетевая форма взаимодействия внутри ассоциации предприятий, занимающихся некоторым общим производством. Предполагается, что ассоциация занимается производством, монтажом, сборкой изделий из некоторого фиксированного набора. Выполнение заказа на производство распределяется внутри ассоциации.

Запишем задачу в терминах модели [1]:
индексы:

- j,l,r – индексы предприятий;
- n – индекс изделия;

параметры:

- $\widehat{P} = \{m\}_{m=1}^{M^0}$ – множество изделий: m – индекс изделия;
- L – множество участников;
- $A = (a_{k,r}^n)$ – матрица поставок, где $a_{k,r}^n$ – поставка изделия n от предприятия k предприятию r;
- $T_l = \left(t_{l,r}^n\right), n \in M^0$ – матрица транспортных расходов, где $t_{l,r}^n$ – стоимость доставки изделия n от предприятия l к предприятию r;
- $W = \{m_n\} \subset \widehat{P}, \overline{W} = \{w_n\}$ – объем изделий из W, где w_n – количество изделий m_n;
- $G(n) \subset \widehat{P}$ – множество деталей, из которых монтируется изделие n;
- $W \in \widehat{P}$, $G(W) = \bigcup_{n \in W} G(n)$ – набор деталей, из которых монтируются изделия из W;
- $\overline{G}(\overline{W})$ – вектор комплектующих, необходимых для сборки всех компонент вектора $\overline{W}$;
- $\overline{S}_l = (s_1^l, s_2^l, ..., s_P^l)$ – ресурсы предприятия l, где s_1^l – финансовые ресурсы предприятия;
- $Y = (y_{j,l}^n)$, $n \in M^0$, $j, l \in L$ – матрица поставок
- $\overline{\Psi}^l(\overline{W}) : \square^{M^0} \to \square^p$ – функция расходов ресурсов предприятия l при производстве продукции $\overline{W}$.

Запишем задачу оптимизации в виде.
Найти

$$\psi_1^l\left(\sum_{l \in L} \overline{Y}_{j,l}\right) + \sum_{n \in M^0} \sum_{j \in L} \sum_{l \in L} t_{j,l}^n \cdot y_{j,l}^n \to \min \qquad (11)$$

при условии:
$\forall l : \overline{G}(\overline{W}_l) = \sum_{j \in L} \overline{Y}_{j,l}$ – балансовое ограничение, где $\overline{Y}_{j,l}$ – вектор поставок изделий от участника j участнику l, $\overline{W}_l$ – это вектор продукции, производимый участником l;

$\forall j : \overline{\Psi}^j\left(\sum_{l \in L} \overline{Y}_{j,l}\right) \le \overline{S}_j$ – ресурсное ограничение, где $\sum_{l \in L} \overline{Y}_{j,l}$ – это вектор изделий, произведённых предприятием j в рамках текущего заказа.

3 Производственная схема

Задачи в работах [1] и [5] используют понятие технологического графа, описывающего составные части при производстве изделий. Всю структуру производства наглядно можно представить в виде направленного ациклического графа. Узлы этого графа соответствуют изделиям $n \in M^0$, так же их и обозначим. На верхнем, корневом уровне графа узлы, соответствующие изделиям, которые названы товарами. Обозначим множество этих узлов, как и множество товаров, т.е. $\widehat{P}^1$.

Дуги, входящие в узел $n \in M^0$ технологического графа, направлены из узлов, соответствующим тем деталям, из которых монтируется изделие n.

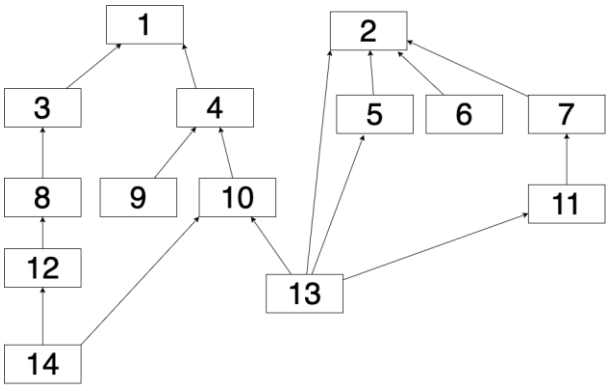

Рис. 1. Пример производственного графа

4 Процедура аукционов

Даже в случае линейности задача поиска оптимального плана может быть очень громоздкой. Для наполнения её данными потребуется постоянный мониторинг ресурсов участников, а также, меняющихся технологических условий производства. Для достаточно большой системы такой централизованный подход может оказаться слишком затратным, если вообще осуществимым. Поэтому возникает желание декомпозировать задачу, передав большую часть функций по оценке производственных мощностей предприятий и разработке общего плана производства самим предприятиям. Аукционная схема – это такой способ декомпозиции общей задачи. Для неё должны выполняться те же балансовые и ресурсные ограничения, и ещё некоторые дополнительные, обусловленные декомпозицией.

5 Программные блоки

Исследование подобных классов задач можно провести при помощи имитационных экспериментов. Для проведения имитационного эксперимента необходимо описать основные блоки, представить их в схемах и описать процедуру расчёта.

5.1 Блок продукт

Под продуктом будем понимать изделие, законченный результат некоторой деятельности. Каждому изделию сопоставляется индекс id. Каждое изделие может состоять из деталей (которые также являются изделиями) или же нет. Назовем описание составных элементов изделия производственной схемой. Для её определения необходимо указать детали исходного изделия и какое количество каждой требуется. Это соответствие легко представить в виде структуры: Map<Product, Int>, где каждой детали сопоставлено их количество, необходимое для сборки одного изделия. Тогда получим описание продукта в коде:

```
1.  class Product(
2.      id: ProductId,// индекс изделия
3.      technicalProductMap: Map<Product, Int> // производственная схема
4.  )
```

5.2 Блок предприятие

Производством изделий занимается предприятие (Company). Пусть каждое имеет свой индекс companyId. Кроме того, за каждым закреплен набор изделий, производством которых оно занимается: обозначим это как products. Для производства предприятие использует собственные ресурсы (resourses), а также комплектующие, которые оно заказывает у других предприятий, входящих в ассоциацию, или покупает на стороне. Для оценки внутренней стоимости изготовления (стоимость изготовления без учета затрат на заказ и доставку недостающих деталей) требуется определить производственные затраты ресурсов для изготовления каждого изделия. Это можно сделать структурой Map<ProductId, List<Resource>> и обозначить как innerProductionsCostMap.

Осуществление поставок между участниками требует затрат и может осуществляться как силами производителей, так и специализированными транспортными компаниями. В работе [1] для моделирования транспорта используется следующее решение: каждому предприятию l приписывается набор таблиц, $T_l = \left(t_{l,r}^n\right), n \in M^0$, в которых указаны цены доставки единицы продукции n от предприятия l другим предприятиям, т.е. r. Для расчета стоимости транспортировки определим структуру Map<ProductId, Map<CompanyId, Double>> и обозначим её как transportCosts. В коде структуру компании можно представить в виде следующей структуры:

```
1.  class Company(
2.      companyId: CompanyId,
3.      products: List<Product>, // продукты, которые изготавливает предприятие
4.      resources: List<Resource>, //  - ресурсы
5.      innerProductionsCostMap: Map<ProductId, List<Resource>>,//производственные схемы
    предприятия
6.      transportCosts: Map<ProductId, Map<CompanyId, Double>> // матрица стоимости доставки
7.  )
```

5.3 Блок Центр

Помимо предприятий, составляющих множество участников системы есть ещё один выделенный участник. Это Центр, который координирует работу участников и производит итоговое (по результатам аукционов) распределение заказа между изготовителями изделий верхнего уровня, т.е. заказанных товаров. Центр также должен контролировать исполнение заказов, расшивать нестыковки, которые могут возникать по причине невыполнения участниками взятых на себя обязательств в плановые сроки, распределять выручку между участниками и т.п. Для выполнения своих функций Центру необходимо хранить список всех участников (companies) и иметь возможность принять начальный заказ – метод produce будет точкой приема заказа.

```
1.  class Center(
2.      companies: List<Company>,
3.  ){
4.      fun produce(product:Product, amount: Int)
5.   }
```

5.3 Расчет стоимости производства

Предположим, Центр принимает заказ на производство изделия (product) в объёме (amount). Производство распределяется между теми участниками, которые специализируются на сборке этого изделия. После выполнения заказа готовые изделия поступают в Центр в виде поставок:

```
1.  fun takeDelivery(comanyId:CompanyId, product:Product, amount:Int )
```

Матрицу поставок участника с companyId представим в виде: supplyMap = Map<Product, Int> . Объединение всех поставок участнику обозначим как allSupplies = Map<Company, Map<Product,Int>>. Добавим метод для нахождения всех необходимых частей для производства продукта product в объеме amount.

```
1.  fun evaluateRequiredProductParts(product:Product, amount: Int): Map<Product, Int> {
2.      val parts: Map<Product, Int>
3.      product.technicalProductMap.assembleParts.forEach { (product, partCount) ->
4.          parts[product] = parts[product] + partCount * amount
5.      }
6.      return parts
7.  }
```

Помним, что должны выполняться балансовые соотношения: объем поставленных изделий должен соответствовать объему необходимых для исполнения заказа на производство продуктов в соответствующем объеме order = Map<Product, Int>:

sumRequiredParts == sumSuppliedParts,

где

```
1.  val sumRequiredParts = order.map{ (product, amount) ->evaluateRequiredProductParts(product,
    amount)}.sumBy( product) // суммарный объем необходимых деталей
2.
3.  val sumSuppliedParts = allSupplies.map { (companyId, supplyMap) -> supplyMap.map{ (product,
    amount)}}.sumBy(product)// суммарный объем поставленных деталей
```

Кроме балансовых должны выполняться ресурсные ограничения. Приведем метод для подсчета затраченных ресурсов на изготовление изделий из поставки supplyMap:

```
1.  fun evaluateResourcesOfSupplyOrder(supplyMap:Map<Product,Int>):Map<Resource,Int> {
2.      val totalSupplyResources: Map<Resource, Int>
3.
4.      supplyMap.forEach{ (product, productAmount) ->
5.          innerProductionsCostMap[product].forEach { (resource, recourceAmount) ->
6.              totalSupplyResources[resource]+=recourceAmount*productAmount
7.          }
8.      }
```

```
9.      return totalSupplyResources
10. }
```

Тогда для проверки выполнения ресурсного ограничения добавим метод:

```
1.  fun compareResources(totalSupplyResources:Map<Resource, Int>, companyResources:Map<Resource,
    Int>): Boolean {
2.
3.      totalSupplyResources.map { (supplyResource, supplyCount) ->
4.
5.          if (companyResources[supplyResource] < supplyCount) {
6.              return false
7.          }
8.      }
9.      return true
10. }
```

Расчет затрат на транспортировку всех поставок allSupplies участнику

```
1.  fun calculateTransportCost(allSupplies: Map<CompanyId, Map<Product, Int>>): Int {
2.      var totalCost = 0
3.      allSupplies.forEach { (companyId, companySupply) ->
4.          companySupply.forEach { (product, count) ->
5.              totalCost += transportCosts[companyId][product] * count
6.          }
7.      }
8.      return totalCost
9.  }
```

Согласно договорённости, один из ресурсов – это денежные средства, идущие на производство изделий. Посчитать расходы на производство можно с помощью метода:

```
1.  fun calculateProductionCost(totalSupplyResources: Map<Resource, Int>): Int {
2.      return totalSupplyResources[Resource.Money]
3.  }
```

Тогда найти общую стоимость производства изделия (product) в объеме (amount) можно с помощью метода:

```
1.  fun totalCost(allSupplies : Map<Company, Map<Product,Int>>) : Double{
2.    val totalProductionCost = allSupplies.map{ (company, supplyMap) ->
3.              calculateProductionCost(evaluateResourcesOfSupplyOrder(supplyMap))
4.                }.toSum()
5.    val totatlTransportCost = calculateTransportCost(allSupplies)
6.
7.    return totalProductionCost + totatlTransportCost
8.  }
```

Заключение

Приведенные выше блоки позволяют написать программный комплекс для расчета стоимости исполнения заказа при проведении имитационных экспериментов. В данной работе не рассмотрено в подробностях сетевое взаимодействие участников: распределение заказов при помощи аукционов. Данный вопрос является предметом следующей работы.

Литература

1. *Гасанов И.И.* Организация аукционов в сетевых моделях// Управление развитием крупномасштабных систем MLSD’2021. Труды четырнадцатой международной конференции. М.: ИПУ РАН, 202 (27–29 сентября 2021 года, Москва, Россия), 2021 – С. 414–422.
2. *Razali M., Ab. Rashid M. F. F., Make M.* Mathematical Modelling of Mixed-Model Assembly Line Balancing Problem with Resources Constraints // IOP Conference Series: Materials Science and Engineering. 2016.
3. *Cao Y., Li Y., Liu O., Zhan J.*. An Optimization Model for Assembly Line Balancing Problem with Uncertain Cycle Time // Mathematical Problems in Engineering, 2020. – P. 1–13.

4. *Yılmaz H., Demir Y.* A New Mathematical Model for Assembly Line Worker Assignment and Balancing // Journal of the Institute of Science and Technology. vol.9, 2019.
5. *Becker Ch., Scholl A.* A survey on problems and methods in generalized assembly line balancing // European Journal of Operational Research. 168, 2006 – P. 694–715.
6. *Boysen, N., Schulze P., Scholl A.* Assembly line balancing: What happened in the last fifteen years? // European Journal of Operational Research, Elsevier, vol. 301(3), 2022 – P. 797–814.