САМООРГАНИЗУЮЩИЕСЯ РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ

А.И. Каляев
И.А. Каляев

А.И. Каляев
И.А. Каляев

САМООРГАНИЗУЮЩИЕСЯ РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ

РОССИЙСКАЯ АКАДЕМИЯ НАУК
Южный Федеральный Университет

А.И. Каляев, И.А. Каляев

САМООРГАНИЗУЮЩИЕСЯ РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ

Москва
2023

УДК 004.75
ББК 32.81
К17

Каляев, Анатолий Игоревич
К17 Самоорганизующиеся распределенные системы / А. И. Каляев, И. А. Каляев ; Российская академия наук. – Москва : РАН , 2023. – 156 с.

ISBN 978-5-907645-00-4

УДК 004.75
ББК 32.81

ISBN 978-5-907645-00-4

СОДЕРЖАНИЕ

Введение

В настоящее время все большую актуальность приобретает проблема создания распределенных систем, т. е. систем, состоящих из множества распределенных в пространстве подсистем (ресурсов), каждая из которых в общем случае выполняет некоторый набор индивидуальных функций, и взаимодействующих друг с другом для выполнения общего функционального задания или некоторого множества заданий [1]. Идеология распределенных систем в настоящее время находит все более широкое применение при создании сложных систем различного назначения. Очевидно, что значительно проще строить какую-либо сложную систему на основе некоторого множества относительно простых функционально готовых ресурсов, выполняющих некоторое подмножество операций, чем проектировать ту же систему как единый сложный ресурс, выполняющий одновременно все множество требуемых операций.

К классу распределенных можно отнести большое число различных технических систем. Примерами распределенных систем в технической сфере могут служить, например, облачные вычислительные среды, состоящие из множества распределенных в пространстве вычислительных ресурсов, совместно участвующих в решении некоторого множества вычислительных задач. Еще один пример – это безлюдное производство, включающее в свой состав множество роботизированных обрабатывающих центров, каждый из которых имеет свою функциональную специализацию, и все они совместно используемются для изготовления сложных деталей. К классу распределенных систем можно также отнести такие сложные технические объекты, как атомные и тепловые станции, авиационные и корабельные комплексы, космические станции и т. п. Действительно, все они состоят из множества взаимосвязанных подсистем (ресурсов), каждая из которых имеет свою функциональную специализацию, но при этом все эти подсистемы используются для выполнения общего задания, выполняемого объектом. В настоящее время крайне актуальной является проблема группового взаимодействия роботов при решении общей задачи [2,3]. Очевидно, что группа роботов – это та же распределенная система, в которой каждый

робот (ресурс) может иметь индивидуальную специализацию, но при этом их действия должны быть направлены на достижение общей, групповой цели.

Однако, с другой стороны, при реализации концепции распределенных систем возникает нетривиальная проблема формирования их внутренней пространственной, временной и функциональной структуры системы для оптимального решения поступающих заданий. Если система предназначена для решения лишь одной задачи или некоторого множества заранее известных задач, то ее внутренняя структура для их оптимального решения может быть спроектирована заранее, на этапе создания системы. Однако если перечень заданий, выполняемых системой, заранее неизвестен, либо они могут поступать в заранее неизвестные моменты времени, «пересекаясь» друг с другом, то возникает сложная проблема автоматического формирования внутренней пространственной, временной и функциональной структуры системы для оптимального решения потока заранее неизвестных задач, поступающих в произвольные моменты времени. Такое автоматическое формирование внутренней пространственной, временной и функциональной структуры системы называется самоорганизацией системы [4].

Самоорганизация распределенной системы должна осуществляться с помощью некоторой специальной подсистемы, входящей в ее состав, функции которой заключаются в приеме поступающих заданий и формировании внутренней структуры системы для их оптимального выполнения. При этом под формированием внутренней структурой системы для выполнения некоторого задания следует понимать выделение некоторого сообщества ресурсов, входящих в состав системы, распределение подзадач поступившего задания между ними и формирование временного графика их взаимодействия при их решении для оптимального выполнения всего задания в целом. Такое устройство будем называть диспетчером, а выполняемую им функцию – самоорганизацией системы, а распределенную систему, обладающую таким диспетчером, – самоорганизующейся распределенной системой (СРС).

Можно рассмотреть несколько подходов к организации диспетчера СРС [5]. В простейшем случае диспетчирование ресурсов СРС может осуществляться из единого центра, т. е. с помощью центрального диспетчера. К преимуществам такого подхода следует отнести как

простоту организации диспетчера, так и возможности использования классических методов распределения ресурсов [6]. Однако, с другой стороны, при этом резко повышаются требования к быстродействию ЦД РС, что, в свою очередь, ведет к росту его стоимости и габаритов, поскольку он решать крайне сложную задачу распределения ресурсов СРС по заданиям в реальном времени их поступления, причем в случае большого количества различных ресурсов в составе СРС и поступающих заданий сложность такой задачи возрастает экспоненциально [7]. Кроме того, централизованная организация диспетчера СРС приводит к снижению отказоустойчивости системы, поскольку выход его из строя приводит к катастрофическим последствиям для всей СРС в целом.

Частично указанные недостатки можно устранить путем иерархической организации диспетчера СРС [1]. При этом в состав СРС должны входить один главный (центральный) диспетчер, отвечающий за общую организацию работы всей СРС в целом, а также ряд подчиненных ему локальных диспетчеров нижнего уровня, отвечающих за координацию работы некоторого подмножества ресурсов РС. Преимуществом такой иерархической организации диспетчера РС является снижение размерности задачи диспетчирования ресурсов, возлагаемых на локальные диспетчеры, поскольку каждый из них отвечает за распределение относительно небольшого количества ресурсов РС по поступающим заданиям.

Однако, с другой стороны, при таком подходе существенно усложняется работа главного диспетчера, на который возлагается достаточно сложная задача разбиения общего задания на ряд подзаданий, которые должны выполняться отдельными локальными диспетчерами, а также организации взаимодействия локальных диспетчеров друг с другом в процессе выполнения общего задания. Кроме того, в системе все равно сохраняется «узкое место» в виде главного диспетчера, выход которого из строя опять-таки приводит к отказу всей СРС в целом [8].

Развитие компьютерных технологий и, прежде всего, сетевых технологий позволяет в настоящее время перейти к созданию диспетчеров СРС принципиально нового типа – мультиагентных диспетчеров с децентрализованной, сетевой организацией [9]. При этом предполагается, что каждый ресурс, входящий в состав СРС, должны обладать неким агентом, представляющим «его интересы» при реализации процедуры

самоорганизации системы. Для этого агенты, представляющие различные ресурсы СРС, должны быть объединены с помощью некоторого информационного канала связи, посредством которого они могут взаимодействовать с целью создания сообществ ресурсов для выполнения различных поступающих заданий и распределения их подзадач между ними. Такая децентрализованная мультиагентная организация диспетчера СРС обладает рядом потенциальных преимуществ.

Во-первых, каждый из агентов будет иметь полную и достоверную информацию о функциональных возможностях и текущей загруженности «своего» ресурса, что позволяет обеспечить возможности оптимального распределения ресурсов СРС между поступающими заданиями.

Во-вторых, каждый агент отвечает только за загрузку «своего» ресурса, что существенно снижает алгоритмическую сложность решаемой им задачи и, как следствие, снижает требования к его аппаратно-программной реализации.

В-третьих, в отличие от централизованной и иерархической организации диспетчера СРС в данном случае в системе отсутствует какой-либо узел, выход которого из строя приводит к катастрофическому отказу всей СРС, т. е. система потенциально будет обладать высокой надежностью и отказоустойчивостью.

В-четвертых, существенно упрощается масштабирование ресурсов СРС, для чего необходимо лишь оснастить новый ресурс соответствующим агентом и подключить его к информационному каналу связи.

Настоящая монография посвящена разработке принципов построения, а также методов и алгоритмов функционирования самоорганизующихся распределенных систем с мультиагентным диспетчером.

1. Самоорганизующиеся распределенные системы с мультиагентным диспетчером

1.1. Формализация задачи самоорганизации распределенной системы

Согласно общепринятому определению, самоорганизация – это свойство системы без дополнительного внешнего воздействия формировать свою внутреннюю пространственную, временную и функциональную структуру для оптимального выполнения поставленного перед ней задания [5,10]. Из этого определения следует, что, во-первых, система должна включать в свой состав некоторый набор подсистем (ресурсов), из которых должна формироваться ее внутренняя структура, а во-вторых, должен быть определен некоторый критерий оптимальности для формирования данной структуры. С другой стороны, система, включающая в свой состав некоторое множество подсистем (ресурсов), имеющих различное функциональное назначение, относится к классу распределенных систем [6]. Поэтому объектом дальнейших исследований будут самоорганизующиеся распределенные системы (СРС).

Задачу самоорганизации распределенной системы формально можно представить в следующем виде.

Пусть существует некоторая распределенная система $\boldsymbol{R}$ (рис.1.1), включающая в свой состав N различных ресурсов $R_1, R_2, ... R_N$, которые могут взаимодействовать друг с другом как физически, так и информационно. Каждый из ресурсов $R_i \in \boldsymbol{R}$ может выполнять некоторое множество операций $\boldsymbol{O}_i = < O_1^i, O_2^i, \dots, O_B^i > (i = 1,2,...,N)$, а множество операций $\boldsymbol{O}$, выполняемых всей системой $\boldsymbol{R}$, в целом определяется как $\boldsymbol{O} = \bigcup_{i=1}^{N} \boldsymbol{O}_i$. При этом будем считать, что операция O_j имеет некоторую трудоемкость $v(O_j)$, которая определяется числом элементарных действий, необходимых для ее выполнения, а ресурс R_i при выполнении операции O_j имеет производительность $D_i(O_j)$, определяемую количеством элементарных действий, выполняемым данным ресурсом в единицу времени. Очевидно, что

зная трудоемкость $v(O_j)$ операции O_j и производительность $D_i(O_j)$ ресурса R_i при ее выполнении, можно рассчитать время, затрачиваемое ресурсом R_i на выполнение операции O_j как $t_i(O_j)=\frac{v(O_j)}{D_i(O_j)}$.

В зависимости от конкретных условий организации различные распределенные системы можно разбить на четыре класса.

1. К первому классу относятся распределенные системы, у которых все ресурсы $R_1, R_2, ..., R_N$ выполняют одинаковые множества операций, т. е. $\boldsymbol{O}_i = \boldsymbol{O}_j = \boldsymbol{O}\ (i=1,2,...,N, j=1,2,...,N)$, причем производительность всех ресурсов $R_i\ (i=1,2,...,N)$ при выполнении одной и той же операции $O_s \in \boldsymbol{O}$ также одинакова, т.е. $D_i(O_s)=D_j(O_s)\ (i=1,2,...,N, j=1,2,...,N)$. Системы такого класса будем называть гомогенными распределенными системами первого типа.

2. Ко второму классу относятся распределенные системы, ресурсы $R_1, R_2, ..., R_N$ которых выполняют одинаковые множества операций, т. е. $\boldsymbol{O}_i = \boldsymbol{O}_j = \boldsymbol{O}\ (i=1,2,...,N, j=1,2,...,N)$, но при этом производительность различных ресурсов $R_i \in \boldsymbol{R}$ и $R_j \in \boldsymbol{R}$ $(i \neq j)$ при выполнении одной и той же операции $O_s \in \boldsymbol{O}$ различна, т. е. $D_i(O_s) \neq D_j(O_s)$ $(i=1,2,...,N, j=1,2,...,i-1,i+1,...,N)$. Системы данного класса будем называть гомогенными распределенными системами второго типа.

3. К третьему классу будем относить распределенные системы, у которых множества операций, выполняемых различными ресурсами, различны (т. е. каждый ресурс имеет свою функциональную специализацию) $\boldsymbol{O}_i \neq \boldsymbol{O}_j\ (i=1,2,...,N, j=1,2,...,i-1,i+1,...,N)$, хотя они могут и пересекаться, т. е. $\boldsymbol{O}_i \cap \boldsymbol{O}_j \neq \varnothing$. При этом производительность различных ресурсов $R_i \in \boldsymbol{R}$ и $R_j \in \boldsymbol{R}$ при выполнении одной и той же операции $O_s \in \boldsymbol{O}$ одинаковая, т. е. $D_i(O_s)=D_j(O_s)$ $(i=1,2,...,N, j=1,2,...,N)$. Системы данного класса будем называть гетерогенными распределенными системами первого типа.

4. Наконец, к четвертому классу относятся распределенные системы, все ресурсы которых $R_1, R_2, ..., R_N$ выполняют различные множества операций, т. е. $\boldsymbol{O}_i \neq \boldsymbol{O}_j$ и $\boldsymbol{O}_i \cap \boldsymbol{O}_j \neq \varnothing$ $(i=1,2,...,N, j=1,2,...,i-1,i+1,...,N)$,

причем производительность различных ресурсов $R_i \in \boldsymbol{R}$ и $R_j \in \boldsymbol{R}$ $(i \neq j)$ при выполнении одной и той же операции $O_s \in \boldsymbol{O}_i$ также различна, т. е. $D_i(O_s) \neq D_j(O_s)$. Системы данного класса будем называть гетерогенными распределенными системами второго типа.

Будем считать, что распределенная система $\boldsymbol{R}$ предназначена для выполнения некоторого множества функциональных заданий $\boldsymbol{Z} =< Z_1, Z_2, ..., Z_L >$, которые могут поступать в произвольные моменты времени. В свою очередь для каждого задания $Z_l \in \boldsymbol{Z}$ должен быть установлен некоторый критерий качества его выполнения Y_l**.** Тогда задача самоорганизации распределенной системы $\boldsymbol{R}$ при выполнении задания Z_l сводится к формированию в $\boldsymbol{R}$ такого подмножества (сообщества) взаимодействующих ресурсов $\boldsymbol{R}_l \subseteq \boldsymbol{R}$, с помощью которого задание Z_l может быть выполнено с заданным уровнем критерия качества Y_l.

В общем случае в распределенной системе $\boldsymbol{R}$ может одновременно выполняться сразу несколько различных заданий $Z_l, \ldots, Z_f$, из множества $\boldsymbol{Z} =< Z_1, Z_2, ..., Z_L >$. Тогда в ее структуре должно быть сформировано несколько подмножеств (сообществ) ресурсов $\boldsymbol{R_l}, \ldots, \boldsymbol{R_f}$, обеспечивающих выполнение данных заданий с заданным уровнем качества. При этом кроме критериев оценки качества выполнения отдельных заданий должен быть введен дополнительный критерий Y^c, определяющий качество выполнения всего множества (потока) заданий, поступающих на систему, который в общем случае должен зависеть от критериев качества отдельных выполняемых заданий, т. е.

$$Y^c = F(Y_l, \ldots, Y_f).$$

1.2. Способы построения диспетчера самоорганизующихся распределенных систем

Очевидно, что для формирования в самоорганизующейся распределенной системе (СРС) сообществ ресурсов $\boldsymbol{R_l}, \ldots, \boldsymbol{R_f}$, для решения поступающих заданий система должна обладать некоторой специальной подсистемой, основные функции которой заключаются в приеме поступающих заданий и формировании сообществ ресурсов, необходимых для их выполнения с заданным уровнем качества. В дальнейшем такую подсистему будем называть диспетчером СРС.

При этом необходимо обратить внимание на следующее обстоятельство. Допустим, что в момент времени $T_{\text{п}}^{l}$ на СРС поступает задание $Z_l \in \boldsymbol{Z}$. Период времени t^l, необходимый для выполнения задания Z_l , будет складываться из двух составляющих

$$t^l = t_p^l + t_B^l,$$

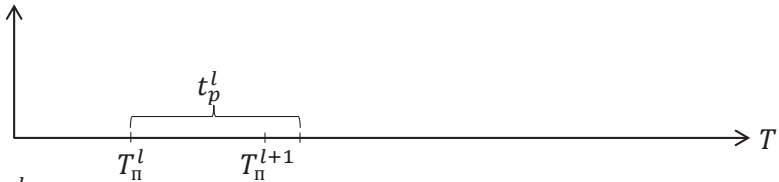
где t_p^l – период времени, затрачиваемый диспетчером на создание сообщества ресурсов $\boldsymbol{R}_l$ для выполнения задания Z_l;

t_B^l – период времени, затрачиваемый ресурсами СРС, входящими в сообщество $\boldsymbol{R}_l$, непосредственно на их совместное выполнение *задания* Z_l .

Если время t_p^l создания сообщества $\boldsymbol{R}_l$ будет велико, то может оказаться, что к моменту $T_{\text{п}}^{l+1}$ поступления следующего задания $Z_{l+1} \in \boldsymbol{Z}$ диспетчер еще не успеет создать сообщество $\boldsymbol{R}_l$ для решения предыдущего задания Z_l (рис. 1.1). В этом случае все действия диспетчера по созданию сообщества для выполнения задания Z_l становятся уже неактуальными и процесс самоорганизации нужно будет начинать заново, уже с учетом сразу двух поступивших заданий Z_l и Z_{l+1}.

Отсюда следует вывод: для того, чтобы СРС обеспечивала выполнение потока заданий в реальном времени их поступления, диспетчер должен успевать создавать сообщество $\boldsymbol{R}_l$ для выполнения очередного поступающего задания Z_l до момента времени $T_{\text{п}}^{l+1}$ поступления следующего задания Z_{l+1} (см. рис. 1.1), т. е. должно выполняться условие

$$t_p^l < T_{\text{п}}^{l+1} - T_{\text{п}}^{l} \qquad (1.1)$$



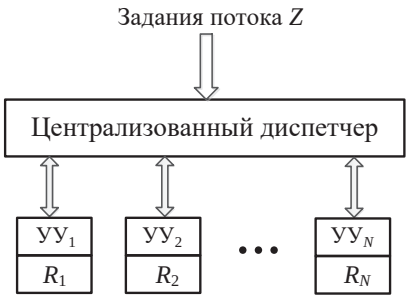
$T_{\text{п}}^l$ – момент времени поступления заданияZ_l;

$T_{\text{п}}^{l+1}$ – момент времени поступления задания Z_{l+1}.

Рис. 1.1. Поступление очередного задания Z_{l+1} до момента завершения процедуры создания сообщества ресурсов $\boldsymbol{R}_l$ для выполнения предыдущего задания Z_l

Исходя из этих соображений рассмотрим различные подходы к построению диспетчера самоорганизующейся распределенной системы $\boldsymbol{R}$, включающей в свой состав N ресурсов $R_1, R_2, ..., R_N$. При этом будем считать, что каждый ресурс $R_i \in \boldsymbol{R}$ обладает своим локальным устройством управления УУ$_i$, с помощью которого он может выполнять операции множества $\boldsymbol{O}_i =< O_1^i, O_2^i, \ldots, O_B^i >$.

В простейшем случае диспетчер СРС может быть реализован в виде некоторого централизованного узла, функции которого заключаются в приеме поступающих заданий, формировании сообществ ресурсов, необходимых для их оптимального выполнения, и распределении операций этих заданий между ресурсами, входящими в данные сообщества (рис. 1.2). После того, как операции очередного поступившего задания Z_l распределены между ресурсами, входящими в состав соответствующего сообщества $\boldsymbol{R}_l$, данные ресурсы переходят к их выполнению с помощью своих локальных устройств управления.



УУ$_i$ (i = 1, 2, …, N) – устройство управления ресурса R_i

Рис. 1.2. СРС с централизованным диспетчером

К преимуществам такого подхода следует отнести простоту организации и алгоритмов функционирования диспетчера СРС. Действительно, такой централизованный диспетчер будет обладать полной информацией о функциональных возможностях и текущей загрузке каждого ресурса, входящего в состав СРС, и на основании этой информации может осуществлять формирование сообществ для оптимального выполнения поступающих заданий. Однако при увеличении числа N ресурсов в составе СРС и числа заданий L сложность задачи самоорганизации, решаемой таким централизованным диспетчером, будет возрастать экспоненциально. Действительно, в общем случае для оптимального распределения L заданий по N ресурсам СРС необходимо будет выполнить полный перебор всех вариантов, количество которых можно оценить с помощью формулы числа размещений с повторениями, т. е. как N^L [11]. Очевидно, что при больших значениях N и L время, затрачиваемое на процедуру самоорганизации может оказаться очень велико, в результате чего условие (1.1) не будет выполняться. Поэтому такой подход к организации диспетчера СРС возможен только в случае относительно небольшого числа ресурсов, входящих в состав СРС, а также достаточно большого временного периода между поступающими заданиями. Кроме того, централизованная организация диспетчера СРС приводит к снижению отказоустойчивости всей системы, поскольку выход, отказ диспетчера приводит к катастрофическим последствиям для всей СРС в целом.

Этих недостатков частично можно избежать при иерархической схеме организации диспетчера (рис. 1.3). При этом диспетчер СРС строится в виде иерархического дерева, в котором каждый из K диспетчеров нижнего уровня отвечает за организацию работы некоторого подмножества ресурсов СРС, а функции диспетчера верхнего уровня заключаются в приеме поступающих заданий и распределении их по диспетчерам нижнего уровня. При этом трудоемкость решения задачи самоорганизации в данном случае можно оценить как

$$\max_{i=1,2,\dots,K}\left(N_i^{L_i}\right),$$

причем

$$\sum_{i=1}^{K} N_i = N \text{ и } \sum_{i=1}^{K} L_i = L,$$

где N_i $(i = 1, 2, \dots, K)$ – число ресурсов, закрепленных за i-м диспетчером нижнего уровня;

L_i $(i = 1, 2, \dots, K)$ – число заданий, назначенных диспетчером верхнего уровня на i-й диспетчер нижнего уровня.

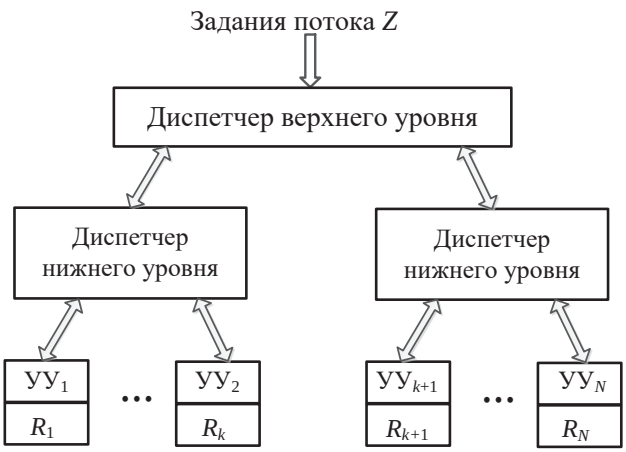


Рис.1.3. СРС с иерархическим диспетчером

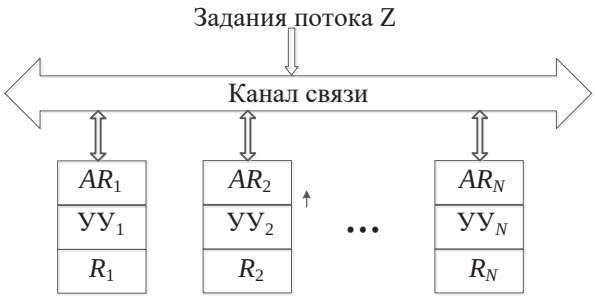
Очевидно, что преимуществами такой организации СРС по сравнению с централизованным диспетчером является, во-первых, снижение трудоемкости задачи самоорганизации, возлагаемой на диспетчеры нижнего уровня, и, как следствие, снижение времени создания сообществ ресурсов для выполнения поступающих заданий, а во-вторых, повышение отказоустойчивости СРС, поскольку выход из строя одного из диспетчеров нижнего уровня не будет приводить к катастрофическим последствиям для всей системы в целом.

Однако, с другой стороны, при этом, во-первых, резко усложняется общая процедура самоорганизации системы, поскольку задания, поступающие на СРС, необходимо каким-то образом распределить по диспетчерами нижнего уровня с учетом специализации и загруженности закрепленных за ними ресурсов, а во-вторых, существенно усложняется проблема масштабирования количества ресурсов при модернизации СРС, поскольку это потребует кардинальной переделки алгоритмов работы как диспетчера верхнего уровня, так и диспетчеров нижнего уровня, обслуживающих эти дополнительные ресурсы. Кроме того, в системе все

равно остается «узкое горло», а именно, диспетчер верхнего уровня, выход которого из строя будет приводить к катастрофическим последствиям для всей СРС в целом.

Поэтому, несмотря на некоторое сокращение времени формирования сообществ для выполнения поступающих заданий и соответствующее снижение требований к временному периоду между поступающими заданиями, данный подход также является малоэффективным.

Указанных выше недостатков позволяет избежать децентрализованная схема организации диспетчера СРС (рис. 1.4).



AR_i $(i = 1,2,...,N)$ - агент ресурса R_i

УУ$_i$ (i = 1, 2, …, N) - устройство управления ресурса R_i

Рис. 1.4. Децентрализованная организация диспетчера СРС

При этом предполагается, что все ресурсы $R_1, R_2, ...R_N$, входящие в состав СРС, должны обладать некоторыми специальными средствами – агентами [12] AR_i $(i = 1, 2, \dots, N)$, которые посредством некоторого канала связи могут обмениваться информацией друг другом и «договариваться» о создании сообществ для выполнения поступающих заданий (рис. 1.4).

Такая децентрализованная организация диспетчера СС имеет целый ряд преимуществ.

Во-первых, трудоемкость процедуры самоорганизации будет существенно ниже, чем в предыдущих случаях, и может быть оценена как $N \cdot L$, где N – число ресурсов СРС, L – число одновременно выполняемых заданий, поскольку каждый агент AR_i $(i = 1, 2, \dots, N)$ самостоятельно принимает решение, в какое из L сообществ ему входить. Это, в свою очередь, позволяет существенно снизить требования к временному интервалу между поступающими заданиями.

Во-вторых, в процессе реализации процедуры самоорганизации агент AR_i ресурса R_i будет иметь полную и достоверную информацию о специализации, производительности при выполнении тех или иных операций и текущей загруженности «своего» ресурса, что обеспечивает возможности создания сообществ ресурсов для выполнения поступающих заданий с высоким уровнем качества.

В-третьих, система будет обладать максимальной отказоустойчивостью, поскольку в ней практически отсутствуют части, выход из строя которых приводит к полной потере работоспособности СРС в целом.

В-четвертых, это простота масштабирования (наращивания ресурсов) СРС. Для этого достаточно просто подключить новый ресурс, обладающий соответствующим агентом, к общему каналу информационного обмена.

И, наконец, в-пятых, это снижение сложности проектирования и создания СРС. Действительно, проектировщику достаточно иметь набор (множество) готовых ресурсов, выполняющих различные операции, оснастить эти ресурсы соответствующими агентами и информационно соединить их друг с другом с помощью канала связи. При этом нет необходимости в разработке сложных программно-аппаратных комплексов, выполняющих функции диспетчеров всей СРС в целом, поскольку эти функции будут выполняться путем скоординированной работы программных агентов отдельных ресурсов СРС. Все это позволяет существенно снизить трудоемкость и, соответственно, стоимость проектирования и создания СРС.

К недостаткам данного подхода следует, по-видимому, отнести возможное снижение качества выполнения заданий по сравнению с централизованным способом организации диспетчера, поскольку создание сообществ для выполнения поступающих заданий осуществляется децентрализовано, что не гарантирует оптимальности решения задачи самоорганизации.

Однако, несмотря на очевидные преимущества, теоретические и практические основы создания СРС с децентрализованным мультиагентным диспетчером слабо развиты. Проблема создания таких систем требует разработки новых методов мультиагентного диспетчирования ресурсов СРС при выполнении потоков заданий, а

также методов и алгоритмов работы программных агентов, обеспечивающих решения задачи самоорганизации системы для выполнения поступающих заданий с допустимым уровнем качества.

1.3 Базовые принципы организации и функционирования самоорганизующихся распределенных систем

Итак, в качестве объекта дальнейшего исследования будем рассматривать самоорганизующуюся распределенную систему $\boldsymbol{R}$, включающую в свой состав N ресурсов $R_1, R_2, ... R_N$, каждый из которых обладает своим локальным устройством управления УУ$_i$, обеспечивающим выполнение операций множества $\boldsymbol{O}_i = < O_1^i, O_2^i, ... O_B^i >$, а также некоторым агентом AR_i, представляющим его «интересы» при реализации процедуры самоорганизации и соединенным с агентами других ресурсов посредством некоторого информационного канала связи (см. рис. 1.4). Кроме того, будем считать, что в произвольные моменты времени на СРС поступает поток функциональных заданий $\boldsymbol{Z} = < Z_1, Z_2, ..., Z_L >$, причем каждое отдельное задание $Z_l \in \boldsymbol{Z}$ представляется в виде некоторого ациклического графа операций $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ (рис. 1.5), вершине $q_j \in \boldsymbol{Q}_l$ которого приписана некоторая операция O_j, принадлежащая множеству $\boldsymbol{O} = \bigcup_{i=1}^{N} \boldsymbol{O}_i$, а дуга $x(q_j, q_{j+1})$ устанавливает взаимосвязь между операциями, приписанными вершинам q_j и q_{j+1}, т. е. определяет, что результат операции O_j, приписанной вершине q_j, необходим для выполнения следующей операции O_{j+1}, приписанной вершине q_{j+1}.

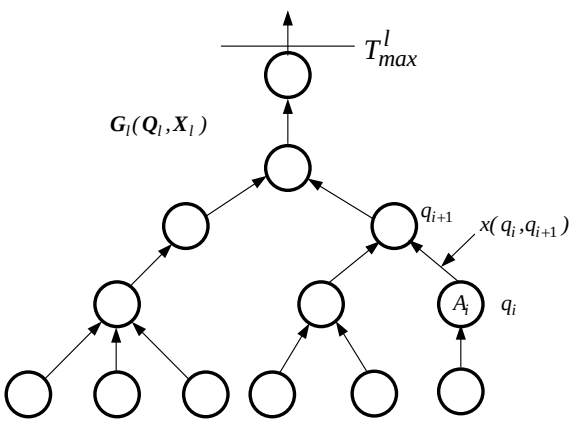


Рис. 1.5. Граф задания Z_l

Под ветвью графа задания Z_l будем понимать некоторую последовательность вершин $\boldsymbol{H}_f =< q_1^f, q_2^f, ..., q_k^f >$ графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, в которой вершины q_j^f и $q_{j+1}^f (j = 1,2,...,k-1)$ соединены дугой $x(q_j^f, q_{j+1}^f)$ (рис. 1.6).

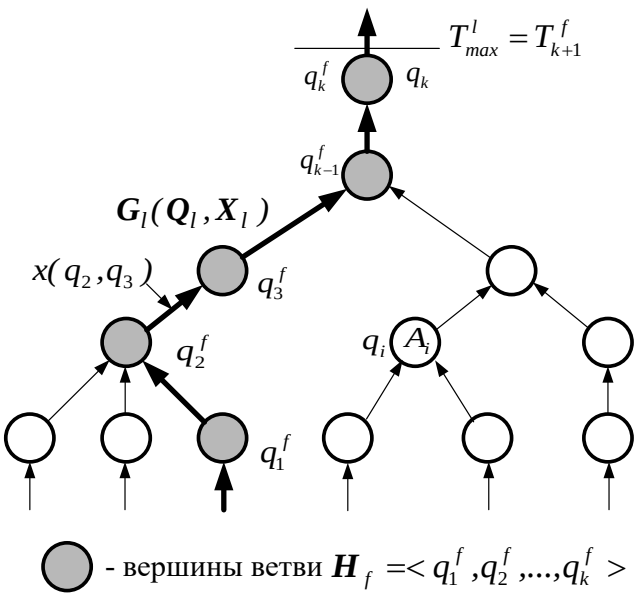


Рис. 1.6. Ветвь $\boldsymbol{H}_f$ графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l

Иными словами, ветвь $\boldsymbol{H}_f$ определяет некоторую последовательность операций задания Z_l, в которой каждая последующая операция использует результат выполнения предыдущей операции. Очевидно, что операции, приписанные вершинам ветви $\boldsymbol{H}_f$, могут выполняться только последовательно. При этом под длиной t_f ветви $\boldsymbol{H}_f$ будем понимать период времени, затрачиваемый на ее выполнение, т. е.

$$t_f = \sum_{i=1}^{k} \left(t_p\left(O_i^f\right) + t_{\text{п}}(O_i^f, O_{i+1}^f) \right),$$

где $t_p\left(O_i^f\right)$ – период времени, затрачиваемый ресурсом $R_p \in \boldsymbol{R}$ на выполнение операции O_i^f, приписанной вершине $q_i^f \in \boldsymbol{H}_f$ $(i = 1,2,...,k)$;

$t_{\text{п}}(O_i^f, O_{i+1}^f)$ – период времени, затрачиваемый на передачу результатов операции O_i^f, выполняемой ресурсом $R_p \in \boldsymbol{R}$, ресурсу $R_c \in \boldsymbol{R}$, выполняющему следующую по очереди операцию O_{i+1}^f ветви $\boldsymbol{H}_f$, причем $t_p(O_i^f) = \frac{v(O_i^f)}{D_p(O_i^f)}$, где $v(O_i^f)$ – трудоемкость операции O_i^f; $D_p(O_i^f)$– производительность ресурса R_p при выполнении операции O_i^f.

Для упрощения дальнейших построений будем считать, что $t_{\text{п}}(O_i^f, O_{i+1}^f) = 0$, если операции O_i^f и O_{i+1}^f выполняет один и тот же ресурс $R_p \in \boldsymbol{R}$, и $t_{\text{п}}(O_i^f, O_{i+1}^f) = t_{\text{П}}$, если операции O_i^f и O_{i+1}^f выполняют различные ресурсы из множества $\boldsymbol{R}$.

Будем считать, что для каждого задания $Z_l \in \boldsymbol{Z}$ установлен момент времени T_{max}^l, к которому это задание должно быть выполнено. Очевидно, что момент времени T_{max}^l должен быть установлен таким образом, чтобы выполнялось условие $T_{max}^l \geq T_{\text{п}}^l + t_{min}^l$, где $T_{\text{п}}^l$ момент времени поступления задания в СРС, t_{min}^l – минимально возможный период времени выполнения задания Z_l.

При этом значение t_{min}^l будет определяться временем выполнения наиболее трудоемкой ветви графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, т. е.

$$t_{min}^l = \sum_{j=1}^{k} t(O_j) = \sum_{j=1}^{k} \frac{v(O_j)}{D_i(O_j)} \geq \sum_{j=1}^{k} \frac{v(O_j)}{D_{max}(O_j)};$$

где k – число операций в наиболее трудоемкой ветви графа-задания Z_l;

$t(O_j)$ – время выполнения j-ой операции наиболее трудоемкой ветви графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$;

$v(O_j)$ – трудоемкость операции O_j;

$D_i(O_j)$ – производительность ресурса R_i, выполняющего операцию O_j;

$D_{max}(O_j)$ – максимальная производительность среди всех ресурсов множества $\boldsymbol{R}$ при выполнении операции O_j.

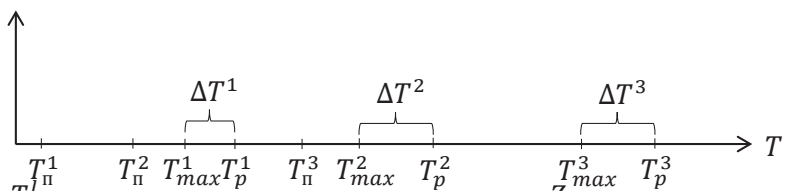
Очевидно, что реальный момент времени T_p^l выполнения задания $Z_l \in \boldsymbol{Z}$ может отличаться от требуемого T_{max}^l, причем если $T_p^l < T_{max}^l$ (т. е. задание выполняется раньше требуемого времени), то это допустимо, а

если $T_p^l > T_{max}^l$ (т. е. задание выполняется позже требуемого момента времени), то это нежелательно. Поэтому в качестве критерия качества при выполнении задания Z_l целесообразно принять минимум величины времени $\Delta T^l = T_p^l - T_{max}^l$, т. е. времени задержки его выполнения относительно требуемого момента времени T_{max}^l (рис. 1.7).

Если же СРС должна выполнять сразу несколько заданий из множества $\boldsymbol{Z} =< Z_1, Z_2, \ldots, Z_L >$, то, как показано выше, в этом случае кроме критерия качества выполнения отдельного задания $Z_l \in \boldsymbol{Z}$ должен быть установлен некоторый критерий качества одновременного выполнения множества (потока) заданий, причем этот критерий должен зависеть от критериев качества выполнения отдельных заданий, входящих в это множество. Таковым критерием можно принять минимум среднего времени задержки выполнения всех заданий потока относительно требуемых моментов времени T_{max}^l, т. е. минимум величины (см. рис. 1.7).

$$\Delta T = \frac{\sum_{l=1}^{L} \Delta T^l}{L} = \frac{\sum_{l=1}^{L} \left(T_p^l - T_{max}^l\right)}{L},$$

где L – число заданий, выполненных СРС.



$T_п^l$ – момент времени поступления задания Z_l;

T_{max}^l – требуемый момент времени выполнения задания Z_l;

T_p^l – реальный момент времени завершения выполнения задания Z_l .

Рис. 1.7. Задержки выполнения заданий потока $\boldsymbol{Z}$

Исходя из приведенных выше соображений, задачу самоорганизации, решаемую диспетчером СРС, можно сформулировать в следующем виде: необходимо обеспечить такое распределение операций $O_j \subseteq \boldsymbol{O}$ заданий $Z_l \in \boldsymbol{Z}$, поступающих в произвольные моменты времени и описываемых графом $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, между ресурсами множества $\boldsymbol{R} =< R_1, R_2, \ldots R_N >$, которое бы минимизировало среднее время ΔT задержки выполнения всех заданий потока $\boldsymbol{Z} =< Z_1, Z_2, \ldots, Z_L >$.

Проблеме распределения операций посвящено достаточно большое число исследований с использованием, например, методов линейного или динамического программирования, теории расписаний и т. д. [13,14,15]. Поэтому, если графы $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)(l = 1,2,...,M)$ всех заданий множества $\boldsymbol{Z}$ и состав ресурсов распределенной системы $\boldsymbol{R}$ известны заранее, то с помощью известных методов, в принципе, такие распределения могут быть получены для каждого из заданий множества $Z_l \in \boldsymbol{Z}$ заранее, до начала их исполнения и просто храниться в памяти системы.

Однако это – идеализированный случай. В реальности графы операций выполняемых СРС заданий $\boldsymbol{Z} =< Z_1, Z_2, ..., Z_L >$ могут быть заранее неизвестны, а состав ресурсов, входящих в СРС, и их характеристики могут динамически изменяться непредсказуемым образом (например, некоторые из них могут выходить из строя либо изменять свою производительность). Кроме того, поскольку мы приняли, что задания на СРС поступают в случайные моменты времени, то невозможно заранее предугадать, какие из заданий могут одновременно выполняться в СРС. Поэтому возникает необходимость разработки такого диспетчера, который бы обеспечивал возможность автоматического распределения операций потока заранее неизвестных заданий $Z_l \in \boldsymbol{Z}$, поступающих в произвольные моменты времени, среди множества ресурсов $R_1, R_2, ...R_N$ самоорганизующейся распределенной системы $\boldsymbol{R}$.

Как мы приняли выше, за каждым ресурсом R_i $(i = 1,2,...,N)$, входящим в состав СРС, закреплен отдельный агент AR_i, причем все агенты, представляющие различные ресурсы СРС, могут общаться друг с другом посредством некоторого информационного канала связи. Будем считать, что все эти агенты заинтересованы в выполнении поставленного перед СРС задания $Z_l \in \boldsymbol{Z}$ к требуемому моменту времени T_{max}^l. Тогда каждый из агентов AR_i $(i = 1,2,...,N)$ в процессе распределения операций задания $Z_l \in \boldsymbol{Z}$ должен постараться взять «на себя» (т. е. на представляемый ими ресурс) исполнение такого подмножества вершин (ветви) графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания $Z_l \in \boldsymbol{Z}$, операции которых он сможет выполнить с помощью «своего» ресурса R_i к установленному моменту времени T_{max}^l. Исходя из этих соображений процедура создания

сообщества $\boldsymbol{R}_l$ по выполнению задания $Z_l \in \boldsymbol{Z}$ может осуществляться путем последовательного выбора всеми агентами ресурсов СРС, не задействованными в выполнении других заданий, максимальных по длине ветвей графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, которые они могут выполнить к установленному моменту времени.

Здесь возникает один вопрос: где должен быть размещен граф $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l и каким образом должен осуществляться доступ к нему агентов различных ресурсов $R_1, R_2, ...R_N$ в процессе самоорганизации системы? Очевидно, что это должно быть какое-то хранилище, на котором могут размещаться поступающие задания и к которому должны иметь доступ агенты всех ресурсов СРС. В качестве такого общедоступного хранилища может быть использован некоторый пассивный узел, подключенный к информационному каналу межагентского обмена и играющий роль «доски объявлений» (ДО) (рис. 1.8). [16] На такой «доске объявлений» с одной стороны, могут размещаться поступающие задания, представленные в виде графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, а с другой стороны – агенты всех ресурсов $R_1, R_2, ...R_N$ могут иметь доступ к размещенным там заданиям для участия в процедуре распределения их операций.

При этом дескриптор задания Z_l, размещаемого на ДО, должен содержать следующую информацию:

– граф $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l, представленный, например, в виде матрицы инцидентности вершин;

– список вершин множества $\boldsymbol{Q}_l$ и приписанных им операций из множества $\boldsymbol{O}$;

– момент времени T_{max}^l, к которому требуется получить результат выполнения задания Z_l;

– список агентов, участвующих в сообществе $\boldsymbol{R}_l$ по выполнению задания Z_l (заметим, что в момент размещения задания на ДО этот список будет пуст);

– время ΔT^l задержки выполнения задания Z_l относительно требуемого момента времени T_{max}^l (момент размещения задания Z_l на ДО $\Delta T^l = 0$).

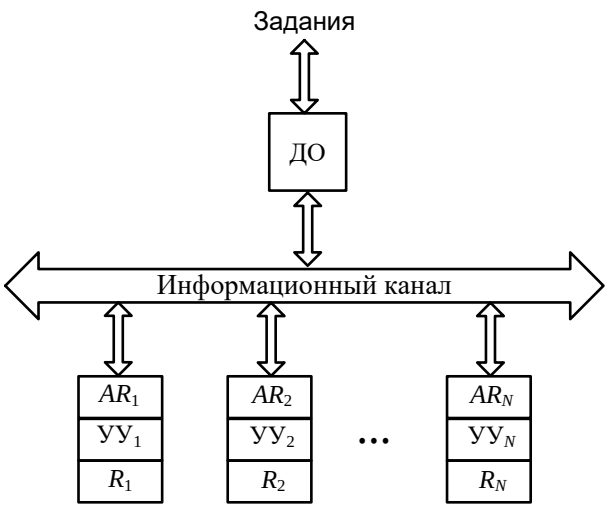


Рис. 1.8. Размещение дескрипторов поступающих заданий на доске объявлений

Учитывая приведенные выше соображения, процедуру работу самоорганизации распределенной системы с мультиагентным диспетчером в общем виде можно представить следующим образом.

1. Дескриптор очередного задания $Z_l \in \boldsymbol{Z}$ размещается на ДО.

2. Агенты ресурсов $R_1, R_2, ... R_N$, входящих в состав СРС, периодически в некоторой последовательности (например, в порядке нумерации) опрашивают ДО в поисках работы для «своих» ресурсов.

3. Агент AR_1 ресурса R_1 при обнаружения на ДО некоторого задания $Z_l \in \boldsymbol{Z}$ делает попытку войти в состав сообщества $\boldsymbol{R}_l$ по его выполнению. Для этого агент AR_1 выделяет в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания $Z_l \in \boldsymbol{Z}$ наиболее длинную ветвь $\boldsymbol{H}_1 = < q_1^1, q_2^1, ..., q_k^1 >$, операции которой он может выполнить с помощью «своего» ресурса R_1 с минимальной задержкой относительно требуемого момента времени T_{k+1}^1, приписанного ее конечной вершине q_k^1. Здесь следует отметить, что при размещении задания Z_l на ДО требуемый момент времени исполнения будет установлен только для конечной вершины q_k всего графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, который определяется требуемым моментом времени T_{max}^l выполнения

всего задания Z_l. Если агент AR_1 обнаруживает ветвь $\boldsymbol{H}_1$ графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, удовлетворяющую данным условиям, то он вступает в сообщество $\boldsymbol{R}_l$ по выполнению задания $Z_l \in \boldsymbol{Z}$.

4. При этом агент AR_1 осуществляет следующие модификации дескриптора задания Z_l:

– выбранная им для исполнения ветвь $\boldsymbol{H}_1$ удаляется из графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, т. е. формируется новый граф $\boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1) = \boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l) / \boldsymbol{H}_1$;

– всем вершинам этого модифицированного графа $\boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1)$, инцидентным вершинам удаленной ветви $\boldsymbol{H}_1$ приписываются моменты времени, когда соответствующие им операции должны быть выполнены с тем, чтобы «вписаться» в график выполнения ветви $\boldsymbol{H}_1$, закрепленной за его ресурсом R_1;

– номер ресурса R_1 записывается в список участников сообщества $\boldsymbol{R}_l$ по выполнению задания Z_l;

– значение времени ΔT^l задержки увеличивается на величину задержки выполнения ветви $\boldsymbol{H}_1$ ресурсом R_1.

5. Далее аналогичный выбор делает агент AR_2, представляющий ресурс R_2 – в графе $\boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1)$, оставшемся на ДО после модификации агентом AR_1, он выбирает такую ветвь $\boldsymbol{H}_2 = < q_1^2, q_2^2, ..., q_r^2 >$, которую с помощью «своего» ресурса R_2 он может выполнить к моменту времени, приписанному ее конечной вершине q_r^2 с минимальной задержкой (в результате модификации, произведенной агентом AR_1, в графе $\boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1)$ будет иметься некоторое множество вершин, которым приписаны требуемые моменты времени их исполнения). После этого агент AR_2 осуществляет модификацию графа задания на ДО:

– из графа $\boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1)$ исключается ветвь $\boldsymbol{H}_2$, в результате чего формируется новый граф $\boldsymbol{G}_l^2(\boldsymbol{Q}_l^2, \boldsymbol{X}_l^2) = \boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1) / \boldsymbol{H}_2$;

– всем вершинам этого модифицированного графа $\boldsymbol{G}_l^2(\boldsymbol{Q}_l^2, \boldsymbol{X}_l^2)$, инцидентным вершинам удаленной ветви $\boldsymbol{H}_2$ приписываются моменты

времени, когда соответствующие им операции должны быть выполнены с тем, чтобы «вписаться» в график выполнения ветви $\boldsymbol{H}_2$;

– номер ресурса R_2 записывается в список участников сообщества $\boldsymbol{R}_l$ по выполнению задания Z_l;

– значение времени ΔT^l задержки увеличивается на величину задержки выполнения ветви $\boldsymbol{H}_2$ ресурсом R_2.

6. Далее аналогичным образом в процесс распределения вступает агент AR_1, представляющий ресурс R_3 и т. д., до тех пор, пока не будут разобраны все ветви графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, т. е. пока не окажется, что после очередной модификации граф задания на ДО становится пустым.

В результате выполнения описанной выше процедуры все ветви графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ будут распределены между агентами сообщества $\boldsymbol{R}_l$, а значение величины ΔT^l на ДО будет определять задержку выполнения задания Z_l относительно требуемого момента времени T^l_{max}.

После этого агенты созданного таким образом сообщества $\boldsymbol{R}_l$ инициируют процедуру выполнения операций, приписанных вершинам закрепленных за ними ветвям графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, с помощью локальных устройств управления «своих» ресурсов.

1.4. Принципы организации бесконфликтного информационного взаимодействия между агентами ресурсов СРС

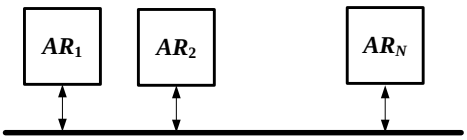
Необходимым условием для реализации описанной выше процедуры самоорганизации системы при выполнении потока заданий $\boldsymbol{Z} =< Z_1, Z_2, ..., Z_L >$ является наличие некоторого канала информационного обмена между агентами ресурсов $R_1, R_2, ... R_N$. При этом очевидно, что от скорости обмена информацией между агентами с помощью данного канала связи во многом будут зависеть и время реализации процедуры самоорганизации, и, как следствие, допустимая частота поступления заданий на СРС. Исходя их этих соображений рассмотрим некоторые способы организации такого межагентского информационного канала связи.

На рис. 1.9 показаны возможные варианты топологии межагентского интерконнекта. Каждый из этих вариантов обладает своими преимуществами и недостатками. Топология «общая шина» (рис.1.9 а) наиболее проста в организации, однако может приводить к возникновению конфликтов между агентами, одновременно пытающимися передать сообщения по одной шине, что в свою очередь может привести к существенным задержкам при передаче сообщений и, как следствие, к увеличению времени процедуры самоорганизации.

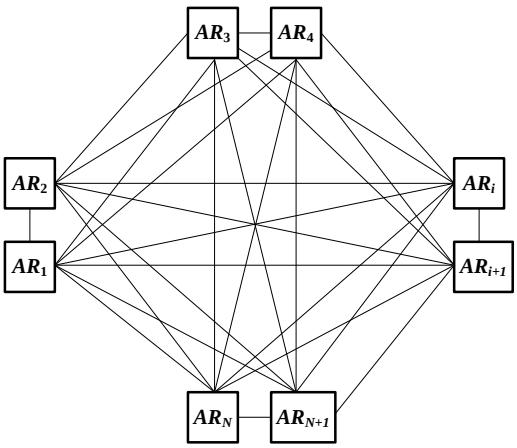
Топология «полный граф» (рис.1.9 б) обеспечивает прямую связь каждого агента со всеми другими, в результате чего обеспечивается минимальное время передачи сообщений между ними. Однако, с другой стороны, организация такой топологии требует больших аппаратурных затрат, поскольку число прямых межагентских связей будет равно N*(N-1)/2, где N – число ресурсов в системе.

В топологиях «решетка» (рис.1.9 с) и «тор» (рис. 1.9 д) между собой напрямую связываются только агенты близлежащих ресурсов. При этом передача сообщений между агентами удаленных ресурсов осуществляется посредством их последовательной передачи по цепочке агентов. Отметим, что длина самой длинной такой цепочки при использовании топологии «решетка» будет равна M/2, где M – число ресурсов, расположенных по периметру решетки, а при использовании топологии «тор» – M/4.

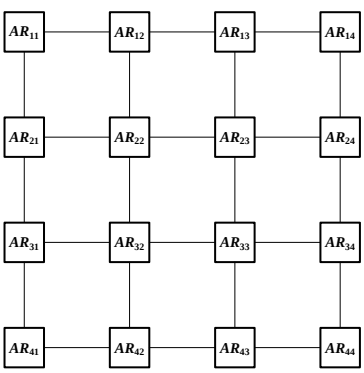
Топология «гиперкуб» (рис. 1.9 е) является некоторым компромиссным вариантом между топологиями «полный граф» и «тор». В топологии «гиперкуб» каждый агент связан лишь с $\log_2 N$ близлежащими агентами, а не с N, как в топологии «полный граф». При этом среднее «информационное расстояние» между агентами не будет превышать величины N.



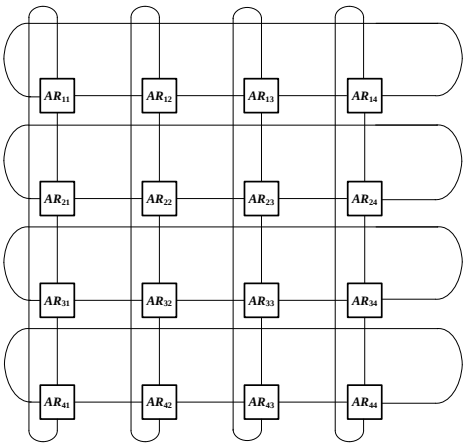
а) топология «общая шина»



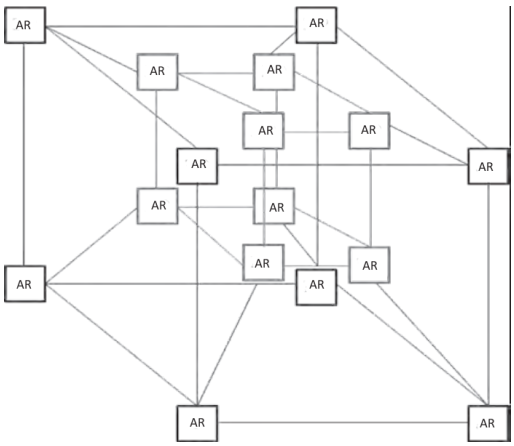
б) топология «полный граф»



с) топология «решетка»



д) топология «тор»



е) топология «гиперкуб»

Рис. 1.9. Варианты топологий межагентского интерконнекта

Выбор той или иной топологии межагентского интерконнекта должен, по-видимому, определяться, во-первых, исходя из требований к времени реализации процедуры самоорганизации системы, которое, в свою очередь, как показано выше, определяется частотой поступления заданий на СРС, а во-вторых, требованиями к аппаратурным затратам на его реализацию.

Как показано выше, к информационному каналу межагентского взаимодействия должна быть подключена и ДО, к которой агенты должны обращаться в процессе реализации процедуры самоорганизации системы. При этом между отдельными агентами могут возникать конфликты, которые могут существенно влиять на эффективность работы СРС в целом. Конфликты могут быть следующих типов.

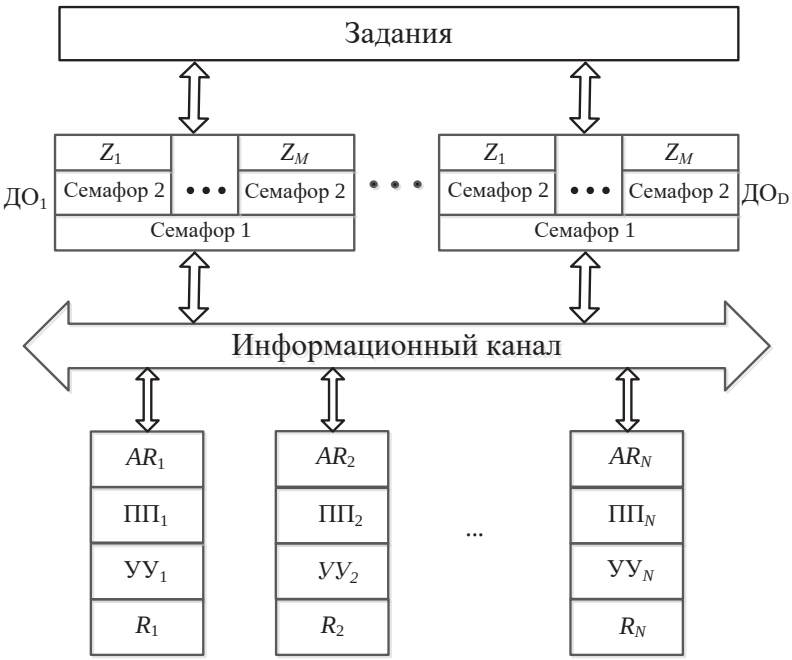
Во-первых, несколько агентов свободных ресурсов могут пытаться одновременно обратиться к ДО в поисках заданий. Если доступ к ДО ограничен (например, в текущий момент времени к ДО может обращаться только один агент), то при этом возникает конфликт между агентами. В простейшем случае данный конфликт может быть разрешен путем установления некоторой очередности обращения агентов к ДО, например в порядке нумерации их ресурсов. Однако такой подход может приводить к длительному ожиданию агентами своей очереди и, как следствие, к увеличению времени реализации процедуры самоорганизации. Частично

такой неэффективный простой ресурсов СС можно сократить путем введения в состав системы нескольких досок объявлений $ДО_1$,…, $ДО_D$, к которым различные агенты могут обращаться одновременно (рис. 1.10). В этом случае агент свободного ресурса R_p может обращаться к той ДО, которая в текущий момент времени не занята работой с агентом другого ресурса R_c. При этом факт занятости $ДО_i$ $(i = 1,2,...,D)$ работой с агентом AR_c может отмечаться установкой некоторого «семафора 1», анализируя который другие агенты будут понимать, что в текущий момент данная ДО занята (рис. 1.10).

Согласно описанным в предыдущем разделе принципам самоорганизации системы, при обнаружении на ДО некоторого задания Z_l агент AR_p должен выполнить ряд манипуляций с его графом, а именно найти наиболее трудоемкую ветвь, которую он может выполнить с помощью своего ресурса к требуемому моменту времени, и далее осуществить соответствующую модификацию графа задания. Очевидно, что данная процедура относительно трудоемка и поэтому может занимать достаточно много времени, в течение которого ДО будет занята и, соответственно, недоступна для работы с другими агентами. Время занятости ДО при работе с агентом AR_p можно сократить путем введения в состав каждого ресурса СРС дополнительной промежуточной памяти (ПП) (рис. 1.10), в которую агент AR_p может считать дескриптор задания Z_l, в выполнении которого он хочет принять участие, после чего он может осуществлять все процедуры и модификации с графом этого задания с помощью данной промежуточной памяти. При таком подходе время занятости ДО при работе с агентом AR_p сокращается до времени считывания дескриптора задания с ДО в промежуточную память $ПП_p$ ресурса R_p и времени записи модифицированного дескриптора из памяти обратно на ДО.

Однако при этом возникает другая проблема. Как только некоторый агент AR_p считал с ДО дескриптор задания Z_l и приступил к его модификации, то никакой другой агент не должен работать с дескриптором данного задания. Иначе может произойти конфликт, когда несколько агентов будут одновременно пытаться принять к исполнению

одни и те же ветви (операции) задания Z_l. Поэтому, как только некоторый агент считал дескриптор задания Z_l с ДО в свою промежуточную память, он должен установить некоторый «семафор 2», запрещающий обращение к этому заданию другими агентами. «Семафор 2», запрещающий работу с заданием, должен сниматься только после возвращения (перезаписи) агентом модифицированного дескриптора задания Z_l обратно на ДО (рис. 1.10).



ПП$_i$ *(i*=1,2,…,*N)*- промежуточная память *i*-го агента AR_i

Рис. 1.10. Схема организации СРС с бесконфликтным доступом агентов на ДО

Учитывая приведенные выше соображения, можно предложить следующие базовые принципы организации бесконфликтных обращений агентов AR_p $(p=1,2,...,N)$ ресурсов СРС к ДО.

1. Агенты периодически опрашивают ДО в порядке своей очереди (нумерации их ресурсов). Если в состав СРС входит несколько ДО, то каждый агент последовательно опрашивает их. Агент может обращаться к тем ДО, у которых отсутствует запрещающий «семафор 1».

2. При обнаружении на ДО некоторого задания $Z_l \in \mathbf{Z}$, не отмеченного запрещающим «семафором 2», агент считывает его дескриптор в свою промежуточную память. При этом на время считывания устанавливается соответствующий «семафор 1», запрещающий обращение к данной ДО другим агентам. После завершения считывания «семафор 1» снимается.

3. После считывания дескриптора задания Z_l в промежуточную память агент устанавливает на ДО «семафор 2», запрещающий работу с данным заданием другим агентам.

4. После завершения манипуляций с дескриптором задания Z_l агент возвращает модифицированный дескриптор на ДО. При этом «семафор 2» с задания Z_l на ДО снимается. На время записи модифицированного дескриптора задания Z_l на ДО устанавливается «семафор 1», запрещающий обращения к ней других агентов, который снимается после завершения процедуры записи.

2. Алгоритмы самоорганизации

2.1. Алгоритм мультиагентного диспетчирования ресурсов в гомогенной самоорганизующейся распределенной системе первого типа

В подразделе 1.1 проведена классификация СРС и показано, что все СРС могут быть разбиты на четыре класса, отличающиеся специализацией отдельных ресурсов СРС, а также их производительностью при выполнении тех или иных операций. Очевидно, что алгоритмы самоорганизации системы также должны зависеть от этих данных параметров, т. е. быть адаптированными под тип самоорганизующейся системы.

Наиболее простым классом самоорганизующихся распределенных систем являются гомогенные системы первого типа, в которых все ресурсы множества $\boldsymbol{R}$ совершенно идентичны, т. е. могут выполнять одинаковый набор операций $\mathbf{O} = < O_1, O_2, ..., O_B >$, причем производительность всех ресурсов R_p ($p = 1,2,...,N$) при выполнении одних и тех же операций также одинакова.

В подразделе 1.3 предложены базовые принципы процедуры самоорганизации, подразумевающие мультиагентное распределение операций некоторого функционального задания Z_l между ресурсами, входящими в состав СРС. При этом агент каждого ресурса старается принять на исполнение наиболее длинную ветвь графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, операции которой он может выполнить с помощью «своего» ресурса с минимальной задержкой относительно приписанного ее конечной вершине требуемого времени исполнения.

Если самоорганизующаяся распределенная система относится к классу гомогенных систем первого типа, все ресурсы множества $\boldsymbol{R}$ имеют одинаковую производительность при выполнении идентичных операций множества $\boldsymbol{O}$ и, следовательно, выполняют их за одинаковый период времени, то, очевидно, выполнение всех операций, приписанных вершинам нити $\boldsymbol{H}_f$, целесообразно осуществлять с помощью одного и того же ресурса $R_p \in \boldsymbol{R}$. Действительно, в этом случае период времени $t_{\text{п}}\left(O_i^f, O_{i+1}^f\right)$, затрачиваемый на передачу результатов операций между ресурсами, исполняющими операции ветви $\boldsymbol{H}_f$, будет равен нулю и, соответственно, длина (период времени выполнения) всей ветви $\boldsymbol{H}_f$ будет минимальной, т. е.

$$t_f = \sum_{i=1}^{k} t_p\left(O_i^f\right) + t_{\text{п}}, \qquad (2.1)$$

где $t_p\left(O_i^f\right)$ – период времени, затрачиваемый ресурсом $R_p \in \boldsymbol{R}$ на выполнение операции O_i^f, приписанной вершине $q_i^f \in \boldsymbol{H}_f$ $(i = 1,2,...,k)$;
$t_{\text{п}}$ – период времени, затрачиваемый на передачу результата исполнения операции, приписанной последней вершине q_k^f ветви $\boldsymbol{H}_f$, ресурсу, исполняющему смежную ветвь графа задания.

При этом, если требуемый момент времени T_{k+1}^f исполнения последней вершины q_k^f ветви $\boldsymbol{H}_f$ известен, то зная период времени $t_p\left(O_i^f\right)$ $(i = 1,2,...,k)$ выполнения ресурсом R_p отдельных операций, приписанных вершинам ветви $\boldsymbol{H}_f$, можно, соответственно, определить требуемый момент времени начала выполнения операции O_d^f, приписанной любой вершине $q_d^f \in \boldsymbol{H}_f$ $(d = 1,2,...,k)$ как (рис 2.1):

$$T_d^f = T_{k+1}^f - \left(\sum_{j=d}^{k} t_p(O_j^f) + t_{\text{п}}\right), \tag{2.2}$$

где $t_p\left(O_j^f\right) = \frac{v\left(o_j^f\right)}{D_p\left(o_j^f\right)}$

.

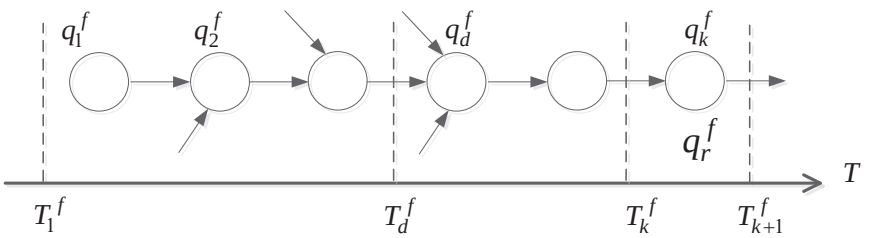


Рис. 2.1. Моменты времени начала выполнения операций, приписанных вершинам ветви $\boldsymbol{H}_f$

Отметим, что при размещении задания Z_l на ДО требуемый момент времени исполнения приписан только конечной вершине $q_k \in \boldsymbol{Q}_l$ графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, причем этот момент времени определяется требуемым моментом времени T_{max}^l выполнения всего задания Z_l.

Исходя из этих соображений, можно предложить следующую процедуру мультиагентного диспетчирования ресурсов в гомогенной СРС первого типа при выполнении потока заданий $\boldsymbol{Z}$.

Дескриптор графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l размещается на ДО, причем в дальнейшем будем считать, что в момент размещения задания на ДО $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l) = \boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1)$. При этом на ДО может одновременно находиться сразу несколько заданий из множества $\boldsymbol{Z}$.

Агенты ресурсов СРС в некотором порядке (например, в порядке нумерации) опрашивают ДО в поисках заданий для «своих» ресурсов. Если агент AR_p ресурса $R_p \in \boldsymbol{R}$ обнаруживает на ДО дескриптор задания Z_l, то он оценивает возможность своего участия в сообществе $\boldsymbol{R}_l$ по его выполнению. Для этого агент AR_p выделяет в графе $\boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1)$ задания Z_l с помощью выражения (2.1) наиболее длинную ветвь $\boldsymbol{H}_1 =< q_1^1, q_2^1, ..., q_k^1 >$, конечной вершине которой приписан момент времени ее исполнения T_{k+1}^f (рис. 2.2).

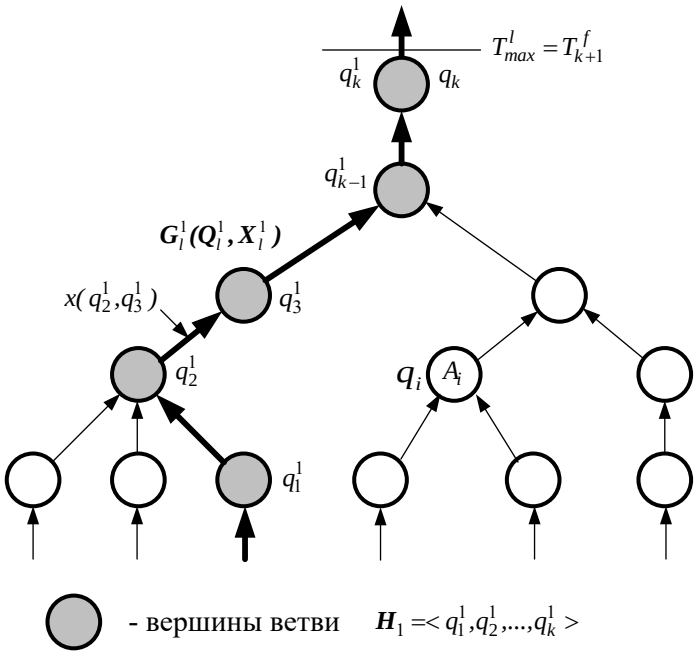


Рис. 2.2. Выделение ветви $\boldsymbol{H}_1$ на графе $\boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1)$ задания Z_l агентом AR_p

Последнее может быть осуществлено с помощью одного из известных алгоритмов поиска максимального пути на графе [17]. Отметим, поскольку изначально, в момент размещения дескриптора задания Z_l на ДО, требуемый момент времени исполнения приписан только одной конечной вершине q_k графа $\boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1)$ (этот момент времени определяется требуемым моментом времени T_{max}^l выполнения всего задания Z_l в целом, т. е. $T_{k+1}^1 = T_{max}^l$), то ветвь $\boldsymbol{H}_1 =< q_1^1, q_2^1, ..., q_k^1 >$ должна заканчиваться на конечной вершине графа $\boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1)$, т.е. $q_k^1 = q_k$ (см. рис. 2.2).

На основании данных о периодах времени $t_p(O_i^1)$ $(i=1,2,...,k)$ выполнения отдельных операций, входящих в выделенную ветвь $\boldsymbol{H}_1$, агент AR_p определяет, согласно (2.2), момент времени T_1^1, когда необходимо приступить к выполнению первой операции ветви $\boldsymbol{H}_1$ (т. е. операции, приписанной вершине $q_1^1 \in \boldsymbol{H}_1$), чтобы успеть завершить исполнение всей ветви $\boldsymbol{H}_1$ к заданному моменту времени T_{k+1}^1, как:

$$T_1^1 = T_{k+1}^1 - \left(\sum_{i=d}^{k} t_p(O_i^1) + t_{\text{П}} \right),$$

где $t_p(O_i^1) = \frac{v(O_i^1)}{D_p(O_i^1)}$ $(d=1,2,...,k)$.

Если при этом оказывается, что $T_1^1 < T_{\text{тек}}$, где $T_{\text{тек}}$ – текущий момент времени; T_1^1 – требуемый момент времени начала выполнения первой операции ветви $\boldsymbol{H}_1$, то это означает, что ресурс R_p не может обеспечить выполнение всей последовательности операций ветви $\boldsymbol{H}_1 =< q_1^1, q_2^1, ..., q_k^1 >$ к установленному моменту времени T_{k+1}^1. Поскольку в данном варианте полагается, что все ресурсы СРС одинаковы и выполняют идентичные операции за одинаковое время, то это также означает, что никакой другой ресурс $R_j \in \boldsymbol{R}$ $(j=1,2,...,p-1,p+1,...,N)$ также не сможет выполнить данную ветвь $\boldsymbol{H}_1$ к установленному моменту времени T_{k+1}^1. При этом значение $\Delta T^l = T_{\text{тек}} - T_1^1$ будет определять величину задержки времени выполнения задания Z_l относительно требуемого T_{max}^i.

Далее агент AR_p принимает на себя исполнение последовательности операций, приписанных вершинам ветви $\boldsymbol{H}_1$. При этом агент AR_p осуществляет модификацию дескриптора задания Z_l на ДО, а именно:

1. Его номер p записывается в список участников сообщества $\boldsymbol{R}_l$ по выполнению задания Z_l.

2. Всем вершинам $q_i^1 \in \boldsymbol{H_1}$ ветви $\boldsymbol{H_1}$ приписываются моменты времени T_d^1 начала их исполнения, определяемые как

$$T_d^1 = T_{k+1}^1 - \left(\sum_{i=d}^{k} t_p(O_i^1) + t_{\text{П}} \right) \ (d=1,2,...,k),$$

где O_i^1 – операция, приписываемая вершине $q_i^1 \in \boldsymbol{H_1}$.

3. Вершины, входящие в ветвь $\boldsymbol{H}_1$, исключаются из графа $\boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1)$, в результате чего формируется новый граф $\boldsymbol{G}_l^2(\boldsymbol{Q}_l^2, \boldsymbol{X}_l^2) = \boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1) / \mathbf{H}_1$ (рис. 2.3).

4. Всем вершинам модифицированного графа $\boldsymbol{G}_l^2(\boldsymbol{Q}_l^2, \boldsymbol{X}_l^2)$, инцидентным вершинам ветви $\boldsymbol{H}_1$, приписываются требуемые моменты времени их исполнения, которые определяются исходя из следующих соображений. Допустим, что некоторая вершина q_f^2 графа $\boldsymbol{G}_l^2(\boldsymbol{Q}_l^2, \boldsymbol{X}_l^2)$

инцидентна вершине q_b^1, принадлежащей ветви $\boldsymbol{H}_1$ (см. рис. 2.3). Это означает, что для выполнения операции, приписанной вершине q_b^1, необходимы результаты выполнения операции, приписанной вершине q_f^2 графа $\boldsymbol{G}_l^2(\boldsymbol{Q}_l^2, \boldsymbol{X}_l^2)$. Поэтому очевидно, что результаты выполнения операции, приписанной вершине q_f^2, должны быть получены и переданы ресурсу R_p, выполняющему операции ветви $\boldsymbol{H}_1$, не позже чем к требуемому моменту времени окончания выполнения операции вершины q_{b-1}^1, определяемому как:

$$T_b^1 = T_{k+1}^1 - \left(\sum_{i=b}^{k} t_p(O_i^1) + t_{\text{П}} \right). \qquad (2.3)$$

Поэтому вершине q_f^2 графа $\boldsymbol{G}_l^2(\boldsymbol{Q}_l^2, \boldsymbol{X}_l^2)$ приписывается требуемое время ее исполнения $T_{f+1}^2 = T_b^1$, а также номер ресурса R_p, которому результаты исполнения операции q_f^2 должны быть переданы (см. рис. 2.3).

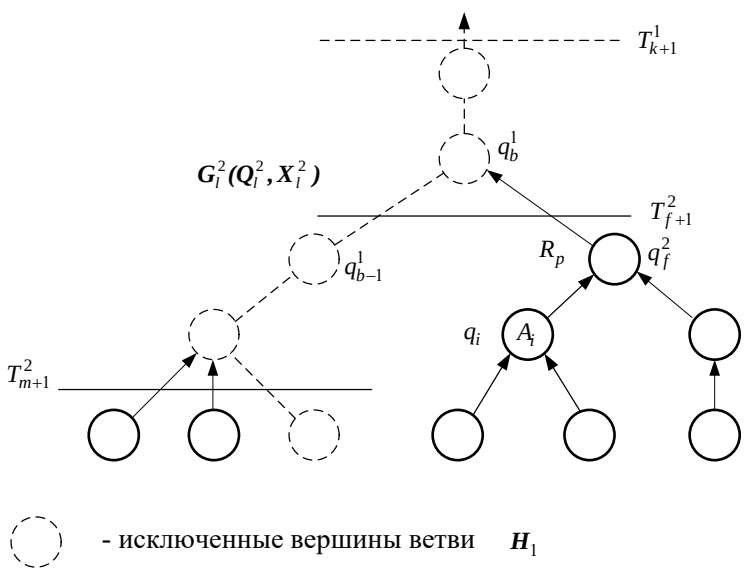


Рис. 2.3. Граф $\boldsymbol{G}_l^2(\boldsymbol{Q}_l^2, \boldsymbol{X}_l^2)$ задания Z_l, модифицированный агентом AR_p

Аналогичным образом определяются и приписываются всем остальным вершинам графа $\boldsymbol{G}_l^2(\boldsymbol{Q}_l^2, \boldsymbol{X}_l^2)$, инцидентным вершинам ветви $\boldsymbol{H}_1$, требуемые моменты времени T_{m+1}^2 их исполнения (см. рис. 2.3).

Если после модификации новый граф $\boldsymbol{G_l^2(Q_l^2, X_l^2)}$ задания Z_l еще не пустой, т. е. $\boldsymbol{G_l^2(Q_l^2, X_l^2)} \neq \varnothing$, то процесс создания сообщества $\boldsymbol{R}_l$ для выполнения задания Z_l продолжается далее. Допустим, что следующий по очереди (по номеру) агент AR_c свободного ресурса (т. е. ресурса, не задействованного в выполнение других заданий) опрашивает ДО и обнаруживает дескриптор задания Z_l. Агент AR_c делает попытку войти в состав сообщества $\boldsymbol{R}_l$ по исполнению данного задания. Для этого агент AR_c выделяет в графе $\boldsymbol{G_l^2(Q_l^2, X_l^2)}$ (который остался в дескрипторе задания Z_l на ДО после модификации, произведенной агентом AR_p) наиболее длинную ветвь $\boldsymbol{H}_2 =< q_1^2, q_2^2, ..., q_f^2 >$ (рис. 2.4) конечной вершины q_f^2, которой приписано требуемое время исполнения T_{f+1}^2, и определяет, согласно выражению (2.2), момент времени, когда необходимо приступить к ее выполнению, как:

$$T_1^2 = T_{f+1}^1 - \left(\sum_{i=1}^{f} t_c(O_i^2) + t_{\text{П}} \right),$$

где T_{f+1}^2 – момент времени, приписанный конечной вершине q_f^2 ветви $\boldsymbol{H}_2$;

O_i^2 – операция, приписанная вершине $q_i^2 \in \boldsymbol{H}_2$ $\boldsymbol{(}i = 1,2,..., f\boldsymbol{)}$

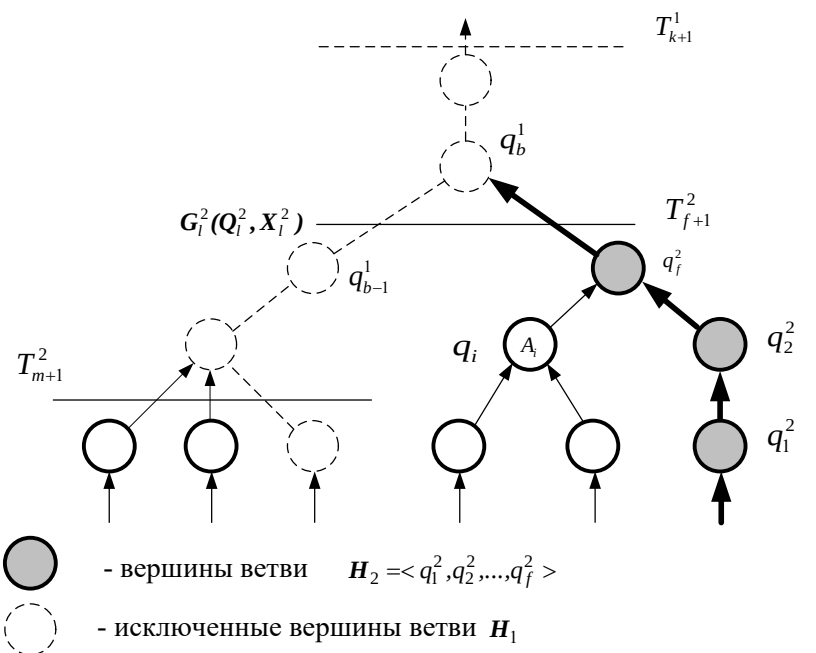


Рис. 2.4. Выделение ветви $\boldsymbol{H}_2$ на графе $\boldsymbol{G_l^2(Q_l^2, X_l^2)}$ задания Z_l агентом AR_c

Если $T_1^2 < T_{\text{тек}}$, где $T_{\text{тек}}$ – текущий момент времени, то это означает, что данная ветвь не может быть выполнена к установленному моменту времени T_{f+1}^2 никаким ресурсом СРС, а время задержки выполнения задания Z_l будет уже составлять

$$\Delta T^l = \Delta T^l + (T_{\text{тек}} - T_1^2).$$

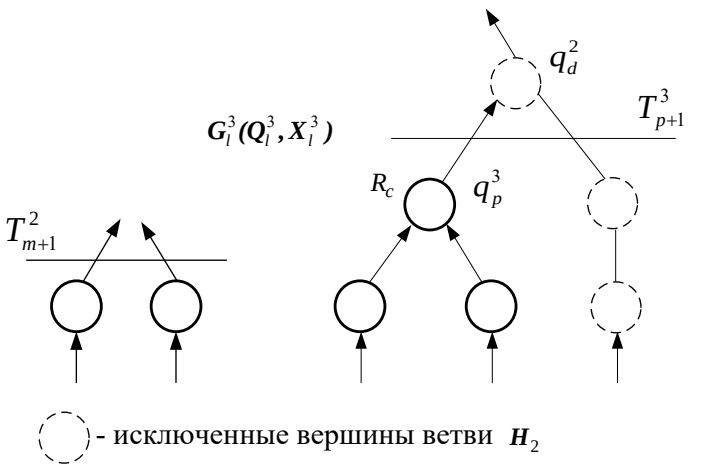


Рис. 2.5. Граф $G_l^3(Q_l^3, X_l^3)$ задания Z_l после модификации агентом AR_c

Агент AR_c принимает на себя исполнение операций, приписанных вершинам нити H_2, и осуществляет очередную модификацию дескриптора задания Z_l на ДО:

1. Номер агента AR_c записывается в список участников сообщества R_l по выполнению задания Z_l.

2. Всем вершинам $q_d^2 \in H_2$ нити H_2 приписывается номер ресурса R_c и моменты времени T_d^2 начала их исполнения, определяемые как

$$T_d^2 = T_{f+1}^2 - \left(\sum_{i=d}^{f} t_c(O_i^2) + t_{\text{П}}\right) \ (d = 1,2,\ldots, f).$$

3. Вершинам $q_b^2, q_{b+1}^2, \ldots, q_k^1$ ветви H_1 приписывается новое время начала их исполнения, определяемое как (см. рис. 2.4.):

$$T_i^2 = T_i^2 + (T_{\text{тек}} - T_1^2) \ (i = b, b+1, \ldots, k),$$

где $(T_{\text{тек}} - T_1^2)$ – время задержки выполнения ветви H_2 относительно требуемого момента T_{f+1}^2.

4. Вершины, входящие в ветвь $\boldsymbol{H}_2$, исключаются из графа $\boldsymbol{G}_l^2(\boldsymbol{Q}_l^2, \boldsymbol{X}_l^2)$, в результате чего образуется новый граф $\boldsymbol{G}_l^3(\boldsymbol{Q}_l^3, \boldsymbol{X}_l^3) = \boldsymbol{G}_l^2(\boldsymbol{Q}_l^2, \boldsymbol{X}_l^2) / \mathbf{H}_2$ (рис. 2.5).

5. Каждой вершине q_p^3 графа $\boldsymbol{G}_l^3(\boldsymbol{Q}_l^3, \boldsymbol{X}_l^3)$, инцидентной вершине q_d^2 ветви $\boldsymbol{H}_2$, приписывается требуемый момент времени ее исполнения, определяемый как

$$T_{p+1}^3 = T_{f+1}^2 - \left(\sum_{i=d}^{f} t_c(O_i^2) + t_{\text{п}} \right),$$

а также номер ресурса R_c, которому необходимо передать результаты исполнения данной операции (рис. 2.5).

Далее в процесс распределения операций задания Z_l включается следующий по очереди агент свободного ресурса, который выделяет на модифицированном агентом AR_c графе $\boldsymbol{G}_l^3(\boldsymbol{Q}_l^3, \boldsymbol{X}_l^3)$ самую длинную ветвь $\boldsymbol{H}_3$ (рис. 2.6) и т. д. до тех пор, пока не окажется, что после очередной модификации граф задания $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ стал пустым, что означает, что все операции задания Z_l разобраны агентами, вошедшими в сообщество $\boldsymbol{R}_l$ по его выполнению.

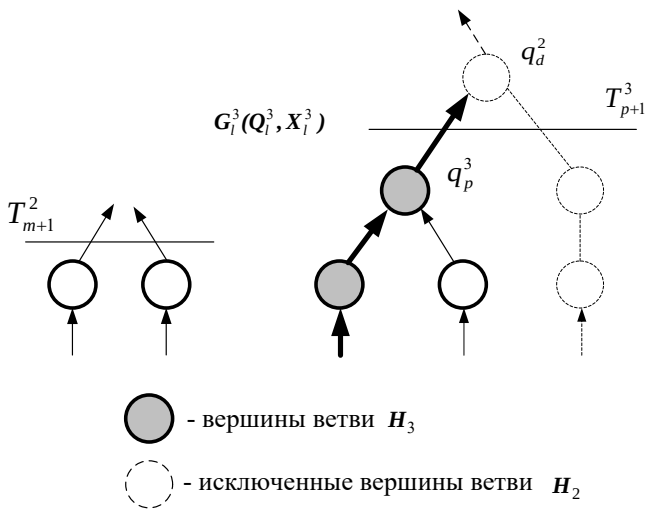


Рис. 2.6. Выделение агентом AR_c ветви $\boldsymbol{H}_3$ на графе задания $\boldsymbol{G}_l^3(\boldsymbol{Q}_l^3, \boldsymbol{X}_l^3)$

После того, как некоторый агент AR_p вошел в состав сообщества $\boldsymbol{R}_l$ по выполнению задания Z_l и выбрал для исполнения ветвь $\boldsymbol{H}_f = < q_1^f, q_2^f, ..., q_k^f >$, он инициирует выполнение операций, приписанных ее вершинам, с помощью локальной УУ$_p$ «своего» ресурса R_p.

В результате выполнения описанной выше процедуры каждой вершине $q_d \in \boldsymbol{Q}_l$ графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l будет приписан момент времени T_d начала ее выполнения и номер ресурса, ответственного за ее выполнение. Кроме того, будет определена суммарная величина задержки ΔT^l выполнения всего задания Z_l относительно требуемого T^l_{max}.

Описанной выше процедуре самоорганизации в гомогенной СРС отвечает следующий алгоритм функционирования агента AR_p ресурса R_p ($p = 1,2,...,N$). При этом полагаем, что в момент размещения задания Z_l на ДО счетчик итераций устанавливается в 1, т. е. $j = 1$, а также $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) = \boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ и $\Delta T^l = 0$.

Алгоритм 2.1

1. Агент AR_p свободного ресурса R_p, не задействованного в выполнении каких-либо заданий, опрашивает ДО в порядке своей очереди.

2. При обнаружении на ДО задания Z_l агент AR_p считывает дескриптор графа задания $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$.

3. Агент AR_p выделяет в графе $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ наиболее длинную ветвь $\boldsymbol{H}_j = < q_1^j, q_2^j, ..., q_k^j >$, согласно выражению (2.1), конечной вершине q_k^j которой приписан требуемый момент времени ее исполнения T_{k+1}^j (если $\boldsymbol{H}_j = \boldsymbol{H}_1$, то $T_{k+1}^j = T_{max}^l$), и определяет допустимый момент времени, когда необходимо начать ее выполнение как

$$T_1^j = T_{k+1}^j - \left(\sum_{i=1}^{k} t_p(O_i^j) + t_{\text{п}} \right), \text{ где } t_p(O_i^j) = \frac{v(O_i^j)}{D_p(O_i^j)}.$$

4. $$\Delta T^l = \begin{cases} \Delta T^l + (T_{\text{тек}} - T_1^j), \text{если } T_{\text{тек}} - T_1^j > 0; \\ 0, \text{если } T_{\text{тек}} - T_1^j \le 0. \end{cases}$$

5. Агент AR_p принимает на себя выполнение операций ветви $\boldsymbol{H}_j$, записывает в список участников сообщества $\boldsymbol{R}_l$ свой номер и модифицирует граф $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ задания Z_l на ДО:

– всем вершинам $q_d^j \in \boldsymbol{H}_j$ $(d = 1, 2, \dots, k)$ приписывается номер ресурса R_p и момент времени T_d начала их исполнения

$$T_d^j = T_{k+1}^j - \left(\sum_{i=d}^{k} t_p(O_i^j) + t_n \right);$$

– если вершина q_{k+1}^j инцидента вершине q_b^{j-1} ветви $\boldsymbol{H}_{j-1} = \langle q_1^{j-1}, q_2^{j-1}, \dots, q_L^{j-1} \rangle$, то моменты времени начала исполнения, приписанные вершинам $q_b^{j-1}, q_{b+1}^{j-1}, \dots, q_L^{j-1}$ ветви $\boldsymbol{H}_{j-1}$, увеличиваются на величину $(T_{\text{тек}} - T_1^j)$;

– вершины ветви $\boldsymbol{H}_j$ исключаются из графа $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$, в результате чего формируется новый граф $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1}) = \boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) / \boldsymbol{H}_j$;

– каждой вершине q_b^{j+1} модифицированного графа $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1})$, инцидентной вершине q_d^j ветви $\boldsymbol{H}_j$, приписывается требуемый момент времени ее исполнения, определяемый как

$$T_{b+1}^{j+1} = T_{k+1}^j - \left(\sum_{i=d}^{k} t_p(O_i^j) + t_{\text{п}} \right),$$

а также номер ресурса R_p, которому необходимо передать результаты исполнения операции, приписанной вершине q_b^{j+1}.

6. Счетчик итераций на ДО увеличивается на единицу, т. е. $j = j + 1$. Модифицированный граф $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$, а также новое значение ΔT^l размещаются в дескрипторе задания Z_l на ДО. Если $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) \neq \varnothing$, то процесс распределения операций задания Z_l заканчивается.

7. Ресурс R_p приступает к выполнению последовательности операций, приписанных вершинам ветви $\boldsymbol{H}_j$ согласно установленному временному графику (т. е. приписанным вершинам ветви $\boldsymbol{H}_j$ моментам времени начала их исполнения).

8. После завершения выполнения всех операций ветви $\boldsymbol{H}_j$ переход к 1.

2.2. Алгоритм мультиагентного диспетчирования ресурсов в гомогенной СРС второго типа

В отличие от полностью гомогенной СРС, рассмотренной в предыдущем разделе, в данном случае производительность различных ресурсов $R_i \in \boldsymbol{R}$ и $R_j \in \boldsymbol{R}$ $(i = 1,2,...,N;\ j = 1,2,...,i-1,i+1,...,N)$ при выполнении идентичных операций $O_l \in \boldsymbol{O}$ также различна, т. е. $D_i(O_l) \neq D_j(O_l)$, и, соответственно, периоды времени их выполнения будут разными, т. е. $t_i(O_l) = \frac{v(O_l)}{D_i(O_l)} \neq t_j(O_l) = \frac{v(O_l)}{D_j(O_l)}$ $(i = 1,2,...,N,\ j = 1,2,...,i-1,i+1,...,N)$.

Следовательно, время выполнения операций одной и той же нити $\boldsymbol{H}_f$ различными ресурсами $R_p \in \boldsymbol{R}$ также будет различным. Поэтому очевидно, что в данном случае ветвь $\boldsymbol{H}_f$ графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l целесообразно закрепить за тем ресурсом R_c, который обеспечивает минимальную задержку $\Delta T_c^f = min(\Delta T_p^f)$ $(p = 1,2, \dots, N)$ ее выполнения. Однако при этом следует учесть то обстоятельство, что агент AR_p свободного ресурса R_p, обнаруживший на ДО задание $Z_l \in \boldsymbol{Z}$, не может знать, имеется ли в составе СРС другой ресурс, способный обеспечить меньшую задержку при выполнении ветвей графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, чем его ресурс R_p. Более того, если даже такой ресурс и существует, агент AR_p не может знать, когда он освободится и сможет ли вообще принять участие в выполнении задания Z_l [18]. Поэтому целесообразно, чтобы среди всего множества нитей графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, которым приписаны моменты времени их исполнения, агент AR_p выбрал ту нить $\boldsymbol{H}_f$, при исполнении которой «его» ресурс R_p обеспечивает минимальную задержку ΔT_p^f. Иными словами, при обнаружении на ДО некоторого задания Z_l агент AR_p должен последовательно перебрать все его ветви, которым приписано требуемое время исполнения, и выбрать для исполнения ту ветвь, которую его ресурс R_p может выполнить с наименьшей задержкой ΔT_p^f.

При этом, как уже было отмечено выше, в момент размещения задания Z_l на ДО требуемый момент времени исполнения T_{max}^l будет приписан только конечной вершине q_k графа $\boldsymbol{G_l(Q_l, X_l)} = \boldsymbol{G_l^1(Q_l^1, X_l^1)}$. Поэтому первый агент AR_p, обнаруживший задание Z_l на ДО, примет к исполнению самую длинную ветвь $\boldsymbol{H}_1 =< q_1^1, q_2^1, ..., q_k^1 >$ графа $\boldsymbol{G_l^1(Q_l^1, X_l^1)}$, заканчивающуюся на конечной вершине q_k графа $\boldsymbol{G_l(Q_l, X_l)}$, т. е. $q_k^1 = q_k$, и, следовательно, задержка ΔT^l выполнения задания Z_l будет не менее, чем ΔT_p^1, т. е. $\Delta T^l \geq \Delta T_p^1$, где ΔT_p^1 – задержка выполнения ветви $\boldsymbol{H}_1$ ресурсом R_p.

После того как некоторая ветвь $\boldsymbol{H}_1$ принята к исполнению агентом AR_p, он осуществляет модификацию дескриптора графа $\boldsymbol{G_l^1(Q_l^1, X_l^1)}$ на ДО в соответствии с процедурой, описанной в подразделе 2.1, и далее переходит к исполнению операций данной ветви с помощью «своего» ресурса R_p.

Процесс распределения операций задания Z_l агентами AR_j $(j = 1,2, \dots, N)$ аналогично будет продолжаться до тех пор, пока все ветви графа $\boldsymbol{G_l(Q_l, X_l)}$ задания Z_l не будут разобраны агентами ресурсов, вступившими в сообщество $\boldsymbol{R}_l$ по его выполнению (т. е. пока, после очередной модификации графа задания Z_l, не окажется, что $\boldsymbol{G_l^j(Q_l^j, X_l^j)} = \varnothing$).

Исходя из приведенных выше соображений, можно предложить следующий алгоритм функционирования агента AR_p некоторого ресурса $R_p \in \boldsymbol{R}$ при выполнении процедуры мультиагентного диспетчирования ресурсов в гомогенной СРС второго типа (с различной производительностью ресурсов). При этом, опять-таки, полагаем, что в момент размещении задания Z_l на ДО счетчик итераций $j = 1$; $\boldsymbol{G_l^j(Q_l^j, X_l^j)} = \boldsymbol{G_l(Q_l, X_l)}$ и $\Delta T^l = 0$.

Алгоритм 2.2

1. Агент AR_p свободного ресурса R_p опрашивает ДО в порядке своей очереди (нумерации).

2. При обнаружении на ДО задания Z_l агент AR_p считывает дескриптор графа $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$.

3. $i=1$; $\boldsymbol{G}_i(\boldsymbol{Q}_i, \boldsymbol{X}_i) = \boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$; $\Delta T_p = \infty$.

4. Агент AR_p выделяет в графе $\boldsymbol{G}_i(\boldsymbol{Q}_i, \boldsymbol{X}_i)$ наиболее длинную ветвь $\boldsymbol{H}_i = < q_1^i, q_2^i, ..., q_k^i >$, согласно выражению (2.1), конечной вершины q_l^i которой приписан требуемый момент времени исполнения T_{k+1}^i, и определяет момент времени T_k^i, когда он должен приступить к ее выполнению как

$$T_1^i = T_{k+1}^i - \left(\sum_{m=d}^{k} t_p(O_m^i) + t_{\text{п}} \right), \text{ где } t_p(O_m^i) = \frac{v(O_m^i)}{D_p(O_m^i)}.$$

5. $\Delta T_p^i = \begin{cases} T_{\text{тек}} - T_1^i, \text{если } T_{\text{тек}} - T_1^i > 0; \\ 0, \text{если } T_{\text{тек}} - T_1^i \le 0. \end{cases}$

6. Если $\Delta T_p^i < \Delta T_p$, то $\Delta T_p = \Delta T_p^i$ и $\boldsymbol{H}_j = \boldsymbol{H}_i$.

7. Ветвь $\boldsymbol{H}_i$ исключается из графа $\boldsymbol{G}_i(\boldsymbol{Q}_i, \boldsymbol{X}_i)$, т. е. $\boldsymbol{G}_{i+1}(\boldsymbol{Q}_{i+1}, \boldsymbol{X}_{i+1}) = \boldsymbol{G}_i(\boldsymbol{Q}_i, \boldsymbol{X}_i) / \boldsymbol{H}_i$

8. Если в графе $\boldsymbol{G}_{i+1}(\boldsymbol{Q}_{i+1}, \boldsymbol{X}_{i+1})$ остались вершины, которым приписано требуемое время исполнения, то $i = i+1$ и перейти к 4, иначе

9. $\Delta T^l = \Delta T^l + \Delta T_p$.

10. Агент AR_p принимает на себя исполнение ветви $\boldsymbol{H}_j$, записывает в список участников сообщества $\boldsymbol{R}_l$ свой номер p и модифицирует дескриптор граф $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ задания Z_l на ДО:

– всем вершинам $q_d^j \in \boldsymbol{H}_j$ приписывается номер ресурса R_p и момент времени T_d начала их исполнения

$$T_d^j = T_{k+1}^j - \left(\sum_{i=d}^{k} t_p(O_i^j) + t_n \right);$$

– если вершина q_{k+1}^j инцидента вершине q_b^{j-1} ветви $\boldsymbol{H}_{j-1}$, ранее закрепленной за другим ресурсом R_c, то моменты времени начала использования, приписанные вершинам $q_b^{j-1}, q_{b+1}^{j-1}, \ldots, q_L^{j-1}$ ветви $\boldsymbol{H}_{j-1}$, увеличиваются на величину ΔT_p.

– вершины ветви $\boldsymbol{H}_j$ исключаются из графа $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$, в результате чего формируется новый граф $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1}) = \boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) \big/ \boldsymbol{H}_j$;

– каждой вершине q_b^{j+1} модифицированного графа $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1})$, инцидентной вершине q_d^j ветви $\boldsymbol{H}_j$, требуемый момент времени ее исполнения, определяемый как

$$T_{b+1}^{j+1} = T_{k+1}^j - \left(\sum_{i=d}^{k} t_p(O_i^j) + t_{\text{п}} \right),$$

а также номер ресурса R_p, которому необходимо передать результаты исполнения операции, приписанной вершине q_b^{j+1}.

11. Счетчик итераций на ДО увеличивается на единицу, т. е. $j = j + 1$. Модифицированный граф $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$, а также новое значение ΔT^l размещаются на ДО. Если $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) \neq \varnothing$, то процесс распределения операций задания Z_l заканчивается.

12. Ресурс R_p приступает к выполнению последовательности операций, приписанных вершинам выбранной его агентом ветви $\boldsymbol{H}_j$ согласно установленному временному графику (т. е. приписанным вершинам ветви $\boldsymbol{H}_j$ моментам времени начала их исполнения).

13. После завершения выполнения всех операций ветви $\boldsymbol{H}_j$ переход к 1.

2.3. Алгоритм мультагентного диспетчирования ресурсов в гетерогенной СРС первого типа

В этом варианте полагается, что все ресурсы $\boldsymbol{R}_i$ $(i=1,2,...,N)$ СРС выполняют различные наборы операций $\boldsymbol{O}_i$ $(i=1,2,...,N)$, причем в общем случае $\boldsymbol{O}_i \cap \boldsymbol{O}_j \neq \emptyset$ $(i=1,2,...,N, j=1,2,...,i-1,i+1,...,N)$, однако все они имеют равную производительность при выполнении идентичных операций O_l, т.е. $D_i(O_l) \neq D_j(O_l)$ $(i=1,2,...,N, j=1,2,...,i-1,i+1,...,N)$. Следовательно, и период времени выполнения идентичных операций различными ресурсами $R_i \in \boldsymbol{R}$ и $R_j \in \boldsymbol{R}$ также одинаков, т. е. $t_i(O_l)=\frac{v(O_l)}{D_i(O_l)}=t_j(O_l)=\frac{v(O_l)}{D_j(O_l)}$ [19].

Рассмотрим основные принципы организации процедуры мультиагентного диспетчирования ресурсов СРС при данных условиях.

Допустим, что агент AR_p свободного ресурса $R_p \in \boldsymbol{R}$ обнаружил на ДО задание Z_l. В отличие от предыдущих вариантов организации СРС, в данном случае может оказаться, что объект R_p способен выполнять не все операции, приписанные вершинам графа $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ задания Z_l, хранимого на ДО. Поэтому прежде всего агент AR_p должен в графе $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ выделить подграф $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp}, \boldsymbol{X}_l^{jp})$, вершинам которого приписаны операции множества $\boldsymbol{O}_p$, выполняемого узлом R_p (рис. 2.7).

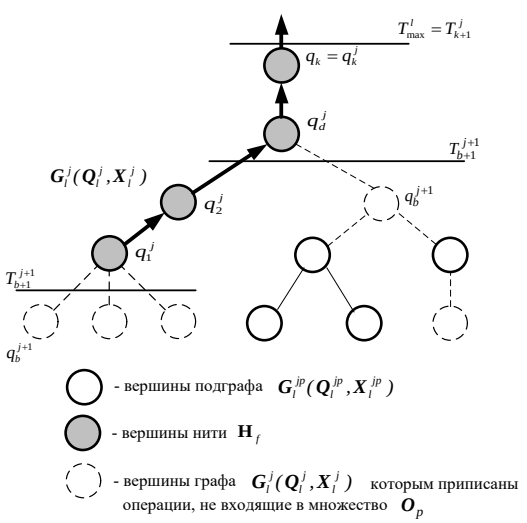


Рис.2.7. Выделение подграфа $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp}, \boldsymbol{X}_l^{jp})$ в графе $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ задания Z_l

После этого агент AR_p должен проанализировать, есть ли среди вершин графа $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp},\boldsymbol{X}_l^{jp})$ вершины, которым приписаны требуемые моменты времени их исполнения (при размещении задания Z_l на ДО требуемый момент времени исполнения T_{max}^l приписан только конечной вершине q_k графа $\boldsymbol{G}_l(\boldsymbol{Q}_l,\boldsymbol{X}_l)$). Если таковых вершин нет, то это говорит о том, что агент AR_p пока что не может вступить в сообщество $\boldsymbol{R}_l$ по выполнению задания Z_l. В противном случае агент AR_p аналогично тому, как описано в подразделе 2.1, выделяет в графе $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp},\boldsymbol{X}_l^{jp})$ с помощью выражения (2.1) наиболее длинную ветвь $\boldsymbol{H}_j =< q_1^j, q_2^j, ..., q_k^j >$, конечной вершине q_k^j которой приписан требуемый момент времени исполнения T_{k+1}^j, и определяет момент времени $T_1{}^j$ начала исполнения операции, как (см. рис. 2.7):

$$T_1{}^j = T_{k+1}^j - \left(\sum_{i=1}^{k} t_p(O_i^j) + t_{\text{п}} \right),$$

где $t_p(O_i^j) = \dfrac{v(O_i^j)}{D_p(O_i^j)}$.

Если $T_{\text{тек}} > T_1{}^j$, то это означает, что ресурс R_p может выполнить данную ветвь $\boldsymbol{H}_j$ с задержкой $\Delta T_p^j = T_{\text{тек}} - T_1{}^j$ относительно требуемого момента времени T_{k+1}^j. Поскольку в данном варианте принято, что все ресурсы множества $\boldsymbol{R}$ выполняют идентичные операции за одинаковое время, то это также означает, что никакой другой ресурс $R_c \in \boldsymbol{R}$ также не сможет выполнить данную ветвь $\boldsymbol{H}_j$ быстрее.

Поэтому агент AR_p входит в состав сообщества $\boldsymbol{R}_l$ по выполнению задания Z_l и выполняет процедуру модификации графа задания на ДО: исключает ветвь $\boldsymbol{H}_j$ из графа задания $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j,\boldsymbol{X}_l^j)$, в результате чего формируется новый граф задания $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1},\boldsymbol{X}_l^{j+1}) = \boldsymbol{G}_l^j(\boldsymbol{Q}_l^j,\boldsymbol{X}_l^j) \,/\, \boldsymbol{H}_j$, а каждой вершине модифицированного графа $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1},\boldsymbol{X}_l^{j+1})$, инцидентной вершине ветви $\boldsymbol{H}_j$, приписывает требуемый момент времени ее исполнения, определяемый согласно выражению (2.3), а также номер узла R_p, которому должны быть переданы результаты исполнения приписанной ей операции.

Процесс распределения операций задания Z_l между агентами ресурсов множества $\boldsymbol{R}$ заканчивается, когда после очередной

модификации граф задания $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1},\boldsymbol{X}_l^{j+1})$ на ДО становится пустым, что означает, что все его операции распределены между узлами множества $\boldsymbol{R}$.

После того, как агент AR_p выбрал некоторую ветвь $\boldsymbol{H}_j$, он приступает к ее исполнению.

Исходя из приведенных выше соображений можно предложить следующий алгоритм работы агента AR_p ресурса R_p при реализации процедуры мультиагентного диспетчирования гетерогенной СРС первого типа (с ресурсами разной специализации, но равной производительности). При этом, как и в предыдущих случаях, полагаем, что при размещении задания Z_l на ДО $j = 1$; $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j,\boldsymbol{X}_l^j) = \boldsymbol{G}_l(\boldsymbol{Q}_l,\boldsymbol{X}_l)$ и $\Delta T^l = 0$.

Алгоритм 2.3

1. Агент AR_p свободного ресурса R_p опрашивает ДО в порядке своей очереди.

2. При обнаружении на ДО задания Z_i агент AR_p считывает дескриптор графа задания $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j,\boldsymbol{X}_l^j)$.

3. В графе $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j,\boldsymbol{X}_l^j)$ выделяется подграф $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp},\boldsymbol{X}_l^{jp})$, вершинам которого приписаны операции множества $\boldsymbol{O}_p$, выполняемые ресурсом R_p.

4. Если $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp},\boldsymbol{X}_l^{jp}) = \varnothing$ или подграфа $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp},\boldsymbol{X}_l^{jp})$ не содержит ни одной вершины, которой приписан требуемый момент времени T_{k+1}^j ее исполнения, то перейти к 1, иначе

5. Агент AR_p выделяет в подграфе $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp},\boldsymbol{X}_l^{jp})$ наиболее длинную ветвь $\boldsymbol{H}_j = < q_1^j, q_2^j, ..., q_k^j >$, конечной вершине которой приписан требуемый момент времени ее исполнения T_{k+1}^j (в момент размещения задания Z_i на ДО требуемый момент времени исполнения $T_{k+1}^j = T_{max}^l$ приписан только конечной вершине q_k графа $\boldsymbol{G}_l(\boldsymbol{Q}_l,\boldsymbol{X}_l)$) и определяет требуемый момент времени T_1^j, когда необходимо начать ее выполнение как:

$$T_1^j = T_{k+1}^j - \left(\sum_{i=1}^{k} t_p(O_1^j) + t_{\text{п}}\right),$$

где $t_p(O_1^j) = \dfrac{v(O_1^j)}{D_p(O_1^j)}$ – период времени, затрачиваемый узлом R_p на выполнение операции O_i^j, приписанной вершине q_i^j $(i = 1,2,...,k)$ ветви $\boldsymbol{H}_j$.

6. $\Delta T_p^j = \begin{cases} T_{\text{тек}} - T_1^j, \text{если } T_{\text{тек}} - T_1^j > 0; \\ 0, \text{если } T_{\text{тек}} - T_1^j \le 0. \end{cases}$

$\Delta T^l = \Delta T^l + \Delta T_p^j$.

7. Агент AR_p принимает на себя исполнение ветви $\boldsymbol{H}_j$, записывает в список участников сообщества $\boldsymbol{R}_l$ свой номер p, а также модифицирует граф $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$:

– из графа задания $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ исключаются вершины ветви $\boldsymbol{H}_j$, в результате чего формируется новый граф задания $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1}) = \boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) / \boldsymbol{H}_j$;

– каждой вершине q_b^{j+1} модифицированного графа $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1})$, инцидентной вершине q_d^j ветви $\boldsymbol{H}_j$, приписывается требуемый момент времени ее исполнения, определяемый как (см. рис. 2.7)

$$T_{b+1}^{j+1} = T_{k+1}^j - \left(\sum_{i=d}^{k} t_p(O_i^j) + t_{\text{п}} \right), \text{ где } t_p(O_i^j) = \frac{v(O_i^j)}{D_p(O_i^j)},$$

а также номер ресурса R_p, которому необходимо передать результаты исполнения операции, приписанной вершине q_b^{j+1};

– если вершина q_{k+1}^j инцидента вершине q_b^{j-1} ветви $\boldsymbol{H}_{j-1}$, то моменты времени начала использования, приписанные вершинам $q_b^{j-1}, q_{b+1}^{j-1}, \ldots, q_L^{j-1}$ ветви $\boldsymbol{H}_{j-1}$, увеличиваются на величину ΔT_p^j.

8. Счетчик итераций на ДО увеличивается на единицу, т. е. $j = j + 1$. Модифицированный граф $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$, а также новое значение ΔT^l размещаются на ДО. Если $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) \neq \varnothing$, то процесс распределения операций задания Z_l заканчивается.

9. Ресурс R_p приступает к выполнению последовательности операций, приписанных вершинам ветви $\boldsymbol{H}_j$, согласно установленному временному графику (т. е. приписанным вершинам ветви $\boldsymbol{H}_j$ моментам времени начала их исполнения).

10. После завершения выполнения всех операций ветви $\boldsymbol{H}_j$ переход к 1.

2.4. Алгоритм мультиагентного диспетчирования ресурсов в гетерогенной СРС второго типа

В самом общем случае все ресурсы множества $\boldsymbol{R} =< R_1, R_2, ..., R_N >$ выполняют различные наборы операции, т. е. $\boldsymbol{O}_i \neq \boldsymbol{O}_j$ $(i = 1,2,...,N, j = 1,2,...,i-1,i+1,...,N)$, причем производительность различных ресурсов при выполнении идентичных операций также различна, т. е. $D_p(O_i) \neq D_c(O_i)$. (p = 1, 2, …, N и c = 1, 2, …,p – 1, p+1, N) [20].

При этом, как и в предыдущем варианте, агент свободного ресурса AR_p, обнаруживший на ДО дескриптор задания Z_l, должен первым делом выделить в графе $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ подграф $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp}, \boldsymbol{X}_l^{jp})$, вершинам которого приписаны операции, входящие во множество операций $\boldsymbol{O}_p$, выполняемых «его» ресурсом R_p. Далее в подграфе $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp}, \boldsymbol{X}_l^{jp})$ агент AR_p должен выделить наиболее длинную ветвь $\boldsymbol{H}_1$, конечной вершине q_k^j которой приписан требуемый момент времени ее исполнения T_{k+1}^1, и определить задержку ее исполнения к данному моменту времени с помощью «его» ресурса R_p. Для этого он определяет требуемый момент времени начала исполнения операции, приписанной первой вершине данной ветви $\boldsymbol{H}_1$ как $T_1^1 = T_{k+1}^1 - \left(\sum_{i=1}^{k} t_p(O_i^j) + t_{\text{п}} \right)$, где $t_p(O_i^j) = \dfrac{v(O_i^j)}{D_p(O_i^j)}$, и сравнивает его с текущим моментом времени $T_{\text{тек}}$. Если $T_{\text{тек}} > T_1^1$, то это означает, что ресурс R_p может выполнить операции данной ветви с задержкой $\Delta T_p^l = T_{\text{тек}} - T_1^1$ относительно требуемого момента времени T_{k+1}^j. Далее аналогичным образом агент AR_p анализирует следующие по убыванию длины ветви подграфа $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp}, \boldsymbol{X}_l^{jp})$, которым приписано требуемое время исполнения, до тех пор, пока он не найдет ветвь $\boldsymbol{H}_j$, которую «его» ресурс R_p сможет выполнить с минимальной задержкой.

Агент AR_p принимает ветвь $\boldsymbol{H}_j$ к исполнению, входит в состав сообщества $\boldsymbol{R}_l$ по выполнению задания Z_l и модифицирует граф задания $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ на ДО: исключает из него ветвь $\boldsymbol{H}_j$, в результате чего формируется новый граф задания $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1}) = \boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) / \boldsymbol{H}_j$, а каждой вершине модифицированного графа $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1})$, инцидентной вершине ветви $\boldsymbol{H}_j$, приписывает требуемый момент времени ее исполнения, определяемый согласно выражению (2.3), а также номер ресурса, которому должны быть переданы результаты исполнения приписанной ей операции.

После этого агент AR_p приступает к процедуре выполнения операций, приписанных вершинам ветви $\boldsymbol{H}_j$.

Исходя из этих соображений алгоритм работы агента AR_p ресурса R_p, входящего в состав гетерогенной СРС второго типа, можно представить в следующем виде. При этом, как и в предыдущих случаях, считаем, что при размещении Заказчиком задания Z_l на ДО счетчик итераций $j = 1$; $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) = \boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ и $\Delta T^l = 0$.

Алгоритм 2.4

1. Агент AR_p свободного ресурса R_p опрашивает ДО в порядке своей очереди (номера).

2. При обнаружении на ДО задания Z_i агент AR_p считывает его дескриптор графа задания $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$.

3. В графе $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ выделяется подграф $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp}, \boldsymbol{X}_l^{jp})$, вершинам которого приписаны операции множества $\boldsymbol{O}_p$, выполняемые ресурсом R_p.

4. $i = 1$; $\boldsymbol{G}_i(\boldsymbol{Q}_i, \boldsymbol{X}_i) = \boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp}, \boldsymbol{X}_l^{jp})$; $\Delta T_p = \infty$

5. Агент AR_p выделяет в графе $\boldsymbol{G}_i(\boldsymbol{Q}_i, \boldsymbol{X}_i)$ наиболее длинную ветвь $\boldsymbol{H}_i = < q_1^i, q_2^i, ..., q_k^i >$, согласно выражению (2.1) конечной вершины q_l^i, которой приписан требуемый момент времени исполнения T_{k+1}^i, и

определяет момент времени T_k^i, когда он должен приступить к ее выполнению как

$$T_1^i = T_{k+1}^i - \left(\sum_{m=1}^{k} t_p(O_m^i) + t_{\text{п}} \right), \text{ где } t_p(O_m^i) = \frac{v(O_m^i)}{D_p(O_m^i)}.$$

6. $\Delta T_p^i = \begin{cases} T_{\text{тек}} - T_1^i, \text{если } T_{\text{тек}} - T_1^i > 0. \\ 0, \text{если } T_{\text{тек}} - T_1^i \le 0. \end{cases}$

7. Если $\Delta T_p^i < \Delta T_p$, то $\Delta T_p = \Delta T_p^i$ и $\boldsymbol{H}_j = \boldsymbol{H}_i$.

8. Ветвь $\boldsymbol{H}_i$ исключается из графа $\boldsymbol{G}_i(\boldsymbol{Q}_i, \boldsymbol{X}_i)$, т. е. $\boldsymbol{G}_{i+1}(\boldsymbol{Q}_{i+1}, \boldsymbol{X}_{i+1}) = \boldsymbol{G}_i(\boldsymbol{Q}_i, \boldsymbol{X}_i) / \boldsymbol{H}_i$.

9. Если в графе $\boldsymbol{G}_{i+1}(\boldsymbol{Q}_{i+1}, \boldsymbol{X}_{i+1})$ остались вершины, которым приписано требуемое время исполнения, то $i = i + 1$ и перейти к 5, иначе

10. $\Delta T^l = \Delta T^l + \Delta T_p$.

11. Агент AR_p принимает на себя выполнение ветви $\boldsymbol{H}_j$, записывает в список участников сообщества $\boldsymbol{R}_l$ свой номер и модифицирует граф $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ задания Z_l на ДО:

– всем вершинам $q_d^j \in \boldsymbol{H}_j$ $(d = 1,2,...,s)$ приписывается номер ресурса R_p и момент времени T_d начала их исполнения

$$T_d^j = T_{k+1}^j - \left(\sum_{i=d}^{k} t_p(O_i^j) + t_{\text{п}} \right);$$

– если вершина q_{k+1}^j инцидента вершине q_b^{j-1} ветви $\boldsymbol{H}_{j-1}$, ранее закрепленной за ресурсом $\boldsymbol{R}_c$ то моменты времени начала исполнения, приписанные вершинам $q_b^{j-1}, q_{b+1}^{j-1}, \ldots, q_L^{j-1}$ ветви $\boldsymbol{H}_{j-1}$, увеличиваются на величину ΔT_p^i;

– вершины ветви $\boldsymbol{H}_j$ исключаются из графа $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$, в результате чего формируется новый граф $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1}) = \boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) / \boldsymbol{H}_j$;

– каждой вершине q_b^{j+1} модифицированного графа $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1})$, инцидентной вершине q_d^j ветви $\boldsymbol{H}_j$, требуемый момент времени ее исполнения, определяемый как

$$T_{b+1}^{j+1} = T_{k+1}^{j} - \left(\sum_{i=d}^{k} t_p(O_i^j) + t_{\text{п}} \right),$$

а также номер ресурса R_p, которому необходимо передать результаты исполнения операции, приписанной вершине q_b^{j+1}.

13. Счетчик итераций на ДО увеличивается на единицу, т. е. $j = j+1$. Модифицированный граф $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$, а также новое значение ΔT^l размещаются на ДО. Если $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) \neq \varnothing$, то процесс распределения операций задания Z_l заканчивается.

14. Ресурс R_p приступает к выполнению последовательности операций, приписанных вершинам ветви $\boldsymbol{H}_j$ согласно установленному временному графику (т. е. приписанным вершинам ветви $\boldsymbol{H}_j$ моментам времени начала их исполнения).

15. После завершения выполнения всех операций ветви $\boldsymbol{H}_j$ переход к 1.

2.5. Алгоритм работы агентов при выполнении операций выбранной ветви задания

Как показано выше, процедура выполнения функционального задания $Z_l \in \boldsymbol{Z}$ в СРС состоит из двух этапов – этапа распределения операций задания Z_l между ресурсами СРС и этапа выполнения операций задания Z_l ресурсами, вошедшими в состав сообщества $\boldsymbol{R}_l$.

В предыдущих разделах рассмотрены алгоритмы работы агентов ресурсов при реализации первого этапа-распределения операций задания Z_l между ресурсами СРС. В данном разделе рассмотрим работу агента AR_p ресурса R_p при реализации второго этапа, т. е. выполнении операций принятой им к исполнению ветви $\boldsymbol{H}_f = < q_1^f, q_2^f, ..., q_k^f >$ графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$.

После того, как некоторый агент AR_p вошел в состав сообщества $\boldsymbol{R}_l$ по выполнению задания Z_l и выбрал для исполнения ветвь $\boldsymbol{H}_f = < q_1^f, q_2^f, ..., q_k^f >$ графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, он приступает к ее исполнению, т. е. инициирует выполнение операций, приписанных ее вершинам, с помощью локального устройства управления $УУ_p$ «своего» ресурса R_p. При этом может оказаться, что для выполнения очередной операции O_d, приписанной вершине $q_d^f \in \boldsymbol{H}_f$ $(d = 1,2,...,k)$, могут требоваться результаты выполнения смежных ветвей графа задания, получаемые другими ресурсами СРС, входящими в состав сообщества $\boldsymbol{R}_l$, которые к моменту T_d^f начала выполнения операции O_d должны поступить на ресурс R_p (рис. 2.8) [21]. Поэтому, прежде чем приступить к выполнению очередной операции O_d, приписанной вершине $q_d^f \in \boldsymbol{H}_f$ $(d = 1,2,...,k)$, агент AR_p должен сначала проверить наличие результатов выполнения другими ресурсами сообщества $\boldsymbol{R}_l$ смежных ветвей графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, необходимых для выполнения операции O_d (рис. 2.8).

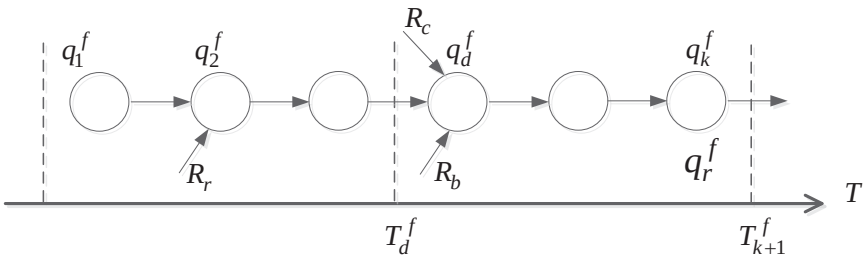


Рис. 2.8. Выполнение ресурсом R_p операции O_d, приписанной вершине q_d^f

Если необходимые для выполнения операции O_d результаты выполнения смежных ветвей графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ еще не готовы и не получены ресурсом R_p, то агент AR_p должен перейти в режим ожидания поступления этих результатов. После того, как ресурс R_p получит все необходимые результаты выполнения смежных ветвей графа задания, агент AR_p инициирует выполнение операции O_d с помощью локального устройства управления ресурса R_p. Аналогичным образом агент AR_p должен обеспечить выполнение всех операций O_d, приписанных вершинам ветви $\boldsymbol{H}_f =< q_1^f, q_2^f, ..., q_k^f >$.

После выполнения всех операций ветви $\boldsymbol{H}_f =< q_1^f, q_2^f, ..., q_k^f >$ агент AR_p выходит из состава сообщества $\boldsymbol{R}_l$ по выполнению задания Z_l (т. е. его номер исключается из множества $\boldsymbol{R}_l$ в дескрипторе задания Z_l на ДО), а результат выполнения операции, приписанной последней вершине q_k^f ветви $\boldsymbol{H}_f =< q_1^f, q_2^f, ..., q_k^f >$ он передает агенту ресурса R_c, номер которого приписан вершине q_k^f, или Заказчику, если последняя вершина q_k^f выполненной ветви $\boldsymbol{H}_f =< q_1^f, q_2^f, ..., q_k^f >$ – это конечная вершина графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, т.е. $q_k^f = q_k$.

Кроме того, агент AR_p должен проверить соблюдение временного графика выполнения всей ветви $\boldsymbol{H}_f =< q_1^f, q_2^f, ..., q_k^f >$. Для этого он сравнивает запланированный ранее момент времени T_{k+1}^f завершения исполнения ветви $\boldsymbol{H}_f =< q_1^f, q_2^f, ..., q_k^f >$, где T_{k+1}^f – момент времени, приписанный конечной вершине q_k^f ветви $\boldsymbol{H}_f =< q_1^f, q_2^f, ..., q_k^f >$, с текущим моментом времени $T_{\text{тек}}$. Если $T_{\text{тек}} > T_{k+1}^f$, то это означает, что произошла задержка в выполнении временного графика решения задания Z_l относительно ранее запланированного, о чем агент AR_p информирует всех остальных агентов, входящих в состав сообщества $\boldsymbol{R}_l$ по выполнению задания Z_l. Последние на основании этой информации должны скорректировать временной график исполнения всех оставшихся вершин графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ путем увеличения планируемых моментов времени их исполнения на величину $\Delta T_p^f = T_{\text{тек}} - T_{k+1}^f$.

Описанной выше процедуре отвечает следующий алгоритм работы агента ресурса R_p.

Алгоритм 2.7

1. После вхождения в состав сообщества $\boldsymbol{R}_l$ и завершения процедуры распределения операций задания Z_l агент AR_p приступает к реализации последовательности операций, приписанных вершинам принятой им к исполнению ветви $\boldsymbol{H}_f =< q_1^f, q_2^f, ..., q_k^f >;\ d = 1.$

2. Агент AR_p проверяет наличие всех промежуточных результатов, необходимых для выполнения операции O_d, приписанной вершине $q_d^f \in \boldsymbol{H}_f$, которые должны получить от ресурсов, выполняющих смежные ветви графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$. Если результаты этих операций еще не поступили на ресурс R_p, то перейти к 2, иначе

3. Агент AR_p инициирует выполнение операции O_d, приписанной вершине $q_d^f \in \boldsymbol{H}_f$, с помощью локального УУ$_p$ «своего» ресурса R_p.

4. $d = d + 1$, если $d \le k$, то переход к 2, иначе

5. После выполнения всех операций ветви $\boldsymbol{H}_f =< q_1^f, q_2^f, ..., q_k^f >$ агент AR_p выходит из состава сообщества $\boldsymbol{R}_l$ по выполнению задания Z_l, а результат выполнения последней операции ветви $\boldsymbol{H}_f =< q_1^f, q_2^f, ..., q_k^f >$ передается агенту ресурса R_c, номер которого приписан последней вершине q_k^f ветви $\boldsymbol{H}_f$, или Заказчику, если последняя вершина выполненной ветви $\boldsymbol{H}_f$ – это конечная вершина графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, т.е. $q_k^f = q_k$.

6. Агент AR_p сравнивает запланированный ранее момент времени T_{k+1}^f завершения исполнения ветви $\boldsymbol{H}_f =< q_1^f, q_2^f, ..., q_k^f >$, приписанный ее конечной вершине q_k^f, с текущим моментом времени $T_{\text{тек}}$. Если $T_{\text{тек}} > T_{k+1}$, то агент AR_p информирует всех агентов, входящих в состав сообщества $\boldsymbol{R}_l$ по выполнению задания Z_l, о возникшей задержке, на основании чего они корректируют временной график исполнения всех оставшихся вершин графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ путем увеличения планируемых моментов времени их исполнения на величину $\Delta T_p^f = T_{\text{тек}} - T_{k+1}^f$.

2.6. Сравнительная оценка эффективности метода мультиагентного диспетчирования ресурсов СРС

В предыдущих подразделах предложен метод и алгоритмы мультиагентного диспетчирования ресурсов в СРС различных типов. При этом возникает вопрос, насколько данные алгоритмы эффективны.

Как показано в разделе 1.1, в качестве критерия эффективности работы диспетчера СРС целесообразно принять среднюю задержку ΔT выполнения заданий потока $\boldsymbol{Z} =< Z_1, Z_2, ..., Z_L >$, т. е. величину

$$\Delta T = \frac{\sum_{i=1}^{L} \Delta T^l}{L} = \frac{\sum_{i=1}^{L} (T_p^l - T_{max}^l)}{L},$$

где ΔT^l – задержка выполнения задания $Z_l \in \boldsymbol{Z}$;

L – число заданий в потоке $\boldsymbol{Z}$;

T_p^l – момент времени завершения задания $Z_l \in \boldsymbol{Z}$;

T_{max}^l – требуемый момент выполнения задания Z_l.

Учитывая, что

$$T_p^l = T_{\Pi}^l + t^l,$$

где T_{Π}^l – момент времени размещения задания $Z_l \in \boldsymbol{Z}$ на ДО,

t^l – период времени выполнения задания Z_l,

получаем

$$\Delta T = \frac{\sum_{i=1}^{L} (T_{\Pi}^l + t^l - T_{max}^l)}{L}.$$

Поскольку значения T_{Π}^l и T_{max}^l известны, то из последнего выражения следует, что для минимизации величины ΔT средней задержки выполнения задания потока $\boldsymbol{Z}$ необходимо минимизировать величину t^l $(l = 1,2,...,L)$ времени выполнения задания $Z_l \in \boldsymbol{Z}$.

В свою очередь, в подразделе 1.2 показано, что

$$t^l = t_p^l + t_B^l,$$

где t_p^l – период времени, затрачиваемый диспетчером на выполнение процедуры распределения операций задания Z_l между ресурсами сообщества $\boldsymbol{R}_l \subseteq \boldsymbol{R}$;

t_B^l – период времени, затрачиваемый ресурсами сообщества $\boldsymbol{R}_l$, непосредственно на выполнение операций задания Z_l.

В общем случае значение времени t_p^l можно оценить с помощью следующего выражения:

$$t_p^l = C \cdot t_a,$$

где C – число переборов различных вариантов распределения ресурсов СРС между ветвями графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$;

t_a – время, затрачиваемое на реализацию алгоритма анализа очередного варианта распределения.

В свою очередь значение времени t_B^l определяется временем исполнения наиболее трудоемкой ветви $\boldsymbol{H}_1 =< q_1^1, q_2^1, \ldots, q_k^1 >$ графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, т. е.

$$t_B^l = \sum_{i=1}^{k} t_p(O_i^1) = \sum_{i=1}^{k} \frac{v(O_i^1)}{D_p(O_i^1)},$$

где $t_p(O_i^1)$ (i = 1, 2, …, k) – время выполнения операции O_i^1, приписанной вершине $q_i^1 \in \boldsymbol{H}_1$, с помощью ресурса R_p;

$v(O_i^1)$ – трудоемкость операции O_i^1;

$Dp(O_i^1)$ – производительность ресурса R_p при выполнении операции O_i^1.

Таким образом, в общем случае, время t^l можно оценить как:

$$t^l = t_p^l + t_B^l = C \cdot t_a + \sum_{i=1}^{k} \frac{v(O_i^1)}{D_p(O_i^1)}.$$

(2.4)

С помощью последнего выражения сравним значения времени t^l, получаемые при использовании предложенных выше метода децентрализованного диспетчирования ресурсов СРС на основе мультиагентных алгоритмов и метода централизованного диспетчирования ресурсов СРС на основе алгоритма полного перебора.

При использовании метода децентрализованного диспетчирования агентами ресурсов R_i (i = 1, 2, …, N) осуществляется последовательный выбор ветвей графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, начиная с самой трудоемкой. При этом общее число переборов C вариантов распределения ветвей графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ между ресурсами R_i (i = 1, 2, …, N) при использовании данного метода можно оценить величиной

$$C \leq N \cdot L,$$

где N – число ресурсов, входящих в состав СРС;

L – число ветвей графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$.

Выбор наиболее трудоемкой ветви $\boldsymbol{H}_1$ графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ для исполнения в данном случае осуществляется первым из ресурсов, обнаруживших задание Z_l на ДО. При этом может оказаться, что производительность данного ресурса R_p при выполнении операций, приписанных вершинам ветви $\boldsymbol{H}_1$, не самая высокая среди всех ресурсов множества $\boldsymbol{R}$.

Поэтому значение величины t_B^l в данном случае можно оценить следующим образом:

$$t_B^l = \sum_{i=1}^{k} \frac{v(O_i^1)}{D_p(O_i^1)} \leq \sum_{i=1}^{k} \frac{v(O_i^1)}{D_{min}(O_i^1)},$$

где $D_{min}(O_i^1)$ – минимальная производительность среди ресурсов множества $\boldsymbol{R}$ при выполнении операции O_i^1 (i = 1, 2, …, k).

Исходя из этих рассуждений общее время t^l выполнения задания Z_l при использовании метода мультиагентного диспетчирования можно оценить как

$$t^l \le N \cdot L \cdot t_a + \sum_{i=1}^{k} \frac{v(O_i^1)}{D_{min}(O_i^1)} \tag{2.5}$$

При использовании метода централизованного диспетчирования на основе алгоритма полного перебора число C анализируемых вариантов распределения ресурсов СРС по ветвям графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ можно оценить с помощью формулы числа размещений с повторениями как

$$C = N^L,$$

где N – число ресурсов СРС; L – число ветвей графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$,

При этом алгоритм полного перебора гарантирует, что наиболее трудоемкая ветвь $\boldsymbol{H}_1 =< q_1^1, q_2^1, \dots, q_k^1 >$ будет закреплена за тем ресурсом R_p, который обеспечивает максимальную производительность при выполнении операций, приписанных ее вершинам. Поэтому значение t_B^l в данном случае можно оценить следующим образом:

$$t_B^l = \sum_{i=1}^{k} \frac{v(O_i^1)}{D_p(O_i^1)} \ge \sum_{i=1}^{k} \frac{v(O_i^1)}{D_{max}(O_i^1)},$$

где $D_{max}(O_i^1)$ $\left(i = 1, 2, \dots, k\right)$ – максимальная производительность среди ресурсов множества $\boldsymbol{R}$ при выполнении операции O_i^1.

Таким образом, время t^l выполнения задания Z_l при использовании метода централизованного диспетчирования и алгоритма полного перебора вариантов распределения можно оценить следующим выражением

$$t^l \ge N^L + \sum_{i=1}^{k} \frac{v(O_i^1)}{D_{max}(O_i^1)}, \tag{2.6}$$

Сравним выражения (2.5) и (2.6). Если

$$N^L + \sum_{i=1}^{k} \frac{v(O_i^1)}{D_{max}(O_i^1)} > N \cdot L \cdot t_a - \sum_{i=1}^{k} \frac{v(O_i^1)}{D_{min}(O_i^1)}$$

или, иначе,

$$(N^L - N \cdot L) \cdot t_a > \sum_{i=1}^{k} v(O_i^1) \cdot \frac{\left(D_{min}(O_i^1) - D_{max}(O_i^1)\right)}{D_{min}(O_i^1) \cdot D_{max}(O_i^1)}, \tag{2.7}$$

то это означает, что использование метода децентрализованного диспетчирования ресурсов СРС на основе мультиагентных алгоритмов при данных значениях N, L, $D_{min}(O_i^1)$ и $D_{max}(O_i^1)$ эффективнее, чем метода централизованного диспетчирования на основе алгоритма полного перебора. В противном случае, наоборот, метод централизованного диспетчирования оказывается эффективнее.

Заметим, что в случае, когда СРС относится к классу гомогенных или гетерогенных СРС первого типа (см. подраздел 1.1) и все ресурсы

имеют одинаковую производительность при выполнении одних и тех же операций, то $D_{max}(O_i^1) = D_{min}(O_i^1)$. При этом выражение (2.7) можно существенно упростить, и условие эффективности метода децентрализованного мультиагентного диспетчирования по сравнению с методом централизованного диспетчирования на основе алгоритма полного перебора принимает следующий вид

$$N^L - N \cdot L > 0.$$

Глава 3. Алгоритмы работы агентов ресурсов СРС при наличии стимулирующих факторов

3.1. Алгоритм работы агента ресурса СРС в случае премирования за выполненную работу

В предыдущем разделе при разработке методов и алгоритмов поведения агентов ресурсов R_i $(i=\overline{1,N})$ СРС считалось, что все они в равной степени заинтересованы в выполнении всех функциональных заданий $Z_l \in \boldsymbol{Z}$, размещаемых на ДО, с минимальной задержкой относительно требуемых моментов времени. При этом полагалось, что никаких дополнительных стимулов либо «личных» интересов ресурсы R_i $(i=\overline{1,N})$ не имеют. В настоящем разделе будут разработаны методы мультиагентного диспетчирования ресурсов R_i $(i=\overline{1,N})$, входящих в состав СРС, при выполнении потока заданий $Z_l \in \boldsymbol{Z}$ в случае наличия у них различных стимулирующих факторов.

Допустим, что за успешное выполнение задания Z_l к требуемому моменту времени $T^l_{\max}$ установлена некоторая премия S^l (премиальные баллы) [22], причем величина этой премии будет снижаться в случае задержки выполнения задания относительно требуемого момента времени $T^l_{\max}$ следующим образом:

$$S^l = \begin{cases} S^l_{max} \cdot \frac{\Delta T^l_{max} - \Delta T^l}{\Delta T^l_{max}}, \text{если } \Delta T^l \le \Delta T^l_{max,} \\ 0, \text{если } \Delta T^l > \Delta T^l_{max} \end{cases} \quad (3.1)$$

где S^l – величина премии, получаемой за выполнение задания Z_l;

S^l_{max} – максимальная величина премии, получаемая при условии, что задание Z_l выполнено к моменту $T^l_{\max}$;

ΔT^l – задержка выполнения задания Z_l относительно момента $T^l_{\max}$;

ΔT^l_{max} – максимально допустимая задержка выполнения задания Z_l относительно момента $T^l_{\max}$.

Кроме того, будем считать, что в случае успешного выполнения задания Z_l каждому ресурсу R_p, входящему в сообщество $\boldsymbol{R}_l$, начисляются премиальные баллы исходя из его «вклада» в выполнение задания Z_l. Этот «вклад» можно оценить отношением трудоемкости операций задания Z_l, выполненных ресурсом R_p, к общей суммарной трудоемкости всех операций задания Z_l. Тогда премиальные баллы, начисляемые ресурсу R_p за участие в выполнении задания Z_l, будут определяться следующим образом:

$$S_p^l = \frac{S^l \cdot \sum_{i=1}^{K} v(O_i^p)}{\sum_{i=1}^{M} v(O_i)}, \qquad (3.2)$$

где S_p^l – количество премиальных баллов, начисляемых ресурсу R_p;

$\sum_{i=1}^{M} v(O_i)$ – суммарная трудоемкость всех операций O_i задания Z_l;

$\sum_{i=1}^{K} v(O_i^p)$ – суммарная трудоемкость операций O_i^p задания Z_l, выполненных ресурсом R_p;

$M = |\boldsymbol{Q_l}|$ – число вершин (операций) в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l;

K – число операций задания Z_l, выполненных ресурсом R_p.

В этом случае, каждый ресурс $R_p \in \boldsymbol{R}_l$, кроме «общественной» цели – выполнения функциональных заданий $Z_l \in \boldsymbol{Z}$ с минимальной задержкой относительно требуемых моментов времени, будет иметь еще и «личный» интерес – получить как можно больше премиальных баллов за свою работу.

Рассмотрим принципы поведения агента AR_p, представляющего «интересы» ресурса $R_p \subseteq \boldsymbol{R}_l$, при наличии такого стимулирующего фактора.

Допустим, что агент R_p обращается к ДО и обнаруживает там дескрипторы нескольких заданий $Z_1, Z_2, ..., Z_L$. Первый вопрос, который должен решить агент AR_p: в сообщество по выполнению какого задания ему следует вступить, чтобы получить максимальное число премиальных баллов? По-видимому, целесообразнее всего ему выбрать такое задание $Z_l \in \boldsymbol{Z}$, у которого среднее число премиальных баллов, приходящихся на единицу трудоемкости операций, максимально, т. е. максимальна величина

$$S_{\text{ср}}^l = \frac{S_{max}^l}{\sum_{i=1}^{M} v(O_i)},$$

которую в дальнейшем для краткости будем называть средней стоимостью единицы трудоемкости задания Z_l [23]. Очевидно, что чем больше значение $S_{\text{ср}}^l$ задания Z_l, тем больше премиальных баллов потенциально может «заработать» агент AR_p, участвующий в его выполнении.

После того, как агент AR_p выбрал задание Z_l, имеющее максимальное значение средней стоимости единицы трудоемкости среди всех заданий, размещенных на ДО, он должен выделить фрагмент (ветвь) этого задания, выполнение которого позволит ему получить максимальное количество премиальных баллов. Поскольку величина премиальных баллов, получаемых

агентом AR_p, определяется выражениями (3.1) и (3.2), то, соответственно, он должен выбрать для исполнения такую ветвь $\boldsymbol{H_f}$ задания Z_l, трудоемкость которой была бы как можно большей, но при этом возникающая при ее исполнении задержка была как можно меньшей. Иными словами, агент AR_p должен выбрать для исполнения такую ветвь $\boldsymbol{H_f} =< q_1^f, q_2^f, \ldots, q_k^f >$ задания Z_l, для которой выполняются условия (рис. 3.1).

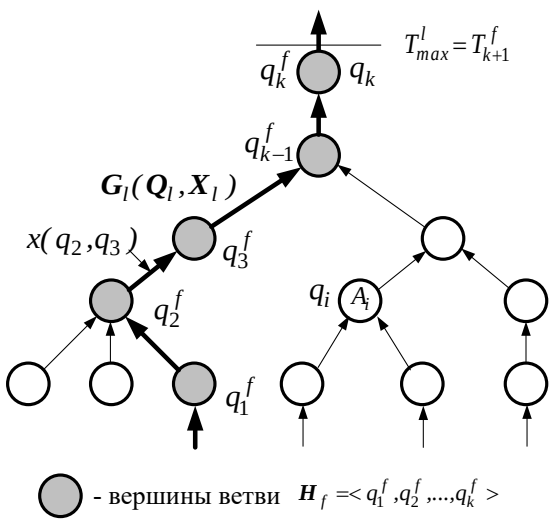


Рис. 3.1. Выделение ветви $\boldsymbol{H}_1$ в графе $\boldsymbol{G_l(Q_l, X_l)}$ задания Z_l

– Во-первых, установлен момент времени ее исполнения T_{k+1}^f (в момент размещения Заказчиком задания Z_l на ДО такое время установлено только для конечной вершины графа $\boldsymbol{G_l(Q_l, X_l)}$, т. е. в этом случае $T_{k+1}^f = T_{max}^l$);

– Во-вторых, величина

$$S_p^{lf} = \begin{cases} \dfrac{\sum_{i=1}^{k} v(O_i^f)}{\sum_{j=1}^{M} v(O_j)} \cdot S_{max}^l \cdot \dfrac{\Delta T^l - \Delta T_p^{lf}}{\Delta T^l}, \text{если } \Delta T_p^{lf} < \Delta T^l \\ 0, \text{если } \Delta T_p^{lf} \geq \Delta T^l \end{cases} \quad (3.3)$$

максимальна, где

S_p^{lf} – количество премиальных баллов, получаемых агентом AR_p за выполнение операций ветви $\boldsymbol{H_f}$ задания Z_l;

ΔT_p^{lf} – задержка выполнения ветви $\boldsymbol{H_f}$ задания Z_l ресурсом R_p;

$$\Delta T^l = \Delta T^l_{max} - \sum_{i=1}^{p-1} \Delta T^l_i$$

$j-1$ – число агентов, уже вступивших ранее в сообщество $\boldsymbol{R_l}$;

ΔT^l_i – задержка, вносимая в выполнение задания Z_l ресурсом R_i (i = 1, 2, …, $p-1$), ранее вступившим в сообщество $\boldsymbol{R_l}$;

$\sum_{i=1}^{k} v(O^f_i)$ – суммарная трудоемкость операций ветви $\boldsymbol{H_f}$.

Учитывая, что величину задержки ΔT^{lf}_p можно определить с помощью следующего выражения (см. выражение (2.2))

$$\Delta T^{lf}_p = \begin{cases} T_{\text{тек}} - T^f_1, \text{если } T_{\text{тек}} > T^f_1 \\ 0, \text{если } T_{\text{тек}} \le T^f_1 \end{cases},$$

где $T^f_1 = T^f_{k+1} - \left(\sum_{i=1}^{k} t_p(O^f_i) + t_{\text{п}}\right)$ – момент времени начала выполнения первой операции ветви $\boldsymbol{H_f}$ ресурсом R_p, при котором последний успевает завершить ее исполнение к требуемому моменту T^f_{k+1};

$t_p(O^f_i)$ – время выполнения ресурсом R_p операции O^f_i (i = 1, 2, …, k), приписанной вершине q^f_j ветви $\boldsymbol{H_f}$;

$t_{\text{п}}$ – время пересылки результата выполнения ветви $\boldsymbol{H_f}$ другому ресурсу, выполняющему смежную ветвь задания Z_l.

Подставляя последнее выражение в (3.3), получаем

$$S^{lf}_p = \begin{cases} \frac{\sum_{i=1}^{k} v(o^f_i) \cdot S^l_{max} \cdot \left(\Delta T^l - T_{\text{тек}} + T^f_{k+1} - \sum_{j=1}^{k} t_p(o^f_j) - t_{\text{П}}\right)}{\sum_{j=1}^{M} v(o_j) \cdot \Delta T^l}, \text{если } \Delta T^{lf}_p < \Delta T^l \\ 0, \text{если } \Delta T^{lf}_p \ge \Delta T^l. \end{cases} \quad (3.4)$$

Таким образом, агент AR_p должен выбрать для исполнения такую ветвь $\boldsymbol{H_f}$ задания Z_l, для которой величина (3.4) максимальна.

После того, как агент AR_p выделил некоторую ветвь $\boldsymbol{H}_f = < q^f_1, q^f_2, ..., q^f_k >$, удовлетворяющую перечисленным выше условиям, и принял ее к исполнению (см. рис.3.1), как и в ранее рассмотренных алгоритмах, он производит модификацию графа $\boldsymbol{G_l(Q_l, X_l)}$ задания Z_l на ДО, исключая из него вершины ветвь $\boldsymbol{H_f}$, в результате чего формируется новый граф $\boldsymbol{G^1_l(Q^1_l, X^1_l) = G_l(Q_l, X_l) / H_f}$, а также приписывает всем вершинам вновь сформированного графа $\boldsymbol{G^1_l(Q^1_l, X^1_l)}$, инцидентным вершинам нити $\boldsymbol{H_f}$, требуемые моменты времени их исполнения, определяемые согласно выражению (2.3), а также номер ресурса R_p, которому должны быть переданы результаты исполнения приписанной им операции.

Исходя из приведенных выше соображений, алгоритм работы агента AR_p ресурса R_p при наличии стимулирующей премии за выполнение задания $Z_l \in \boldsymbol{Z}$ можно представить в следующем виде. При этом полагается, что в дескрипторе задания Z_l на ДО хранится значение величины S^l премиальных баллов, назначаемых за успешное выполнение задания Z_l к требуемому моменту времени $T^l_{\max}$, а также величины задержки выполнения задания Z_l, вносимой ресурсами R_i (i = 1, 2, …, p – 1), ранее вступившим в сообщество $\boldsymbol{R_l}$.

Алгоритм 3.1

1. Агент AR_p свободного ресурса R_p считывает в порядке своей очереди дескрипторы заданий $\boldsymbol{Z} =< Z_1, Z_2, ..., Z_L >$, размещенных на ДО, а также количество премиальных баллов S^l ($l = 1,2,...,L$), назначенных за их успешное выполнение к требуемому моменту времени $T^l_{\max}$.

2. Если $\boldsymbol{Z} = \varnothing$, то перейти к 1, иначе

3. Агент AR_p среди множества заданий $\boldsymbol{Z}$ выбирает то задание Z_l, для которого значение $S^l_{\text{ср}} = \frac{S^l_{max}}{\sum_{i=1}^{M} v(O_i)}$ максимально.

4. Агент AR_p выделяет в графе $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ задания Z_l ветвь $\boldsymbol{H_f} =< q_1^f, q_2^f, \dots, q_n^f >$, для которой выполняются условия:

– конечной вершине q_k^j приписано требуемое время ее исполнения T^f_{k+1};

– значение величины

$$S_p^{lf} = \begin{cases} \dfrac{\sum_{i=1}^{k} v(O_i^p) \cdot S^l_{max} \cdot \left(\Delta T^l - T_{\text{тек}} + T^f_{k+1} - \sum_{j=1}^{k} t_p(O_j^f) - t_{\text{П}} \right)}{\sum_{j=1}^{M} v(O_j) \cdot \Delta T^l}, \text{если } \Delta T_p^{lf} < \Delta T^l \\ 0, \text{если } \Delta T_p^{lf} \geq \Delta T^l. \end{cases}$$

максимально,

где $\Delta T^l = \Delta T^l_{max} - \sum_{i=1}^{p-1} \Delta T_i^l$.

Если таковой нити нет, то $\boldsymbol{Z} = \boldsymbol{Z} / Z_l$, перейти к 2, иначе

5. Агент AR_p принимает на себя выполнение нити $\boldsymbol{H}_f$, после чего модифицирует дескриптор задания Z_l на ДО:

– записывает в список участников сообщества $\boldsymbol{R}_l$ свой номер;

– модифицирует граф $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ задания Z_l путем исключения из него вершин нити $\boldsymbol{H}_f$, т.е. $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1}) = \boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) / \boldsymbol{H}_f$;

– каждой вершине q_b^{j+1} модифицированного графа $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1},\boldsymbol{X}_l^{j+1})$, инцидентной вершине q_d^f нити $\boldsymbol{H}_f$, приписывает требуемый момент времени исполнения, определяемое как

$$T_{b+1}^{j+1} = T_{k+1}^i - \left(\sum_{m=d}^{k} t_p(O_i^f) + t_{\text{п}}\right),$$

а также номер ресурса R_p, которому необходимо передать результаты исполнения операции, приписанной вершине q_b^{j+1};

– определяет величину задержки исполнения операций ветви $\boldsymbol{H}_f$

$$\Delta T_p^l = \begin{cases} T_{\text{тек}} - T_1^f, \text{если } T_{\text{тек}} > T_1^f \\ 0, \text{если } T_{\text{тек}} \le T_1^f \end{cases}$$

где $T_1^f = T_{k+1}^f - \left(\sum_{i=1}^{k} t_p(O_i^f) + t_{\text{п}}\right)$.

6. $i = 1$.

7. Агент AR_p приступает к исполнению последовательности операций, приписанных вершинам ветви $\boldsymbol{H}_f$; $d = 1$.

8. Агент AR_p проверяет наличие всех результатов, необходимых для выполнения операции O_d^f, приписанной вершине $q_d^f \in \boldsymbol{H}_f$, и поступающих от ресурсов, выполняющих смежных с ней вершин графа задания. Если результаты этих операций еще не поступили на ресурс R_p, то перейти к 8, иначе

9. Агент AR_p инициирует выполнение операции O_d^f, приписанной вершине $q_d^f \in \boldsymbol{H}_f$, с помощью локального УУ$_P$ «своего» ресурса R_p.

10. $d = d + 1$, если $d \le k$, то переход к 8, иначе

11. Агент AR_p выходит из сообщества $\boldsymbol{R}_l$. Результат выполнения последней операции ветви $\boldsymbol{H}_f$ передается ресурсу R_c, номер которой приписан конечной вершине q_k^f ветви $\boldsymbol{H}_f$, или на ДО, если последняя вершина выполненной ветви $\boldsymbol{H}_f$ – это конечная вершина графа $\boldsymbol{G}_l(\boldsymbol{Q}_l,\boldsymbol{X}_l)$, т.е. $q_k^f = q_k$.

13. По завершению задания Z_l агенту AR_p начисляются премиальные баллы

$$S_p^l = \begin{cases} \dfrac{S^l \cdot V_l^p}{V_l} = \dfrac{S_{max}^l \cdot V_l^p \cdot (T_{max}^l - \Delta T^l)}{V_l \cdot \Delta T_{max}^l}, \text{если } \Delta T^l < \Delta T_{max}^l, \\ 0, \text{если } \Delta T^l \ge \Delta T_{max}^l. \end{cases}$$

где $V_l^p = \sum_{j=1}^{K} v(O_j^f)$ – суммарная трудоемкость операций ветви $\boldsymbol{H_f}$ задания Z_l, выполненных ресурсом R_p;

$V_l = \sum_{j=1}^{M} v(O_j)$ – суммарная трудоемкость всех операций задания Z_l.

S_{max}^l – максимальное число премиальных баллов, начисляемых в случае выполнения задания Z_l к требуемому моменту T_{max}^l;

ΔT_{max}^l – максимальная допустимая задержка выполнения задания Z_l.

ΔT^l – задержка выполнения задания Z_l относительно требуемого момента времени T_{max}^l.

3.2. Алгоритм поведения агентов ресурсов СРС в условиях штрафных санкций

В предыдущем разделе был предложен подход, стимулирующий агентов ресурсов СРС к выполнению задания Z_l к требуемому моменту времени, за счет введения премиальных баллов. При этом количество премиальных баллов, получаемых агентом AR_p ресурса R_p, прямо пропорционально зависит от трудоемкости выполненных им операций задания Z_l и обратно пропорционально – от общей задержки ΔT^l выполнения задания Z_l, причем личный «вклад» данного ресурса в эту задержку никоим образом не учитывается.

В результате агенты, обнаружившие задания Z_l на ДО, будут стараться «заработать» как можно больше премиальных баллов, мало «задумываясь» о том, какую «личную» задержку в выполнение задания Z_l они при этом вносят. Это, в свою очередь, может приводить к увеличению общей задержки ΔT^l выполнения задания Z_l в целом.

Данного недостатка можно избежать, если ввести систему индивидуальной ответственности агентов за задержку, вносимую их ресурсом в общую задержку ΔT^l выполнения задания Z_l. Последнее можно осуществить путем наложения некоторых «индивидуальных» штрафов на тех агентов сообщества $\boldsymbol{R}_l$, которые допустили возникновение задержки при выполнении задания Z_l.

Как и в предыдущем случае, будем считать, что общая премия, получаемая сообществом $\boldsymbol{R}_l$ за выполнение задания Z_l, составляет

$$S^l = \begin{cases} S^l_{max} \cdot \frac{\Delta T^l_{max} - \Delta T^l}{\Delta T^l_{max}}, & \text{если } \Delta T^l < \Delta T^l_{max} \\ 0, & \text{если } \Delta T^l \geq \Delta T^l_{max} \end{cases}$$

где ΔT^l_{max} – максимально допустимая задержка выполнения задания Z_l;

S^l_{max} – максимальная премия, получаемая сообществом $\boldsymbol{R}_l$ при условии выполнения задания Z_l к моменту $T^l_{\max}$.

При этом величина

$$\Delta S^l = S^l_{max} - S^l = S^l_{max} - S^l_{max} \cdot \frac{\Delta T^l_{max} - \Delta T^l}{\Delta T^l_{max}} = S^l_{max} \cdot \frac{\Delta T^l}{\Delta T^l_{max}} \qquad (3.5)$$

будет определять общую величину штрафа, начисляемого на сообщество $\boldsymbol{R}_l$ за задержку выполнения задания Z_l на время ΔT^l относительно требуемого момента времени $T^l_{\max}$, причем

$$\Delta T^l \leq \Delta T^l_1 + \Delta T^l_2 + \ ... \ + \Delta T^l_n, \qquad (3.6)$$

где ΔT^l_i (i = 1, 2, …, n) – задержка, вносимая i-м агентом сообщества $\boldsymbol{R}_l$;

$n = |\boldsymbol{R}_l|$.

Подставляя (3.6) в (3.5), получаем

$$\Delta S^l \le S^l_{max}\frac{\Delta T^l_1+\Delta T^l_2+\ldots+\Delta T^l_n}{\Delta T^l_{max}} = S^l_{max}\cdot\frac{\Delta T^l_1}{\Delta T^l_{max}} + S^l_{max}\cdot\frac{\Delta T^l_2}{\Delta T^l_{max}} + \cdots + S^l_{max}\cdot\frac{\Delta T^l_n}{\Delta T^l_{max}},$$

причем величина $\Delta S^l_p = S^l_{max}\cdot\frac{\Delta T^l_i}{\Delta T^l_{max}}$ определяет штраф, начисляемый на *i*-го агента AR_p сообщества $\boldsymbol{R}_l$ (p = 1, 2, …, n).

Учитывая, что

$$\Delta T^l_p = \begin{cases} T_{\text{тек}} - T^f_1, \text{если } T_{\text{тек}} > T^f_1 \\ 0, \qquad \text{если } T_{\text{тек}} \le T^f_1 \end{cases}, (p = 1, 2, \ldots, n),$$

где $T^f_1 = T^f_{k+1} - \left(\sum_{j=1}^{k} t_p(O^f_j) + t_{\text{п}}\right)$, получаем

$$\Delta S^l_p = S^l_{max}\frac{T_{\text{тек}} - T^f_{k+1} + \sum_{j=1}^{k} t_p(O^f_i) + t_{\text{п}}}{\Delta T^l_{max}}. \qquad (3.7)$$

Последнее выражение определяет величину «индивидуального» штрафа, который должен накладываться на агента AR_p (p = 1, 2, …, n) сообщества $\boldsymbol{R}_l$ за задержку выполнения задания Z_l относительно установленного Заказчиком момента времени $T^l_{\max}$.

Соответственно, премия, получаемая агентом AR_p, будет составлять

$$S^l_p = \begin{cases} S^l_{max}\cdot\frac{V^p_l}{V_l} - \Delta S^l_p, \text{если } \Delta T^l < \Delta T^l_{max}; \\ 0, \qquad\qquad \text{если } \Delta T^l \ge \Delta T^l_{max}, \end{cases} \qquad (3.8)$$

где $V^p_l = \sum_{j=1}^{K} v(O^i_j)$ – суммарная трудоемкость операций задания Z_l, выполненных ресурсом R_p;

$V_l = \sum_{j=1}^{M} v(O_i)$ – суммарная трудоемкость всех операций, приписанных вершинам графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l;

ΔT^l – общая задержка выполнения задания Z_l относительно требуемого момента времени $T^l_{\max}$.

Исходя из представленных соображений, можно предложить следующий алгоритм поведения агента AR_p ресурса R_p при распределении операций заданий потока $\boldsymbol{Z}$ при наличии «индивидуальных» штрафов.

Алгоритм 3.2

1. Агент AR_p свободного ресурса R_p считывает в порядке своей очереди дескрипторы заданий $\boldsymbol{Z} =< Z_1, Z_2, \ldots, Z_L >$, размещенных на ДО, а также количество премиальных баллов S^l (l =1,2,...,L), назначенных за их успешное выполнение к требуемому моменту времени $T^l_{\max}$.

2. Если $\boldsymbol{Z} = \varnothing$, то перейти к 1, иначе

3. Агент AR_p среди множества заданий $\boldsymbol{Z}$ выбирает то задание Z_l, для которого значение $S^l_{\text{ср}} = \frac{S^l_{max}}{\sum_{i=1}^{M} v(O_i)}$ максимально.

4. Агент AR_p выделяет в графе $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ задания Z_l ветвь $\boldsymbol{H_f} = < q_1^f, q_2^f, \ldots, q_k^f >$, для которой выполняются условия:

– конечной вершине q_k^j приписано требуемое время ее исполнения T_{k+1}^f;

– значение величины

$$S_p^l = S_{max}^l \cdot \frac{V_l^f}{V_l} - \Delta S_p^f$$

максимально,

где $V_l^f = \sum_{i=1}^{k} v(O_i^f)$ – трудоемкость операций ветви $\boldsymbol{H_f}$;

$V_l = \sum_{i=1}^{M} v(O_i)$ – трудоемкость всех операций графа $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ задания Z_l;

$$\Delta S_p^f = S_{max}^l \cdot \frac{T_{\text{тек}} - T_{k+1}^f + \sum_{j=1}^{k} t_p(O_i^f) + t_{\text{п}}}{\Delta T_{max}^l}$$

Если таковой нити нет, то $\boldsymbol{Z} = \boldsymbol{Z}/Z_l$, перейти к 2, иначе

5. Агент AR_p принимает на себя выполнение нити $\boldsymbol{H_f}$, для чего модифицирует дескриптор задания Z_l на ДО:

– записывает в список участников сообщества свой номер;

– модифицирует граф $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ задания Z_l путем исключения из него вершин нити $\boldsymbol{H_f}$, т.е. $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1}) = \boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)/\boldsymbol{H_f}$;

– каждой вершине q_b^{j+1} модифицированного графа $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1})$, инцидентной вершине q_d^j нити $\boldsymbol{H_f}$, приписывает требуемый момент времени исполнения, определяемое как

$$T_{b+1}^{j+1} = T_{k_j+1}^i - \left(\sum_{m=d}^{k} t_p(O_i) + t_{\text{п}} \right),$$

а также номер ресурса R_p, которому необходимо передать результаты исполнения операции, приписанной вершине q_b^{j+1};

– определяет величину задержки исполнения операций ветви $\boldsymbol{H_f}$

$$\Delta T_p^l = \begin{cases} T_{\text{тек}} - T_1^f, & \text{если } T_{\text{тек}} > T_1^f \\ 0, & \text{если } T_{\text{тек}} \le T_1^f \end{cases}$$

где $T_1^f = T_{k+1}^f - \left(\sum_{i=1}^{k} t_p(O_i^f) + t_{\text{п}} \right)$

6. $i = 1$.

7. Агент AR_p приступает к исполнению последовательности операций, приписанных вершинам ветви $\boldsymbol{H_f}$; $d = 1$.

8. Агент AR_p проверяет наличие всех результатов, необходимых для выполнения операции O_d^f, приписанной вершине $q_d^f \in \boldsymbol{H}_f$, и поступающих от ресурсов, выполняющих смежных с ней вершин графа задания. Если результаты этих операций еще не поступили на ресурс R_p, то перейти к 8, иначе

9. Агент AR_p инициирует выполнение операции O_d^f, приписанной вершине $q_d^f \in \boldsymbol{H}_f$, с помощью локального УУ$_P$ «своего» ресурса R_p.

10. $d = d + 1$, если $d \le k$, то переход к 8, иначе

11. Агент AR_p выходит из сообщества $\boldsymbol{R}_l$. Результат выполнения последней операции ветви $\boldsymbol{H}_f$ передается ресурсу R_c, номер которого приписан конечной вершине q_k^f ветви $\boldsymbol{H}_f$, или потребителю, если последняя вершина выполненной ветви $\boldsymbol{H}_f$ – это конечная вершина графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, т.е. $q_k^f = q_k$.

13. По завершении задания Z_l агенту AR_p начисляются премиальные баллы

$$S_p^l = \begin{cases} S_{max}^l \cdot \dfrac{V_l^f}{V_l} - \Delta S_p^l, & \text{если } \Delta T^l < \Delta T_{max}^l, \\ 0, & \text{если } \Delta T^l \ge \Delta T_{max}^l, \end{cases}$$

где $V_l^f = \sum_{i=1}^{k} v(O_i^f)$ – суммарная трудоемкость операций ветви $\boldsymbol{H}_f$, выполненных ресурсом R_p;

$V_l = \sum_{j=1}^{M} v(O_j)$ – суммарная трудоемкость всех операций задания Z_l;

S_{max}^l – максимальное число премиальных баллов, начисляемых в случае выполнения задания Z_l к требуемому моменту T_{max}^l;

ΔT_{max}^l – максимальная допустимая задержка выполнения задания Z_l;

ΔT^l – задержка выполнения задания Z_l относительно момента T_{max}^l, установленного Заказчиком;

$$\Delta S_p^l = S_{max}^l \cdot \frac{\Delta T_p^f}{\Delta T_{max}^l};$$

ΔT_p^f – задержка выполнения ветви $\boldsymbol{H}_f$ задания Z_l ресурсом R_p.

3.3. Алгоритм поведения агентов ресурсов СРС с учетом вероятности успешного выполнения операций

Выше считалось, что время выполнения операции $O_i \in \boldsymbol{O}$ ресурсом R_p определяется выражением

$$t_p(O_i) = \frac{v(O_i)}{D_p(O_i)},$$

где $v(O_i)$ – трудоемкость операции O_i;

$D_p(O_i)$ – производительность ресурса R_p при выполнении операции O_i.

Однако в реальности время выполнения операции O_i ресурсом R_p может зависеть не только от производительности ресурса R_p, но и от множества других факторов, в том числе внешних.

Поэтому в более общем случае целесообразно ввести понятие вероятности успешного выполнения операции O_i ресурсом R_p, как

$$a_p(O_i) = F_p(t_i), \tag{3.9}$$

где $a_p(O_i)$ – вероятность выполнения операции O_i ресурсом R_p;

t_i – период времени, в течение которого операция O_i выполняется ресурсом R_p;

F_p – некоторая функция.

Очевидно, что чем больше период времени t_i, тем больше вероятность того, что ресурс R_p успешно выполнит операцию O_i. В простейшем случае функция $F_p(t_i)$ может иметь следующий вид (рис. 3.2):

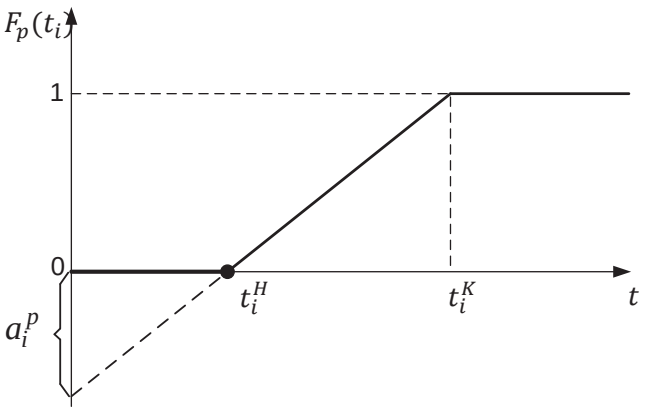


Рис. 3.2. Возможный вид функции $F_p(t_i)$

и описываться следующим выражением

$$a_p(O_i) = \begin{cases} 0, & \text{если } t_i \le t_i^H, \\ \frac{t_i - t_i^H}{t_i^K - t_i^H}, & \text{если } t_i^H < t_i < t_i^K, \\ 1, & \text{если } t_i \ge t_i^K, \end{cases} \quad (3.10)$$

где t_i^H – период времени, в течение которого и меньше операция O_i вообще не выполнима ресурсом R_p;

t_i^K – период времени, в течение которого и больше операция O_i гарантированно выполнима ресурсом R_p.

При этом очевидно, что значения t_i^H и t_i^K должны зависеть от трудоемкости $v(O_i)$ операции O_i, производительности $D_p(O_i)$ ресурса R_p при ее выполнении, а также других факторов.

Рассмотрим принципы организации процедуры мультиагентного диспетчирования ресурсов СРС при таких исходных условиях.

Как и ранее, будем считать, что дескриптор задания Z_l, размещаемого на ДО, содержит момент времени $T_{\max}^l$, к которому это задание должно быть выполнено, а также количество премиальных баллов S^l, которые будут начислены за его успешное выполнение к требуемому моменту времени, причем

$$S^l = \begin{cases} S_{max}^l \cdot \frac{\Delta T_{max}^l - \Delta T^l}{\Delta T_{max}^l}, & \text{если } \Delta T^l \le \Delta T_{max}^l, \\ 0, & \text{если } \Delta T^l > \Delta T_{max}^l, \end{cases}$$

где S_{max}^l – максимальная величина премии, получаемая при условии выполнения задания Z_l к установленному моменту времени $T_{\max}^l$;

ΔT^l – задержка выполнения задания Z_l относительно момента $T_{\max}^l$;

ΔT_{max}^l – максимально допустимая задержка выполнения задания Z_l относительно момента $T_{\max}^l$

Будем считать, что каждый агент AR_p знает функции вероятности $a_p(O_i)$ выполнения всех операций подмножества $\boldsymbol{O}_p$ с помощью «его» ресурса R_p.

Допустим, что агент AR_p ресурса R_p обнаружил на ДО задание Z_l. Как показано в предыдущем разделе (см. выражение 3.8), премия, получаемая агентом AR_p за участие в сообществе $\boldsymbol{R}_l$ по выполнению задания Z_l будет составлять

$$S_p^l = \begin{cases} S_{max}^l \cdot \frac{V_l^p}{V_l} - \Delta S_p^l, & \text{если } \Delta T^l < \Delta T_{max}^l, \\ 0, & \text{если } \Delta T^l \ge \Delta T_{max}^l, \end{cases}$$

где V_l^p – суммарная трудоемкость операций задания Z_l, выполненных ресурсом R_p;

V_l – суммарная трудоемкость всех операций задания Z_l;

ΔS_p^l – штраф, накладываемый на агента AR_p за задержку ΔT_p^l, которую он внес в общую задержку ΔT^l выполнения задания Z_l.

В предыдущем разделе величина ΔS_p^l определялась с помощью выражения (3.7). Однако при этом считалось, что время выполнения отдельных операций O_i однозначно определяется отношением их трудоемкости к производительности ресурса R_p. В рассматриваемом случае данное выражение не работает.

Поэтому в данном случае для определения величины ΔS_p^l можно предложить следующий подход.

Допустим, что агент R_p выделил в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ среди всех ветвей, конечным вершинам которых приписано требуемые моменты времени T_{k+1}^f их исполнения, ветвь $\boldsymbol{H}_f =< q_1^f, q_2^f, ..., q_k^f >$.

Оценим вероятность успешного выполнения данной ветви $\boldsymbol{H}_f$ к установленному моменту времени T_{k+1}^f. Очевидно, что вероятность успешного выполнения всех операций ветви $\boldsymbol{H}_f$ будет определяться следующим выражением

$$A_p^f = \prod_{i=1}^{k} a_p\left(O_i^f\right), \tag{3.11}$$

причем, согласно выражению (3.10), значение $a_p\left(O_i^f\right)$ (i = 1, 2, …, n) зависит от периода времени, выделяемого ресурсом R_p на выполнение операции O_i^f.

При этом из выражений (3.10) и (3.11) следует, что вероятность успешного выполнения всех операций ветви $\boldsymbol{H}_f$ будет равна 1, если на выполнение операции O_i^f (i = 1, 2, …, n) будет выделяться t_i^K времени. При этом суммарное время, необходимое для того, чтобы вся ветвь $\boldsymbol{H}_f$ была выполнена ресурсом R_p с вероятностью 1, будет составлять

$$t_p^f = \sum_{i=1}^{k} t_i^k.$$

Если $t_p^f \le T_{k+1}^f - T_{\text{тек}}$, то это означает, что ресурс R_p со 100-процентной вероятностью сможет выполнить ветвь $\boldsymbol{H}_f$ к установленному моменту времени T_{k+1}^f. В противном случае, если $t_p^f > T_{k+1}^f - T_{\text{тек}}$, то это означает, что ресурс R_p гарантированно сможет выполнить ветвь $\boldsymbol{H}_f$ с максимальной задержкой

$$\Delta T_{p\,max}^l = t_p^f - T_{k+1}^f + T_{\text{тек}}. \tag{3.12}$$

Оценим теперь вероятность успешного выполнения ветви $\boldsymbol{H}_f =< q_1^f, q_2^f, ..., q_k^f >$ к установленному моменту T_{k+1}^f. Для этого, согласно

выражению (3.10), необходимо определить периоды времени t_i (i = 1, 2, …, k), выделяемые для выполнения отдельных операций O_i^f (i = 1, 2, …, k), приписанных вершинам q_i^f (i = 1, 2, …, n) ветви $\boldsymbol{H}_f$. Последнее можно осуществить следующих образом.

Поскольку известно общее время $T_{k+1}^f - T_{\text{тек}}$, отводимое на выполнение всей ветви $\boldsymbol{H}_f$, а также трудоемкости $v(O_i^f)$ отдельных операций O_i^f (i = 1, 2, …, k), приписанных ее вершинам, то промежуток времени t_i^f, отводимый на выполнение операции O_j^f можно оценить как

$$t_i^f = \frac{(T_{k+1}^f - T_{\text{тек}}) \cdot v(O_j^f)}{\sum_{j=1}^{k} v(O_j^f)}, \tag{3.13}$$

где $v(O_i^f)$ – трудоемкость операции O_i^f (i = 1, 2, …, k).

Тогда вероятность выполнения ресурсом R_p всей ветви $\boldsymbol{H}_f$ к требуемому моменту времени T_{k+1}^1 можно оценить как (см. выражение (3.10))

$$A_p^f = \prod_{i=1}^{k} F_p\left(t_i^f\right). \tag{3.14}$$

Следует отметить, что полученное таким образом значение вероятности A_p^f можно повысить без увеличения общего времени выполнения ветви $\boldsymbol{H}_f$, если учесть следующие соображения. Для этого необходимо проанализировать значения t_i^f (i = 1, 2, …, k) времени, выделяемого на выполнение операции O_i^f, получаемые с помощью выражения (3.13). Если $t_i^f > t_i^K$, то это означает, что период времени t_i^f оказывается больше, чем необходимо для гарантированного выполнения операции O_i^f ветви $\boldsymbol{H}_f$ (см. рис. 3.2), и поэтому время, выделяемое для исполнения операции O_i^f ветви $\boldsymbol{H}_f$, можно уменьшить до величины t_i^K без уменьшения общей вероятности A_p^f успешного выполнения всей ветви $\boldsymbol{H}_f$. Аналогичным образом можно сравнить времена t_i^f и t_i^K для всех операций O_i^f (i = 1, 2, …, k) ветви $\boldsymbol{H}_f$ и в случае, если $t_i^f > t_i^K$ принять, что $t_i^f = t_i^K$, «сэкономив» при этом время $\Delta t = \sum_{i=1}^{k} \Delta t_i$,

где $\Delta t_i = \begin{cases} t_i - t_i^K, \text{если } t_i^f > t_i^K, \\ 0, \qquad \text{если } t_i^f \le t_i^K. \end{cases}$

«Сэкономленное» таким образом время в первую очередь следует направить на увеличение времени выполнении тех операций O_j^f ветви $\boldsymbol{H}_f$, для которых выполняется условие $t_j^f \le t_j^H$, а во вторую очередь на увеличение времени тех операций, для которых выполняется условие $t_j^H < t_j^f < t_j^K$, что позволит повысить вероятность их успешного

выполнении и, как следствие, вероятность успешного выполнения всей ветви $\boldsymbol{H}_f$ в целом.

Описанной выше процедуре отвечает следующий формальный алгоритм оптимизации времени выполнения отдельных операций ветви $\boldsymbol{H}_f$ ресурсом R_p.

Алгоритм 3.3

1. $t_i^f = \frac{\left(T_{k+1}^f - T_{\text{тек}}\right)\cdot v\left(o_i^f\right)}{\sum_{\text{j}=1}^{\text{k}} v\left(o_i^f\right)}$ (i = 1, 2, …, k)
2. $i = 1$; $\Delta t = 0$
3. Если $t_i^f > t_i^K$, то $t_i^f = t_i^K$ и $\Delta t = \Delta t + \left(t_i^f - t_i^K\right)$
4. $i = i + 1$, если $i \le k$, то перейти к 3, иначе
5. $i = 1$
6. Если $\Delta t = 0$, то перейти к 14
7. Если $t_i^f < t_i^H$, то $t_i^f = t_i^H$ и $\Delta t = \Delta t - \left(t_i^H - t_i^f\right)$
8. $i = i + 1$, если $i \le k$, то перейти к 6, иначе
9. $i = 1$
10. Если $\Delta t = 0$, то перейти к 14
11. Если $t_i^H < t_i^f < t_i^K$, то $t_i^f = t_i^f + 1$ и $\Delta t = \Delta t - 1$
12. $i = i + 1$, если $i \le k$, то перейти к 10, иначе
13. Если $\Delta t > 0$, то перейти к 9, иначе
14. Конец

После того, как значение вероятности A_p^f успешного выполнения ветви $\boldsymbol{H}_f$ к установленному моменту времени T_{k+1}^f получено, можно определить вероятность задержки исполнения ветви $\boldsymbol{H}_f$ относительно момента времени T_{k+1}^f как

$$A_{p3}^f = 1 - A_p^f.$$

Тогда величину штрафа ΔS_p^l, накладываемого на агента AR_p, можно установить как

$$\Delta S_p^l = S_{max}^l \cdot \frac{A_{p3}^f \cdot \Delta T_{p\,max}^f}{\Delta T_{max}^l}, \qquad (3.15)$$

где $\Delta T_{p\,max}^l$ определяется с помощью выражения (3.12).

Исходя из приведенных выше соображений, можно сделать вывод, что выбор для исполнения агентом AR_p ветви $\boldsymbol{H}_f$ задания Z_l должен осуществляться на основании следующих правил.

1. Ветви $\boldsymbol{H}_f$ должен быть приписан требуемый момент времени T_{k+1}^f ее исполнения.

2. Премиальные баллы, получаемые агентом AR_p за ее исполнение и определяемые с помощью выражений

$$S_p^f = S_{max}^l \cdot \frac{V_l^f}{V_l} - \Delta S_p^f$$

должны быть максимальны, где $\Delta S_p^f = S_{max}^l \cdot \frac{A_{p3}^f \cdot \Delta T_{p\,max}^f}{\Delta T_{max}^l}$

$$A_{p3}^f = 1 - A_p^f = 1 - \prod_{i=1}^{k} a_p^i$$

$$\Delta T_{p\,max}^f = t_p^f - T_{k+1}^f - T_{\text{тек}}$$

$$t_p^f = \sum_{i=1}^{k} t_i^k$$

Исходя из представленных соображений, можно предложить следующий алгоритм поведения агента AR_p ресурса R_p при распределении операций заданий потока $\boldsymbol{Z}$ с учетом вероятности выполнения отдельных операций и наличии штрафных санкций за задержку выполнения заданий к требуемому моменту времени.

Алгоритм 3.4

1. Агент AR_p свободного ресурса R_p считывает в порядке своей очереди дескрипторы заданий $\boldsymbol{Z} =< Z_1, Z_2, ..., Z_L >$, размещенных на ДО, а также количество премиальных баллов S^l $(l = 1,2,...,L)$, назначенных Заказчиками за их успешное выполнение к требуемому моменту времени $T_{\max}^l$.

2. Если $\boldsymbol{Z} = \varnothing$, то перейти к 1, иначе

3. Агент AR_p среди множества заданий $\boldsymbol{Z}$ выбирает то задание Z_l, для которого значение $S_{\text{ср}}^l = \frac{S_{max}^l}{\sum_{i=1}^{M} v(O_i)}$ максимально.

4. Агент AR_p выделяет в графе $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ задания Z_l ветвь $\boldsymbol{H}_f =< q_1^f, q_2^f, \dots, q_n^f >$, для которой выполняются условия:

– конечной вершине q_k^j приписано требуемое время ее исполнения T_{k+1}^f;

– значение величины

$$S_p^f = S_{max}^l \cdot \frac{V_l^f}{V_l} - \Delta S_p^f$$

максимально, где

$$\Delta S_p^f = S_{max}^l \cdot \frac{A_{p3}^f \cdot \Delta T_{p\,max}^f}{\Delta T_{max}^l}$$

$$A_{p3}^f = 1 - \prod_{i=1}^{k} a_p\left(O_i^f\right)$$

$$\Delta T_{p\,max}^{l} = \sum_{i=1}^{k} t_i^k - T_{k+1}^f - T_{\text{тек}},$$

причем значения t_i^f и, соответственно, $a_p(O_i^f) = F(t_i^f)$ определяются с помощью алгоритма 3.3.

Если таковой нити нет, то $\boldsymbol{Z} = \boldsymbol{Z}/Z_l$, перейти к 2, иначе

5. Агент AR_p принимает на себя выполнение нити $\boldsymbol{H}_f$, для чего модифицирует дескриптор задания Z_l на ДО:

– записывает в список участников сообщества $\boldsymbol{R}_l$ свой номер;

– модифицирует граф $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ задания Z_l путем исключения из него вершин нити $\boldsymbol{H}_f$, т. е. $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1}) = \boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) / \boldsymbol{H}_f$;

– каждой вершине q_b^{j+1} модифицированного графа $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1})$, инцидентной вершине q_d^j нити $\boldsymbol{H}_f$, приписывает требуемый момент времени исполнения, определяемое как

$$T_{b+1}^{j+1} = T_{k_j+1}^i - \left(\sum_{m=d}^{k} t_p(O_i) + t_{\text{п}} \right),$$

а также номер ресурса R_p, которому необходимо передать результаты исполнения операции, приписанной вершине q_b^{j+1}.

6. $i = 1$.

7. Агент AR_p приступает к исполнению последовательности операций, приписанных вершинам ветви $\boldsymbol{H}_f$; $d = 1$.

8. Агент AR_p проверяет наличие всех результатов, необходимых для выполнения операции O_d^f, приписанной вершине $q_d^f \in \boldsymbol{H}_f$, и поступающих от ресурсов, выполняющих смежных с ней вершин графа задания. Если результаты этих операций еще не поступили на ресурс R_p, то перейти к 8, иначе

9. Агент AR_p инициирует выполнение операции O_d^f, приписанной вершине $q_d^f \in \boldsymbol{H}_f$, с помощью локального УУ «своего» ресурса R_p.

10. $d = d + 1$, если $d \le k$, то переход к 8, иначе

11. Агент AR_p выходит из сообщества $\boldsymbol{R}_l$. Результат выполнения последней операции ветви $\boldsymbol{H}_f$ передается ресурсу R_c, номер которой приписан конечной вершине q_k^f ветви $\boldsymbol{H}_f$, или на ДО, если последняя вершина выполненной ветви $\boldsymbol{H}_f$ – это конечная вершина графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, т. е. $q_k^f = q_k$.

12. По завершении задания Z_l агенту AR_p начисляются премиальные баллы

$$S_p^l = \begin{cases} S_{max}^l \cdot \dfrac{V_l^f}{V_l} - \Delta S_p^l, \text{если } \Delta T^l < \Delta T_{max}^l, \\ 0, \qquad\qquad \text{если } \Delta T^l \geq \Delta T_{max}^l, \end{cases}$$

где $V_l^f = \sum_{i=1}^{k} v(O_i^f)$ – суммарная трудоемкость операций ветви $\boldsymbol{H}_f$, выполненных ресурсом R_p;

$V = \sum_{j=1}^{M} v(O_j)$ – суммарная трудоемкость всех операций задания Z_l;

S_{max}^l – максимальное число премиальных баллов, начисляемых в случае выполнения задания Z_l к требуемому моменту T_{max}^l;

ΔT_{max}^l – максимальная допустимая задержка выполнения задания Z_l;

ΔT^l – задержка выполнения задания Z_l относительно момента T_{max}^l, установленного Заказчиком;

$$\Delta S_p^l = S_{max}^l \cdot \frac{\Delta T_p^f}{\Delta T_{max}^l};$$

ΔT_p^f – задержка выполнения ветви $\boldsymbol{H}_f$ ресурсом R_p.

3.4. Алгоритм поведения агента в условиях ограниченного запаса, необходимого для выполнения операций

Выше мы считали, что ресурс R_p может выполнить неограниченное количество операций задания Z_l. В более общем случае количество операций, которое может выполнить ресурс R_p, может зависеть от имеющегося у него некоторого запаса. В качестве такого запаса может выступить, например, ограниченный энергозапас ресурса R_j или, в случае боевых действий, ограниченный боезапас, используемый для выполнения операций по поражению объектов противника.

Будем считать, что ресурс R_i (i = 1, 2, …, N) обладает некоторым ограниченным запасом B_i, причем известно, что при выполнении операции $O_j \in \boldsymbol{O_i}$ ресурс R_i тратит $b_i(O_j)$ своего запаса. Отсюда следует, что общее количество операций $O_1, O_2, \dots, O_k$, которое может выполнить ресурс R_i, ограничено величиной имеющегося у него запаса B_i, причем

$$\sum_{j=1}^{k} b_i(O_j) \le B_i.$$

Выше мы считали, что производительность $D_p(O_i)$ ресурса R_p при выполнении операции O_i известна и не может изменяться. В более общем случае производительность $D_p(O_i)$ может зависеть от запаса $b_p(O_i)$, выделяемого ресурсом R_p для выполнения операции O_i, причем общее количество запаса B_p, находящегося в распоряжении ресурса R_p, как и ранее, считаем ограниченным. В этом случае, изменяя запас $b_p(O_i)$, выделяемый для исполнения операции O_i, ресурс R_p может изменять свою производительность при выполнении той или иной операции и, соответственно, изменять время ее исполнения. Возможность изменения производительности ресурса R_p особенно актуальна при наличии некоторых, заранее неизвестных противоборствующих сил, затрудняющих выполнение операции O_i, поскольку в данном случае ресурс R_p может увеличить свою производительность при выполнении данной операции, компенсируя тем самым заранее непредвиденные временные потери. Однако при этом возникает проблема распределения ограниченного запаса B_p, имеющегося в распоряжении ресурса R_p, между выполняемым им множеством операций задания Z_l.

Будем считать, что производительность $D_p(O_i^f)$ ресурса R_p при выполнении операции O_i^f ветви $\boldsymbol{H}_f$ задания Z_l зависит от выделяемого им для этого запаса b_p^i, т. е.

$$D_p\left(O_i^f\right) = F\left(b_p(O_i^f)\right),$$

где F – некоторая функция,

причем общий запас, имеющийся в распоряжении ресурса R_p и направляемый на выполнение различных операций задания Z_l ограничен величиной B_p, т. е.

$$\sum_{i=1}^{k} b_p(O_i^f) \le B_p,$$

где k – общее число операций ветви $\boldsymbol{H}_f$ задания Z_l, выполняемых ресурсом R_p.

При этом, в простейшем случае, функция $F\left(b_p(O_i^f)\right)$ может иметь следующий вид (рис. 3.3.):

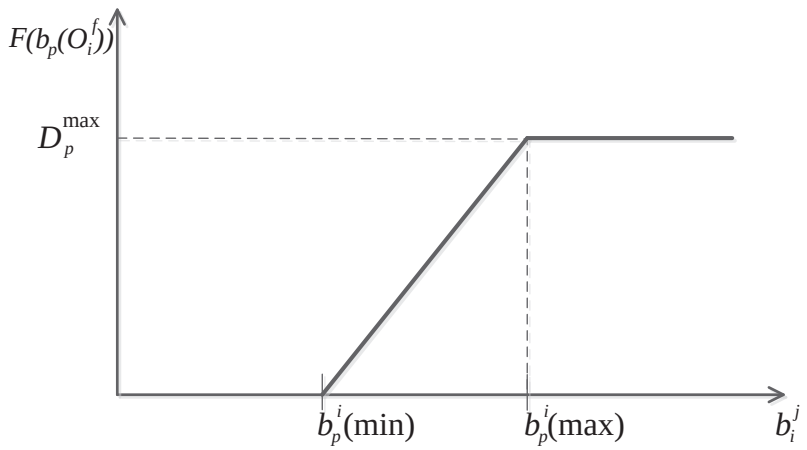


Рис. 3.3. Вид функции $F\left(b_p(O_i^f)\right)$

т.е.

$$D_p\left(O_i^f\right) = \begin{cases} 0, \text{ если } b_p(O_i^f) < b_{min}(O_i^f) \\ K_i b_p(O_i^f), \text{ если } b_{min}(O_i^f) \le b_p(O_i^f) \le b_{max}(O_i^f) \\ D_p^{\max}\left(O_i^f\right), \text{ если } b_p\left(O_i^f\right) > b_{max}\left(O_i^f\right), \end{cases} \quad (3.16)$$

где K_i – некоторый коэффициент.

Иными словами, если ресурс R_p выделяет для исполнения операции O_i^f количество запаса $b_p\left(O_i^f\right)$ меньше, чем $b_{min}(O_i^f)$, то его производительность при выполнении данной операции становится нулевой, т. е. операция невыполнима. Если ресурс R_p выделяет для выполнения операции O_i^f количество запаса $b_p\left(O_i^f\right)$ в пределах от $b_{min}(O_i^f)$ до $b_{max}(O_i^f)$, то его производительность может изменяться линейно с коэффициентом K_i. И, наконец, если ресурс R_p выделяет для выполнения операции O_i^f количество запаса $b_p\left(O_i^f\right)$ большее, чем $b_{max}(O_i^f)$, то его производительность становится максимальной, неизменной и равной $D_p^{\max}\left(O_i^f\right)$.

Допустим, что агент AR_p выделил в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l ветвь $\boldsymbol{H}_f$, операции которой он может выполнить с помощью «своего» ресурса R_p. При этом возникает проблема распределения имеющегося у него запаса B_p между операциями ветви $\boldsymbol{H}_f$ таким образом, чтобы минимизировать общее время t_f исполнения операции всей ветви $\boldsymbol{H}_f$.

В принципе, данная задача может быть решена с помощью методов линейного программирования [24]. Однако применение подобных методов потребует больших вычислительных и, соответственно, временных затрат, что является недопустимым вследствие имеющихся ограничений на общее время выполнения задания Z_l. Поэтому можно предложить следующий упрощенный метод распределения имеющегося у ресурса R_p запаса B_p между операциями ветви $\boldsymbol{H}_f$ с целью минимизации суммарного времени их исполнения.

Общее время исполнения t_f всей ветви $\boldsymbol{H}_f$ в данном случае определяется следующим выражением:

$$t_f = \sum_{i=1}^{k} \frac{v(O_i^f)}{D_p(O_i^f)} = \sum_{i=1}^{k} \frac{v(O_i^f)}{F(b_p(O_i^f))},$$

где $v(O_i^f)$ – трудоемкость выполнения операции O_i^f ветви $\boldsymbol{H}_f$;

$D_p(O_i^f)$ – производительность ресурса R_p при выполнении операции O_i^f;

$b_p(O_i^f)$ – запас, выделяемый ресурсом R_p для выполнения операции O_i^f;

k – число вершин (операций) в ветви $\boldsymbol{H}_f$.

Очевидно, что минимум времени выполнения ветви $\boldsymbol{H}_f$ будет достигаться в случае, если $b_p(O_i^f) = b_{max}(O_i^f)$, $(i = 1,2,\dots,k)$ (см. рис. 3.3.). Однако при этом может оказаться, что $\sum_{i=1}^{k} b_{max}(O_i^f) > B_p$, т. е. общий запас B_p, имеющийся в распоряжении ресурса R_p, будет превышен. Соответственно, необходимо решить, для каких операций O_i^f нити $\boldsymbol{H}_f$ выделяемый на их исполнение запас $b_p(O_i^f)$ может быть уменьшен, с тем чтобы, с одной стороны, суммарно не превысить значение B_p, а с другой стороны, обеспечить минимальное значение времени t_f.

Рассмотрим некоторую операции O_i^f, принадлежащую ветви $\boldsymbol{H}_f$. Если запас $b_p(O_i^f)$, выделяемый для ее исполнения, лежит в пределах

$b_{min}(O_i^f) \le b_p(O_i^f) \le b_{max}(O_i^f)$, то время выполнения этой операции будет равно (см. выражение 3.16):

$$t_p(O_i^f) = \frac{v(O_i^f)}{D_p(O_i^f)} = \frac{v(O_i^f)}{K_i \cdot b_p(O_i^f)}.$$

Допустим, что значение запаса $b_p(O_i^f)$, выделяемого на исполнение операции O_i^f, уменьшается на единицу. При этом, соответственно, время выполнения операции O_i^f увеличивается на величину

$$\Delta t_p(O_i^f) = \frac{v(o_i^f)}{K_i(b_p(o_i^f)-1)} - \frac{v(o_i^f)}{K_i \cdot b_p(o_i^f)} = \frac{v(o_i^f)}{K_i(b_p(o_i^f)-1)\cdot b_p(o_i^f)}. \quad (3.17)$$

Очевидно, что чем меньше будет значение величины (3.17), тем меньше снижение запаса $b_p(O_i^f)$, выделяемого ресурсом R_p на исполнение операции O_i^f, будет влиять на увеличение общего времени выполнения всей ветви $\boldsymbol{H}_f$.

Учитывая эти соображения, можно предложить следующую итерационную процедуру распределения запаса B_p ресурса R_p по операциям O_i^f $(i = 1,2,\dots,k)$ ветви $\boldsymbol{H}_f$.

Сначала всем операциям O_i^f $(i = 1,2,\dots,k)$ устанавливается такое значение запаса $b_p(O_i^f)$, при котором производительность $D_p(O_i^f)$ ресурса R_p становится максимальной, т. е. $b_p(O_i^f) = b_{max}(O_i^f)$. Если при этом выполняется условие

$$\sum_{i=1}^{k} b_p(O_i^f) \le B_p, \quad (3.18)$$

то данное распределение принимается в качестве оптимального. Если же условие (3.18) не выполняется, то для всех операций O_i^f $(i = 1,2,\dots,k)$ вычисляется значение $\Delta t_j(O_i^f)$ (см. выражение (3.17)), после чего определяется номер i-той операции O_i^f, для которой значение $\Delta t_p(O_i^f)$ минимально.

Далее осуществляется проверка условия $b_p(O_i^f) - 1 \le b_{min}(O_i^f)$. Если это условие выполняется, то это говорит о том, что уменьшать значение запаса $b_p(O_i^f)$, выделяемого на исполнение операции O_i^f, нельзя, поскольку оно становится меньше минимально допустимого. В этом случае выделяется следующая по критерию минимальности значения $\Delta t_p(O_c^f)$ операция O_c^f и проверяется условие $b_p(O_c^f) - 1 \le b_{min}(O_c^f)$. Если это условие выполняется, то процесс продолжается далее до тех пор, пока для какой-либо следующей по критерию минимальности значения (3.17) операции O_m^f не окажется, что $b_p(O_m^f) - 1 > b_{min}(O_m^f)$. После этого принимается, что $b_p(O_m^f) = b_p(O_m^f) - 1$, и вновь проверяется условие

(3.18). Если это условие не выполняется, то описанная выше процедура уменьшения выделяемого на операции ветви $\boldsymbol{H}_f$ запаса повторяется заново, и так до тех пор, пока не окажется, что либо $b_p(O_i^f) = b_{min}(O_i^f)$ для всех операций O_i^f $(i = 1,2,\dots,k)$, что говорит о том, что ветвь $\boldsymbol{H}_f$ невыполнима при имеющемся запасе B_p ресурса R_p, либо пока не будет выполнено условие (3.18), что говорит о том, что найдено оптимальное распределение имеющегося запаса B_p по операциям ветви $\boldsymbol{H}_f$.

Очевидно, что выполнение описанной выше процедуры потребует не более чем $\sum_{i=1}^{k} b_{max}(O_i^f) - B_p$ итерационных шагов, т. е. она не потребует больших временных затрат со стороны агента AR_p.

Описанной выше процедуре распределения имеющегося в распоряжении ресурса R_p запаса B_p между отдельными операциями ветви $\boldsymbol{H}_f$, обеспечивающей минимизацию общего времени t_f ее исполнения, отвечает следующий алгоритм.

Алгоритм 3.5

1. Для всех операций O_i^f $(i = 1,2,\dots,k)$ ветви $\boldsymbol{H}_f =< q_1^f, q_2^f, \dots, q_k^f >$ принимается, что $b_p(O_i^f) = b_{max}(O_i^f)$ $i = 1,2,\dots,k)$.

2. Если $\sum_{i=1}^{k} b_p(O_i^f) \le B_p$, где B_p – суммарный запас, имеющийся в распоряжении ресурса R_p, то переход к 9, иначе

3. Для всех операций O_i^f $(i = 1,2,\dots,k)$ ветви $\boldsymbol{H}_f$ вычисляется значение

$$\Delta t_p(O_i^f) = \frac{v(O_i^f)}{K_i(b_p(O_i^f) - 1) \cdot b_p(O_i^f)}.$$

4. Операции O_i^f $(i = 1,2,\dots,k)$ ветви $\boldsymbol{H}_f$ ранжируются в порядке возрастания величины $\Delta t_p(O_i^f)$.

5. $i = 1$.

6. Если $b_p(O_i^f) - 1 > b_{min}(O_i^f)$, то перейти к 8, иначе

7. $i = i + 1$, если $i > k$, то перейти к 10, иначе перейти к 6.

8. $b_p(O_i^f) = b_p(O_i^f) - 1$, перейти к 2.

9. Оптимальное распределение запаса B_p ресурса R_p по операциям O_i^f $i = 1,2,\dots,k)$ ветви $\boldsymbol{H}_f$ найдено, перейти к 11.

10. Ресурс R_p не может выполнить ветвь $\boldsymbol{H}_f$ вследствие недостаточности имеющегося у него запаса B_p.

11. Конец.

На основе изложенных выше соображений можно предложить следующий укрупненный алгоритм поведения агента AR_p ресурса R_p $(p = 1,2,\dots,N)$ при распределении ветвей графа $\boldsymbol{G_l(Q_l, X_l)}$ заданием Z_l в условиях ограниченного запаса B_p на их исполнение.

Алгоритм 3.6

1. Агент AR_p свободного ресурса R_p опрашивает ДО в поисках работы для «своего» ресурса.

2. В случае обнаружения на ДО графа $\boldsymbol{G_l(Q_l, X_l)}$ задания Z_l агент AR_p выделяет в нем подграф $\boldsymbol{G_l^j(Q_l^j, X_l^j)}$, вершинам которого приписаны операции множества $\boldsymbol{O_p}$ (т. е. операции, выполняемые ресурсом R_p).

3. Агент AR_p выделяет в подграфе $\boldsymbol{G_l^j(Q_l^j, X_l^j)}$ ветвь $\boldsymbol{H_f} =< q_1^f, q_2^f, \dots, q_k^f >$, для которой выполняются условия:

– конечной вершине нити q_k^f нити $\boldsymbol{H_f}$ приписано требуемое время ее исполнения T_{k+1}^f;

– $\sum_{j=1}^{k} b_p(O_j^f) \le B_p$, где B_p – имеющийся в наличие у ресурса R_p запас, причем значения $b_p\left(O_i^f\right)$ (i = 1, 2, …, k) определяются с помощью алгоритма 3.5.;

– значение величины премии S_p^f, получаемой агентом AR_p за выполнение ветви $\boldsymbol{H_f}$ максимально, т. е. максимально значение величины

$$S_p^f = S_{max}^l \cdot \frac{V_l^f}{V_l} - \Delta S_p^f,$$

где S_p^f– премия, получаемая агентом AR_p за выполнение операций ветви $\boldsymbol{H_f}$;

V_l^f – суммарная трудоемкость операций ветви $\boldsymbol{H_f}$;

V_l – суммарная трудоемкость всех операций задания Z_l;

ΔS_p^f – штраф за задержку выполнения ветви $\boldsymbol{H_f}$ к требуемому моменту времени T_{k+1}^f.

4. Агент AR_p принимает ветвь $\boldsymbol{H_f}$ к исполнению и модифицирует дескриптор задания Z_l на ДО:

– записывает в список участников сообщества свой номер;

– модифицирует граф $\boldsymbol{G_l^j(Q_l^j, X_l^j)}$ задания Z_l путем исключения из него вершин нити $\boldsymbol{H_f}$, т.е. $\boldsymbol{G_l^{j+1}(Q_l^{j+1}, X_l^{j+1}) = G_l^j(Q_l^j, X_l^j) / H_f}$;

– каждой вершине q_b^{j+1} модифицированного графа $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1},\boldsymbol{X}_l^{j+1})$, инцидентной вершине q_d^j нити $\boldsymbol{H}_f$, приписывает требуемый момент времени исполнения, определяемое как

$$T_{b+1}^{j+1} = T_{k_j+1}^{i} - \left(\sum_{m=d}^{k} t_p(O_i) + t_{\text{п}} \right),$$

а также номер ресурса R_p, которому необходимо передать результаты исполнения операции, приписанной вершине q_b^{j+1}.

5. $i = 1$.

6. Агент AR_p приступает к исполнению последовательности операций, приписанных вершинам ветви $\boldsymbol{H}_f$; $d = 1$.

7. Агент AR_p проверяет наличие всех результатов, необходимых для выполнения операции O_d^f, приписанной вершине $q_d^f \in \boldsymbol{H}_f$, и поступающих от ресурсов, выполняющих смежных с ней вершин графа задания. Если результаты этих операций еще не поступили на ресурс R_p, то перейти к 7, иначе

8. Агент AR_p инициирует выполнение операции O_d^f, приписанной вершине $q_d^f \in \boldsymbol{H}_f$, с помощью локального УУ «своего» ресурса R_p.

9. $d = d + 1$, если $d \le k$, то переход к 8, иначе

10. Агент AR_p выходит из сообщества $\boldsymbol{R}_l$. Результат выполнения последней операции ветви $\boldsymbol{H}_f$ передается ресурсу R_c, номер которой приписан конечной вершине q_k^f ветви $\boldsymbol{H}_f$, или на ДО, если последняя вершина выполненной ветви $\boldsymbol{H}_f$ – это конечная вершина графа $\boldsymbol{G}_l(\boldsymbol{Q}_l,\boldsymbol{X}_l)$, т.е. $q_k^f = q_k$.

11. По завершении задания Z_l агенту AR_p начисляются премиальные баллы

$$S_p^l = \begin{cases} S_{max}^l \cdot \dfrac{V_l^f}{V_l} - \Delta S_p^l, \text{если } \Delta T^l < \Delta T_{max}^l, \\ 0, \text{если } \Delta T^l \ge \Delta T_{max}^l, \end{cases}$$

где $V_l^f = \sum_{i=1}^{k} v(O_i^f)$ – суммарная трудоемкость операций ветви $\boldsymbol{H}_f$, выполненных ресурсом R_p;

$V_l = \sum_{j=1}^{M} v(O_j)$ – суммарная трудоемкость всех операций задания Z_l;

S_{max}^l – максимальное число премиальных баллов, начисляемых в случае выполнения задания Z_l к требуемому моменту T_{max}^l;

ΔT_{max}^{l} – максимальная допустимая задержка выполнения задания Z_l;

ΔT^{l} – задержка выполнения задания Z_l относительно момента T_{max}^{l}, установленного Заказчиком;

$$\Delta S_p^l = S_{max}^l \cdot \frac{\Delta T_p^f}{\Delta T_{max}^l};$$

ΔT_p^f – задержка выполнения ветви $\boldsymbol{H}_f$ ресурсом R_p;

$$\Delta T_p^f = \begin{cases} T_{\text{тек}} - T_f, \text{если } T_{\text{тек}} > T_f \\ 0, \text{если } T_{\text{тек}} \le T_f, \end{cases}$$

где $T_f = \sum_{i=1}^{k} \frac{v(O_i^f)}{F\left(b_p\left(O_i^f\right)\right)} + t_{\text{п}}$.

3.5. Метод мультиагентного диспетчирования, минимизирующий время выполнения заданий

Выше предполагалось, что премия S^l устанавливается за успешное выполнение задания Z_l к требуемому моменту времени $T^l_{\max}$. В данном разделе рассмотрим случай, когда требуемое время выполнения задания Z_l не установлено, а премия S^l зависит от времени t^l выполнения задания Z_l, причем чем меньше это время, тем больше размер премии, т. е.

$$S^l = F(t^l),$$

где F – некоторая функция с обратно пропорциональной зависимостью.

Например, в простейшем случае функция $S^l = F(t^l)$ может иметь следующий вид.

$$S^l = \frac{K_l}{t^l},$$

где K_l – некоторый коэффициент.

При такой постановке цель работы мультиагентного диспетчера СРС будет заключаться в минимизации времени выполнения всех поступающих заданий $Z_l \subseteq \boldsymbol{Z}$ $(j = 1,2,...,M)$ с помощью имеющегося множества $\boldsymbol{R}$ ресурсов СРС [25].

Для достижения этой цели в состав мультиагентного диспетчера кроме агентов AR_i $(i = 1,2,...,N)$, представляющих различные ресурсы $R_i \in \boldsymbol{R}$ $(i = 1,2,...,N)$, предлагается ввести дополнительных агентов – агентов заданий AZ_j $(j = 1,2,...,M)$, отвечающих за минимизацию общего времени выполнения того или иного задания $Z_j \subseteq \boldsymbol{Z}$ $(j = 1,2,...,M)$ (рис. 3.4).

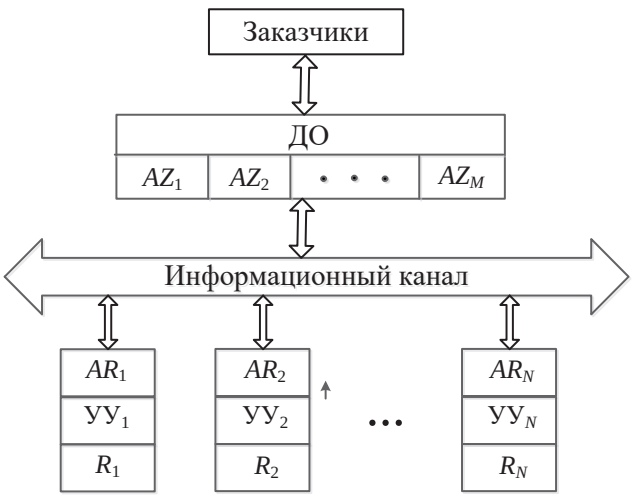


Рис. 3.4. СРС с мультиагентным диспетчером, включающим агентов двух типов – агентов задач AZ и агентов ресурсов AR

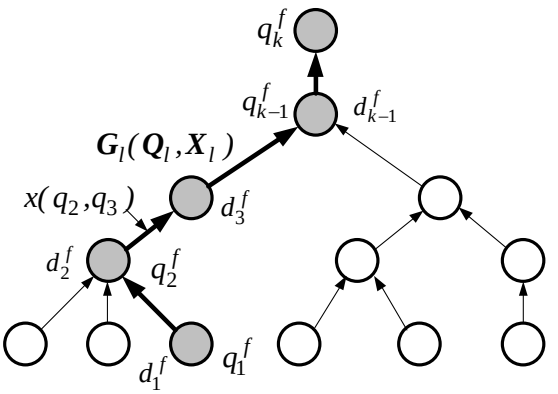
Для общности дальнейших рассуждений будем считать, что мультиагентный диспетчер реализуется на базе наиболее сложной СРС, а именно – гетерогенной СРС второго типа, т. е. все ресурсы $R_i(i=1,2,...,N) \in \boldsymbol{R}$ выполняют различные множества операций $\boldsymbol{O}_i$, причем $\boldsymbol{O}_i \cap \boldsymbol{O}_j \neq \boldsymbol{O}$ $(i=1,2,...,N, j=1,2,...,i-1,i+1,...,N)$, а производительность различных ресурсов при выполнении идентичных операций может быть также различна.

Тогда работу мультиагентного диспетчера СРС, целью которого является минимизация времени выполнения потока заданий $\boldsymbol{Z}$, поступающих от различных в произвольные моменты времени, в укрупненном виде можно представить следующим образом.

1. Дескриптор задания $Z_l \in \boldsymbol{Z}$, включающий графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, а также размер премии $S^l = F(t^l)$ за его выполнение, где t^l время выполнения задания Z_l, размещается на ДО.

2. Каждому заданию $Z_l \in \boldsymbol{Z}$, размещенному на ДО, назначается свой агент AZ_l.

3. Агент AZ_l задания Z_l выделяет в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ наиболее трудоемкую ветвь $\boldsymbol{H}_1 =< q_1^1, q_2^1, ..., q_k^1 >$, для которой значение $V_1 = \left(\sum_{i=1}^{k} v(O_i^1) \right)$ максимально, где $v(O_i^1)$ $(j=1,2,...,k)$ – трудоемкость операции O_i^1, приписанной вершине $q_i^1 \in \boldsymbol{H}_1$, и устанавливает возможный (желательный) момент времени начала ее исполнения $T_{\text{н}}^1 = T_{\text{тек}}$, где $T_{\text{тек}}$ – текущий момент времени. После этого ветвь $\boldsymbol{H}_1$ выставляется на ДО для исполнения (рис. 3.5).



- вершины нити $\boldsymbol{H}_1 =< q_1^f, q_2^f, ..., q_k^f >$

Рис. 3.5. Выделение ветви $\boldsymbol{H}_1$ в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l

4. Агенты AR_j различных ресурсов R_j $(i=1,2,...,N)$ последовательно (например, в порядке нумерации) опрашивают ДО в поисках работы для «своего» ресурса R_j.

5. Если агент AR_j обнаружил на ДО ветвь $\boldsymbol{H}_1$, выставленную на исполнение агентом AZ_l задания Z_l, то он оценивает возможность своего участия в ее исполнении.

6. Для этого агент AR_j выделяет в ветви $\boldsymbol{H}_1 =< q_1^1, q_2^1, ..., q_k^1 >$ подветвь $\boldsymbol{H}_{1j}^1 =< q_1^1, q_2^1, ..., q_b^1 > \subseteq \boldsymbol{H}_1$ $(b \le k)$, вершинам которой приписаны операции множества $\boldsymbol{O}_j$, т. е. операции, выполняемые ресурсом R_j (рис.3.6). Кроме того, агент AR_j определяет момент времени $T_{\text{нj}}^1$, когда он сможет приступить к исполнению подветви $\boldsymbol{H}_{1j}^1$ (т. е. когда он освободиться от выполнения ранее закрепленных за ним ветвей графа задания), а также момент времени $T_{\text{кj}}^1$ окончания исполнения подветви $\boldsymbol{H}_{1j}^1$, причем

$$T_{\text{кj}}^1 = T_{\text{нj}}^1 + \sum_{i=1}^{b} \frac{v(O_i^1)}{D_j(O_i^1)}, \tag{3.19}$$

где $v(O_i^1)$ – трудоемкость операции O_i^1, приписанной вершине $q_i^1 \in \boldsymbol{H}_{1j}^1$ $(i=1,2,...,b)$;

$D_j(O_i^1)$ – производительность ресурса R_j при выполнении операции O_i^1.

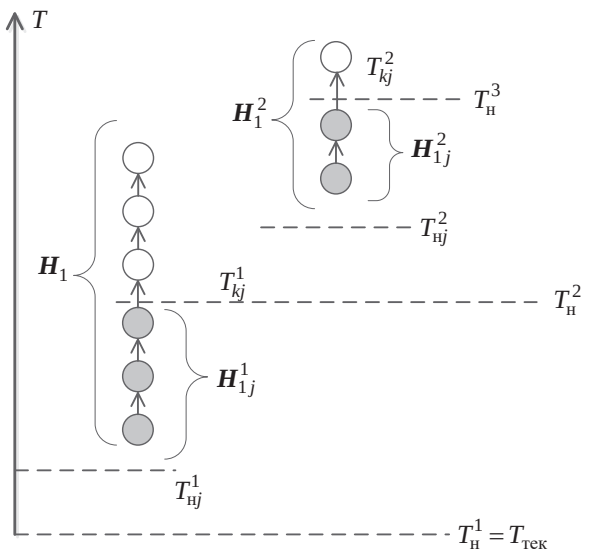


Рис. 3.6. Распределение операции ветви $\boldsymbol{H}_1$ между агентами AR_j $(j=1,2,...,N)$ ресурсов СРС

Выделенная таким образом подветвь $\boldsymbol{H}_{1j}^{1}$ с приписанными ей моментами времени начала исполнения $T_{\text{нj}}^{1}$ и окончания исполнения $T_{\text{кj}}^{1}$ сообщается агентом AR_p агенту AZ_l задания Z_l в качестве «предложения» по его возможному участию в исполнении ветви $\boldsymbol{H}_1$.

7. После того, как агенты всех ресурсов R_j $(i=1,2,...,N)$, входящих в состав СРС, аналогичным образом сформируют свои «предложения» по участию в исполнении ветви $\boldsymbol{H}_1$, агент AZ_l задачи Z_l среди всех поступивших «предложений» должен выбрать наилучшее. В качестве такого наилучшего «предложения» целесообразно принять ту подветвь $\boldsymbol{H}_{1p}^{1}$, «предложенную» агентом AR_p, для которой значение величины

$$E_p = \frac{\sum_{i=1}^{b} v(O_i^1)}{T_{\text{кр}}^1 - T_{\text{н}}^1}$$

максимально, где $v(O_i^1)$ – трудоемкость операции O_i^1 $(i=1,2,...,b)$, приписанной вершине $q_i^1 \in \boldsymbol{H}_{1p}^{1}$; $T_{\text{н}}^1$ – желательный момент времени начала исполнения ветви $\boldsymbol{H}_1$, установленный агентом задачи AZ_l; T_{kp}^1 – момент времени завершения исполнения подветви $\boldsymbol{H}_{1p}^{1}$, предлагаемый агентом AR_p.

Последнее выражение определяет некоторую относительную трудоемкость «предложения» агента AR_p в единицу времени. Очевидно, что чем больше значение величины E_p, тем более трудоемкая часть ветви $\boldsymbol{H}_1$ будет выполнена агентом AR_p, начиная с желательного момента времени $T_{\text{н}}^1$ начала ее исполнения.

8. Подветвь $\boldsymbol{H}_{1p}^{1}$, для которой значение E_p максимально, закрепляется за агентом AR_p, о чем ему направляется соответствующее сообщение от агента задания AZ_l, и агент AR_p включается в состав сообщества $\boldsymbol{R}_l$ по выполнению задания Z_l. В графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l каждой вершине q_j^1 $(j=1,2,...,b)$ подветви $\boldsymbol{H}_{1p}^{1}$ приписывается номер ресурса R_p, за которым закреплено ее исполнение, а также момент времени ее исполнения T_j^1, определяемый как

$$T_j^1 = T_{\text{нр}}^1 + \sum_{i=1}^{j} \frac{v(O_i^1)}{D_p(O_i^1)}, \qquad (3.20)$$

где $v(O_i^1)$ – трудоемкость операции O_i^1, приписанной вершине q_i^1;

$D_p(O_i)$ – производительность ресурса R_p при выполнении операции O_i^1.

9. Далее агент AZ_l задания Z_l исключает вершины $q_1^1, q_2^1, ..., q_b^1$, подветви $\boldsymbol{H}_{1p}^1$ из ветви $\boldsymbol{H}_1$, в результате чего формируется новая (укороченная) ветвь $\boldsymbol{H}_1^2 =< q_{b+1}^1, ..., q_k^1 >$ (рис. 3.6), а первой вершине q_{b+1}^1 этой ветви приписывается номер ресурса R_p, от которого должны поступить результаты, необходимые для ее выполнения, а также возможный (желательный) момент времени начала ее исполнения $T_{\text{н}}^2 = T_{\text{кр}}^1$, где $T_{\text{кр}}^1$ – момент времени окончания исполнения подветви $\boldsymbol{H}_{1p}^1$, определяемый согласно выражению (3.19).

10. Далее аналогично, как описано выше, агенты различных ресурсов R_j $(i = 1,2,...,N)$ оценивают свои возможности по участию в исполнении ветви $\boldsymbol{H}_1^2$. При этом агент AR_j $(j = 1,2,...,N)$ выделяет подветвь $\boldsymbol{H}_{1j}^2 =< q_{b+1}^1, ..., q_m^1 > \subseteq \boldsymbol{H}_1^2$ $(m \le k)$ (см. рис. 3.6), вершинам которой приписаны операции подмножества $\boldsymbol{O}_j \subseteq \boldsymbol{O}$, выполняемые ресурсом R_j, а также определяет момент времени начала ее исполнения $T_{\text{нj}}^2$, (т. е. момент, когда он может приступить к ее исполнению, причем $T_{\text{нj}}^2 \ge T_{\text{н}}^2$), а также момент времени окончания ее исполнения $T_{\text{кj}}^2$, определяемый как

$$T_{\text{кj}}^2 = T_{\text{нj}}^2 + \sum_{i=b+1}^{m} \frac{v(O_i^1)}{D_j(O_i^1)}. \qquad (3.21)$$

Выделенная таким образом подветвь $\boldsymbol{H}_{1j}^2$ направляется агентом AR_j агенту AZ_l задания Z_l в качестве его «предложения» по участию в выполнении ветви $\boldsymbol{H}_1^2$.

11. После того, как все агенты AR_j $(j = 1,2,...,N)$ сообщат агенту AZ_l задания Z_l свои «предложения» по участию в выполнении ветви $\boldsymbol{H}_1^2$, последний выбирает наилучшее «предложение» того агента AR_c, предлагающего к исполнению подветвь $\boldsymbol{H}_{1c}^2 =< q_{b+1}^1, q_{b+2}^1, ..., q_m^1 >$ $(m \le k)$, для которой величина

$$E_c = \frac{\sum_{b+1}^{m} v(O_i^1)}{T_{кс}^2 - T_н^2} \qquad (3.22)$$

максимальна.

Вершины подветви $\boldsymbol{H}_{1c}^2$ закрепляются за агентом AR_c, о чем ему направляется соответствующее сообщение агентом AZ_l, и агент AR_p включается в сообщество $\boldsymbol{R}_l$ по выполнению задания Z_l. Кроме того, в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ каждой вершине q_j^1 $(i = b+1,...,m)$ подветви $\boldsymbol{H}_{1c}^2$ приписывается номер ресурса R_c, за которым закреплено ее исполнение, и момент времени T_{jc}^1 ее исполнения, определяемый как

$$T_j^1 = T_{нс}^2 + \sum_{i=b+1}^{j} \frac{v(O_i^1)}{D_c(O_i^1)}, \qquad (3.23)$$

где $v(O_i^1)$ – трудоемкость операции O_i^1, приписанной вершине q_i^1;

$D_c(O_i^1)$ – производительность ресурса R_c при выполнении операции O_i^1.

12. После этого агент задания AZ_l формирует новую ветвь $\boldsymbol{H}_1^3 =< q_{m+1}^1,...,q_k^1 >$ путем исключения из ветви $\boldsymbol{H}_1^2$ вершин подветви $\boldsymbol{H}_{1c}^2$, т. е. $\boldsymbol{H}_1^3 = \boldsymbol{H}_1^2 / \boldsymbol{H}_{1c}^2$, и первой вершине q_{m+1}^1 этой ветви приписывает возможный (желательный) момент время начала ее исполнения $T_н^3 = T_{кс}^2$, где $T_{кс}^2$ – момент времени завершения выполнения подветви $\boldsymbol{H}_{1c}^2$, определяемый согласно (3.21).

Далее ветвь $\boldsymbol{H}_1^3$ вновь выставляется агентом задания AZ_l на исполнение и т. д., до тех пор, пока не окажется, что очередная ветвь $\boldsymbol{H}_1^k = \varnothing$, что говорит о том, что все вершины ветви $\boldsymbol{H}_1$ закреплены за агентами различных ресурсов R_j $(i = 1,2,...,N)$.

13. После этого агент AZ_l задания Z_l исключает ветвь $\boldsymbol{H}_1$ из графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, в результате чего формируется новый граф $\boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1) = \boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l) / \boldsymbol{H}_1$ (рис. 3.7). В модифицированном графе $\boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1)$ агент AZ_l снова выделяет наиболее трудоемкую ветвь $\boldsymbol{H}_2 =< q_1^2, q_2^2,...,q_r^2 >$ (рис. 3.7), которая выставляется на исполнение и распределяется между агентами ресурсов R_j $(i = 1,2,...,N)$ аналогично тому, как это было описано выше. В результате выполнения этой процедуры всем вершинам ветви $\boldsymbol{H}_2 =< q_1^2, q_2^2,...,q_r^2 >$ в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$

будут приписаны номера ресурсов R_j, за которыми закреплено их исполнение, а также моменты времени T_f^2 $(f = 1,2,...,r)$ их исполнения.

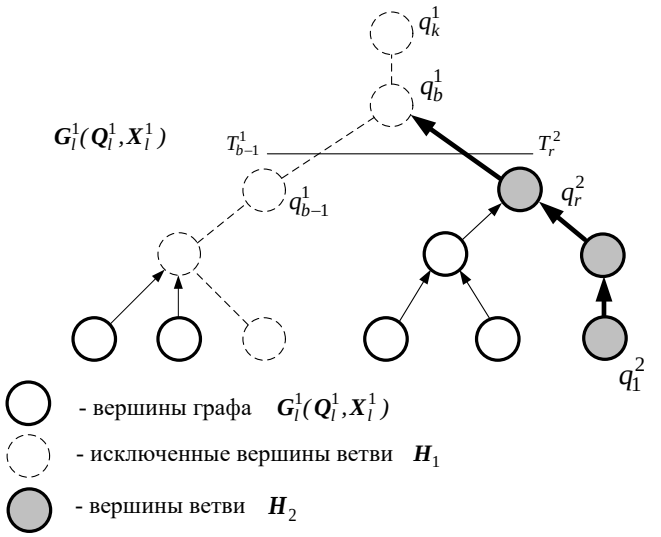


Рис. 3.7. Граф $G_l^1(Q_l^1, X_l^1)$ задачи Z_l, модифицированный агентом AZ_l

14. Поскольку ветвь $H_2 =< q_1^2, q_2^2, ..., q_r^2 >$ является «ответвлением» ветви $H_1 =< q_1^1, q_2^1, ..., q_k^1 >$ (т. е. конечная вершина q_r^2 ветви H_2 инцидентна одной из вершин q_b^1 ветви H_1) (см. рис.3.7), то необходимо проверить согласованность момента времени завершения исполнения ветви H_2 и момента времени начала исполнения инцидентной ей вершины q_b^1 ветви H_1. Для этого агент AZ_l задания Z_l сравнивает момент времени $T_к^2$ завершения исполнения ветви H_2, определяемый как

$$T_к^2 = T_r^2 + t_п, \tag{3.24}$$

где T_r^2 – момент времени, приписанный конечной вершине q_r^2 ветви H_2, $t_п$ – время пересылки результата выполнения ветви H_2 от ресурса R_c к ресурсу R_p, исполняющему ветвь H_1, с моментом времени $T_{b-1}^1 = T_b^1 - t_p(O_b^1)$, где T_b^1 – момент времени исполнения вершины q_b^1 ветви H_1, смежной с конечной вершиной q_r^2 ветви H_2, $t_p(O_b^1) = \dfrac{v(O_b^1)}{D_p(O_b^1)}$ – период времени выполнения операции O_b^1, приписанной вершине q_b^1, ресурсом R_p, за которым закреплено ее исполнение.

Если оказывается, что $T_{\text{к}}^2 > T_{b-1}^1$, то это означает, что результат выполнения операций ветви $\boldsymbol{H}_2$, необходимый для выполнения операции, приписанной вершине $q_b^1 \in \boldsymbol{H}_1$, будет получен позднее, чем результат выполнения операции вершины q_{b-1}^1 ветви $\boldsymbol{H}_1$, также необходимый для выполнения операции, приписанной вершине q_b^1. В этом случае агент задания AZ_l осуществляет корректировку временного графика исполнения ветви $\boldsymbol{H}_1$ путем увеличения моментов времени исполнения, приписанных ее вершинам $q_b, q_{b+1}, ... q_k$, на величину $\Delta T = T_{\text{к}}^2 - T_{b-1}^1$.

15. После этого агент AZ_l задания Z_l формирует новый граф задания $\boldsymbol{G}_l^2(\boldsymbol{Q}_l^2, \boldsymbol{X}_l^2) = \boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1) / \boldsymbol{H}_2$ путем исключения вершин ветви $\boldsymbol{H}_2$ (рис. 3.8), и в этом графе вновь выделяет наиболее трудоемкую ветвь $\boldsymbol{H}_3$, которую выставляет на исполнение на ДО. Процесс продолжается далее до тех пор, пока не будут распределены все вершины (операции) графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ между различными ресурсами СРС, т. е. до тех пор, пока не окажется, что после очередной модификации $\boldsymbol{G}_l^d(\boldsymbol{Q}_l^d, \boldsymbol{X}_l^d) = \varnothing$. В результате такой процедуры всем вершинам $q_i \in \boldsymbol{Q}_l$ графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ будут приписаны номера ресурсов R_j $(j = 1,2,...,N)$, отвечающих за их исполнение, а также планируемые моменты времени T_i их исполнения, т. е. агентом AZ_l будет сформирован план (временной график) выполнения всего задания Z_l.

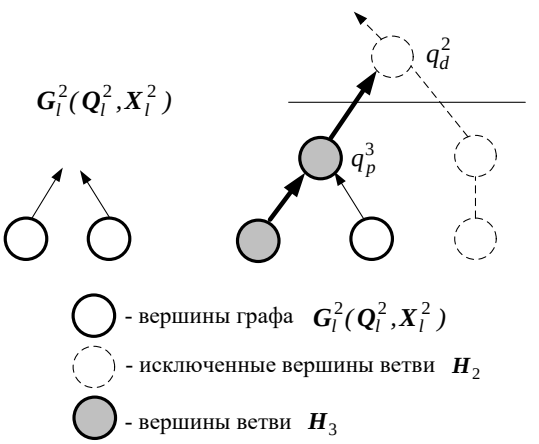


Рис. 3.8. Граф $\boldsymbol{G}_l^2(\boldsymbol{Q}_l^2, \boldsymbol{X}_l^2)$, модифицированный агентом AZ_l

16. После этого агент AZ_l задания Z_l отправляет всем агентам ресурсов R_j $(i = 1,2,...,N)$, задействованным в выполнении данного задания (т. е. входящим в сообщество $\boldsymbol{R}_l$), команду на выполнение

принятых на себя операций согласно сформированному плану (временному графику).

17. Когда агент AR_p ресурса R_p получает подтверждение от агента AZ_l задания Z_l команду о выполнении закрепленной за ним подветви $\boldsymbol{H}_f^m =< q_1^f, q_2^f, ..., q_r^f >$ графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, он включает операции, приписанные вершинам подветви $\boldsymbol{H}_f^m$, в свой график работы.

18. При наступлении момента времени $T_{нр}^f$ начала исполнения подветви $\boldsymbol{H}_f^m$ агент AR_p приступает к выполнению последовательности операций O_i^f $(i = 1,2,...,r)$, приписанных ее вершинам $q_i^f \in \boldsymbol{H}_f^m$.

19. По завершении выполнения задания Z_l получаемая премия $S^l = F(t^l)$ делится между агентом AZ_l задания Z_l и агентами AR_p $(p = 1,2,\dots,N_l)$ $(N_l = |\boldsymbol{R}_l|)$, ресурсов, участвующих в его выполнении, в некоторой установленной пропорции: $a_1 \cdot S^l$ премиальных баллов зачисляются агенту AR_2 и $a_2 \cdot S^l$ премиальных баллов зачисляются агентам сообщества $\boldsymbol{R}_l$, внутри которого, как и в предыдущих случаях, они делятся между отдельными агентами пропорционально трудоемкости выполненных ими операций, причем $a_1 + a_2 = 1$.

Рассмотрим более подробно алгоритмы работы агента задания AZ_l и агента ресурса AR_p при реализации описанной выше процедуры мультиагентного диспетчирования ресурсов СРС, минимизирующей время выполнения отдельных заданий потока $\boldsymbol{Z}$.

Основная функция агента AZ_l заключается минимизации времени выполнения всего задания Z_l. Поскольку время выполнения задания Z_l в первую очередь зависит от времени выполнения наиболее трудоемкой ветви (критического пути) в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, то первым делом агент AZ_l должен обеспечить распределение ее операций между отдельными ресурсами СРС. С этой целью агент AZ_l выделяет в графе задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ наиболее трудоемкую ветвь $\boldsymbol{H}_1 =< q_1^1, q_2^1, ..., q_k^1 >$ и устанавливает желательный (возможный) момент времени начала ее исполнения $T_н^1 = T_{тек}$, после чего выставляет нить $\boldsymbol{H}_1$ на исполнение на ДО (см. рис. 3.6).

После того, как агенты всех ресурсов R_j $(i = 1,2,...,N)$ проанализируют возможность своего участия в выполнении ветви $\boldsymbol{H}_1$ и направят свои «предложения» агенту AZ_l задания Z_l, последний среди всех поступивших выбирает «предложение» того агента AR_p, который обеспечивает выполнение подветви $\boldsymbol{H}_{1p}^1 =< q_1^1, q_2^1, ..., q_b^1 > \subseteq \boldsymbol{H}_1$, для которой величина

$$E_p = \frac{\sum_{i=1}^{b} v(O_i^1)}{T_{кр}^1 - T_н^1}$$

максимальна, где $v(O_i^1)$ $(i = 1,2,...,b)$ – трудоемкость операции O_i^1, прописанной вершине $q_i^1 \in \boldsymbol{H}_{1p}^1$; $T_{кр}^1$ – момент времени завершения исполнения подветви $\boldsymbol{H}_{1p}^1$ агентом AR_p, определяемый с помощью выражения (3.19).

После этого агент AR_p включается в сообщество $\boldsymbol{R}_l$ по выполнению задания Z_l, а в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ всем вершинам подветви $\boldsymbol{H}_{1p}^1$ приписывается номер агента AR_p, отвечающего за их исполнение, а также планируемые моменты времени их исполнения, определяемые с помощью выражения (3.20).

Далее агент AZ_l исключает подветвь $\boldsymbol{H}_1^1$ из ветви $\boldsymbol{H}_1$, в результате чего формируется новая ветвь $\boldsymbol{H}_1^2 \subseteq \boldsymbol{H}_1$, которой агент AZ_l приписывает возможный момент времени начала ее исполнения $T_н^2$, определяемый моментом временем исполнения подветви $\boldsymbol{H}_{1p}^1$ ресурсом R_p, т. е. $T_н^2 = T_{кр}^1$ (см. рис. 3.6). После этого ветвь $\boldsymbol{H}_1^2$ выставляется агентом AZ_l на ДО для исполнения.

Далее агент AZ_l вновь собирает предложения агентов ресурсов AR_j $(j = 1,2,...,N)$ по участию в исполнении ветви $\boldsymbol{H}_1^2$ и среди них выбирает «предложение» того агента AR_c, который обеспечивает выполнение такой подветви $\boldsymbol{H}_{1c}^2 = < q_{b+1}^1, q_{b+2}^1, ..., q_m^1 >$ $(m \le k)$, для которой значение E_c (см. выражение (3.22)) максимально. В результате агент AR_c включается в сообщество $\boldsymbol{R}_l$, а в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$вершинам этой подветви $\boldsymbol{H}_{1c}^2$ приписывается номер агента AR_c, за которым закрепляется ее исполнение, а также планируемые моменты времени их исполнения T_1^j, определяемые с помощью выражения (3.23).

Аналогичный процесс продолжается до тех пор, пока не будут распределены все вершины (операции) ветви $\boldsymbol{H}_1$, т.е. пока не окажется, что очередная подветвь $\boldsymbol{H}_1^m = \varnothing$.

Далее агент задания AZ_l исключает из графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ ветвь $\boldsymbol{H}_1$, в результате чего формируется новый граф $\boldsymbol{G}_l^1(\boldsymbol{Q}_l^1, \boldsymbol{X}_l^1) = \boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l) / \boldsymbol{H}_1$, и выделяет в последнем наиболее трудоемкую ветвь $\boldsymbol{H}_2$, которая выставляется на ДО на исполнение (см. рис. 3.7). После того, как все вершины (операции) ветви $\boldsymbol{H}_2 = < q_1^2, q_2^2, ..., q_r^2 >$ будут

разобраны агентами ресурсов AR_j $(j=1,2,...,N)$, в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l,\boldsymbol{X}_l)$ им будут приписаны номера соответствующих ресурсов, за которыми закреплено их исполнение, а также планируемые моменты времени их исполнения T_i^2 $(i=1,2,...,r)$.

После этого агент задания AZ_l должен проверить согласованность моментов времени исполнения операции O_b^1, приписанной вершине q_b^1 ветви $\boldsymbol{H}_1$, и операций инцидентной ей ветви $\boldsymbol{H}_2$. Для этого агент задания AZ_l сравнивает момент времени $T_k^2 = T_r^2 + t_{\text{п}}$ где T_r^2 – момент времени исполнения конечной вершине q_r^2 ветви $\boldsymbol{H}_2$, а $t_{\text{п}}$ – период времени передачи результатов ее исполнения ресурсу R_p, исполняющему нить $\boldsymbol{H}_1$, с момента времени $T_{b-1}^1 = T_b^1 - t_p(O_b^1)$, где T_b^1 – момент времени исполнения вершины q_b^1 ветви q_r^2 $\boldsymbol{H}_1$, смежной с вершиной q_r^2 ветви $\boldsymbol{H}_2$, а $t_p(O_b^1)$ – период времени выполнения операции O_b^1, приписанной вершине q_b^1, ресурсом R_p (см. рис. 3.7). Если $T_r^2 + t_{\text{п}} > T_{b-1}^1$, то это означает, что результаты исполнения ветви $\boldsymbol{H}_2$, необходимые для выполнения операции O_b^1, ветви $\boldsymbol{H}_1$ поступят позже, чем результаты, получаемые в результате выполнения операции O_{b-1}^1, приписанной предыдущей вершине q_{b-1} ветви $\boldsymbol{H}_1$. В этом случае агент задания AZ_l должен скорректировать в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l,\boldsymbol{X}_l)$ планируемые моменты времени исполнения всех последующих вершин $q_b^1, q_{b+1}^1, ..., q_k^1$ ветви $\boldsymbol{H}_1$ путем их увеличения на величину $\Delta T = T_r^2 - T_{b-1}^1$.

Далее агент AZ_l исключает ветвь $\boldsymbol{H}_2$ из графа задания $\boldsymbol{G}_l^1(\boldsymbol{Q}_l^1,\boldsymbol{X}_l^1)$, в результате чего формируется новый граф $\boldsymbol{G}_l^2(\boldsymbol{Q}_l^2,\boldsymbol{X}_l^2) = \boldsymbol{G}_l^1(\boldsymbol{Q}_l^1,\boldsymbol{X}_l^1)/\boldsymbol{H}_2$, в котором агентом AZ_l вновь выделяется наиболее трудоемкая ветвь $\boldsymbol{H}_3$, которая выставляется на ДО для исполнения (см. рис. 3.8). Процесс продолжается до тех пор, пока все вершины графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l,\boldsymbol{X}_l)$ не будут закреплены за различными ресурсами сообщества $\boldsymbol{R}_l$ и им будут приписаны планируемые моменты времени их исполнения. По завершении данного процесса агент AZ_l задания Z_l отправляет агентам всех ресурсов, включенным в сообщество $\boldsymbol{R}_l$, команду на выполнение закрепленных за ними операций.

Ниже приведен формальный алгоритм работы агента AZ_l задания Z_l, отвечающий описанной выше процедуре.

Алгоритм 3.5

1. $j=1;\ \boldsymbol{G}_l^j(\boldsymbol{Q}_l^j,\boldsymbol{X}_l^j) = \boldsymbol{G}_l(\boldsymbol{Q}_l,\boldsymbol{X}_l);\ \boldsymbol{R}_l = \varnothing.$

2. Агентом AZ_l задания Z_l в графе $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ выделяется наиболее трудоемкая ветвь $\boldsymbol{H}_j =< q_1^j, q_2^j, ..., q_k^j >$, для которой значение $V_j = \sum_{i=1}^{k} v(O_i^j)$ максимально, где $v(O_i^j)$ – трудоемкость операции O_i^j, приписанной вершине q_i^j $(i = 1,2,...,k)$.

3. $m = 1$; $\boldsymbol{H}_j^m = \boldsymbol{H}_j$; $T_{\text{н}}^m = T_{\text{тек}}$.

4. Ветвь $\boldsymbol{H}_j^m$ выставляется агентом AZ_l на ДО для исполнения.

5. $r = 1$; $p = 0$; $E_p = \infty$.

6. Агент AZ_l получает «предложение» от агента AR_r ресурса R_r о возможности исполнении подветви $\boldsymbol{H}_{jr}^m =< q_1^j, q_2^j, ..., q_b^j > \subseteq \boldsymbol{H}_j^m$ $(b \le k)$, а также моменты времени $T_{\text{нг}}^m$ начала и $T_{\text{кг}}^m$ завершения ее исполнения.

7. Если $E_r = \dfrac{\sum_{i=1}^{b} v(O_i^j)}{T_{\text{кг}}^m - T_{\text{н}}^m} \ge E_p$, где $v(O_i^j)$ – трудоемкость операции вершины q_i^j $(i = 1,2,...,b)$, то перейти к 9, иначе

8. $E_p = E_r$; $p = r$.

9. $r = r + 1$; если $r \le N$, то перейти к 6, иначе

10. Агент AZ_l включает агента AR_p ресурса R_p в сообщество $\boldsymbol{R}_l$ по выполнению задания Z_l, и за ним закрепляется выполнение операций подветви $\boldsymbol{H}_{jp}^m$. В графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ вершинам подветви $\boldsymbol{H}_{jp}^m$ приписывается номер ресурса R_p, за которым закреплено их исполнение, а также планируемые моменты времени их исполнения, определяемые как

$$T_f^j = T_{\text{нр}}^m + \sum_{i=1}^{f} \frac{v(O_i)}{D_p(O_i)} \quad (f = 1,2,...,b).$$

11. $\boldsymbol{H}_j^{m+1} = \boldsymbol{H}_j^m \big/ \boldsymbol{H}_{jp}^m$; если $\boldsymbol{H}_j^{m+1} = \varnothing$, то перейти к 13, иначе

12. $m = m + 1$; $T_{\text{н}}^m = T_f^b + t_{\text{п}}$; перейти к 4.

13. Если $j = 1$ или $T_k^j + t_{\text{п}} \le T_b^{j-1} - t_{p-1}(O_b^{i-1})$, где T_k^j – момент времени исполнения конечной вершины q_k^j ветви $\boldsymbol{H}_j$, $t_{\text{п}}$ – период времени пересылки результатов ресурсу R_{p-1}, исполняющему смежную ветвь $\boldsymbol{H}_{j-1}$, T_b^{j-1} – момент времени исполнения операции O_b^{i-1} вершины $q_b^{j-1} \in \boldsymbol{H}_{i-1}$ инцидентной вершине $q_k^j \in \boldsymbol{H}_j$, $t_{p-1}(O_b^{i-1})$ – период времени

исполнения операции O_b^{i-1}, приписанной вершине q_b^{j-1} ресурсом R_{p-1}, то перейти к 15, иначе

14. В графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ вершинам $q_b^{j-1}, q_{b+1}^{j-1}, ..., q_r^{j-1}$, ветви $\boldsymbol{H}_{j-1}$ приписываются новые планируемые моменты времени их исполнения $T_i^{j-1} = T_i^{j-1} + \Delta T \ \ (i = b, b+1, ..., r)$, где $\Delta T = T_k^j - T_{b-1}^{j-1}$.

О корректировке моментов времени исполнения вершин $q_b^{j-1}, ..., q_r^{j-1}$ ветви $\boldsymbol{H}_{j-1}$ сообщается агенту AR_c, за которым закреплено их исполнение.

15. $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1}) = \boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) \big/ \boldsymbol{H}_j$, если $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1}) = \varnothing$, то перейти к 17, иначе

16. $j = j + 1$, перейти к 2.

17. Агент AZ_l отправляет агентам всех ресурсов, включенным в сообщество $\boldsymbol{R}_l$, команду на исполнение закрепленных за ними последовательностей операций.

18. Если от агента $AR_p \subseteq \boldsymbol{R}_l$ поступило сообщение о завершении исполнения закрепленной за ним подветви $\boldsymbol{H}_{jp}^m = < q_1^j, q_2^j, ..., q_b^j >$, то агент AZ_l сравнивает запланированный ранее момент времени $T_к^j$ завершения данной ветви, определяемый моментом времени, приписанном конечной вершине $q_b^1 \in \boldsymbol{H}_{ip}^m$, с текущим временем $T_{тек}$. Если $T_{тек} > T_к^j$, то планируемые моменты времени исполнения всех последующих вершин графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ увеличиваются на величину $\boldsymbol{\Delta} T = T_{тек} - T_{b-1}^1$.

19. Если конечная вершина q_b^j выполненной подветви $\boldsymbol{H}_{jp}^m$ является конечной вершиной q_k графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, то задание Z_l снимается с ДО.

Функции агента AR_p заключаются в поиске работы для «своего» ресурса R_p. С этой целью агент AR_p периодически опрашивает ДО и в случае обнаружения некоторой ветви $\boldsymbol{H}_j^m$ графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, выставленной на исполнение агентом AZ_l задания Z_l, делает попытку вхождения в сообщество $\boldsymbol{R}_l$ по его выполнению. Для этого агент AR_p выделяет в ветви $\boldsymbol{H}_j^m = < q_1^j, q_2^j, ..., q_k^j >$ подветвь $\boldsymbol{H}_{jp}^m = < q_1^j, q_2^j, ..., q_b^j >$ $(b \le k)$, операции, приписанные вершинам которой входят в множество $\boldsymbol{O}_p$, т. е. в множество операций, выполняемых «его» ресурсом R_p (см. рис. 3.6).

Кроме того, агент AR_p определяет момент времени $T_{нр}^m$, когда он может приступить к исполнению подветви $\boldsymbol{H}_{jp}^m$ (т. е. когда он освободится от исполнения ранее закрепленных за ним операций), а также момент

времени $T_{кр}^{m}$ окончания исполнения подветви $\boldsymbol{H}_{jp}^{m}$, определяемый согласно выражению (3.19).

Выделенная таким образом подветвь $\boldsymbol{H}_{jp}^{m}$ направляется агенту AZ_l задания Z_l в качестве «предложения» агента AR_p по участию в сообществе $\boldsymbol{R}_l$ по выполнению задания Z_l. Если «предложение» агента AR_p наилучшее среди всех поступивших, т. е. значение E_p, определяемое с помощью выражения (3.22), для нее максимально, то агент AZ_l задания Z_l направляет ему подтверждение о включении в сообществе $\boldsymbol{R}_l$ и закреплении подветви $\boldsymbol{H}_{jp}^{m}$ за ресурсом R_p.

Как только наступает запланированный момент $T_{нр}^{m}$ начала исполнения подветви $\boldsymbol{H}_{jp}^{m}$, закрепленной за ресурсом R_p, агент AR_p приступает к ее исполнению.

Описанному выше процессу отвечает следующий формальный алгоритм работы агента AR_p.

Алгоритм 3.6

1. Агент AR_p опрашивает ДО в поисках работы для «своего» ресурса R_p.

2. Если агент AR_p обнаружил на ДО выставленную агентом AZ_l для исполнения ветвь $\boldsymbol{H}_{j}^{m} =< q_1^j, q_2^j, ..., q_k^j >$ задания Z_l, то он выделяет в ветви $\boldsymbol{H}_{j}^{m}$ подветвь $\boldsymbol{H}_{jp}^{m} =< q_1^j, q_2^j, ..., q_b^j >$ $(b \le k)$, вершины которой удовлетворяют условию $O_i^j \subseteq \boldsymbol{O}_p$ $(j = 1,2,...,b)$, где O_i^j – операция, приписанная вершине q_i^j ветви $\boldsymbol{H}_{jp}^{m}$.

3. Агент AR_p определяет момент времени $T_{нр}^{m}$, когда он может приступить к исполнению ветви $\boldsymbol{H}_{j}^{m}$, а также момент времени $T_{кр}^{m}$ завершения ее исполнения, как

$$T_{кр}^{m} = T_{нр}^{m} + \sum_{i=1}^{b} \frac{v(O_i)}{D_p(O_i)}.$$

4. Сформированное таким образом «предложение» агента AR_p по участию в выполнении ветви $\boldsymbol{H}_{j}^{m}$ задания Z_l направляется агенту задания AZ_l.

5. Если агент AR_p получает от агента задания AZ_l подтверждение о его включении в сообщество $\boldsymbol{R}_l$, то он в момент времени $T_{нр}^{m}$ приступает к исполнению операций подветви $\boldsymbol{H}_{jp}^{m} =< q_1^j, q_2^j, ..., q_b^j >$.

6. $i = 1$

7. Если для выполнения очередной операции O_i^j, приписанной вершине q_i^j $(i = 1,2,...,b)$ ветви $\boldsymbol{H}_{jp}^m$, необходимы результаты операций, выполняемых другими ресурсами РС, то агент AR_j проверяет их наличие. Если необходимые результаты еще не получены, то агент AR_j переходит в режим ожидания.

8. После поступления всех необходимых результатов операций, выполняемых другими ресурсами РС, агент AR_p инициирует выполнение операции O_i^j $(i = 1,2,...,r)$ с помощью локального устройства управления $УУ_P$ «своего» ресурса R_p.

9. $i = i + 1$, если $i < b$, то перейти к 7, иначе

10. Агент AR_p сообщает агенту AZ_l задания Z_l о завершении выполнения операций ветви $\boldsymbol{H}_{jp}^m$.

11. Переход к 1.

Глава 4.
Примеры самоорганизующихся распределенных систем

4.1. Самоорганизующееся безлюдное роботизированное производство

Ускорение научно-технического прогресса приводит к усилению конкурентной борьбы на рынке высокотехнологичной продукции. Побеждает тот, кто сможет быстрее довести свою идею до демонстрационного или опытного образца, способного заинтересовать рыночных инвесторов. В этом плане все большую актуальность приобретает проблема создания безлюдных роботизированных производств (БРП), способных в кратчайшее сроки изготавливать разнообразные опытные образцы и единичные изделия по требованиям заказчика [26,27]. Такое БРП должно представлять собой распределенную систему, включающую в свой состав широкий набор роботизированных обрабатывающих центров (РОЦ) различной функциональной направленности, каждый из которых обладает своим локальным устройством управления (УУ) (рис. 4.1).

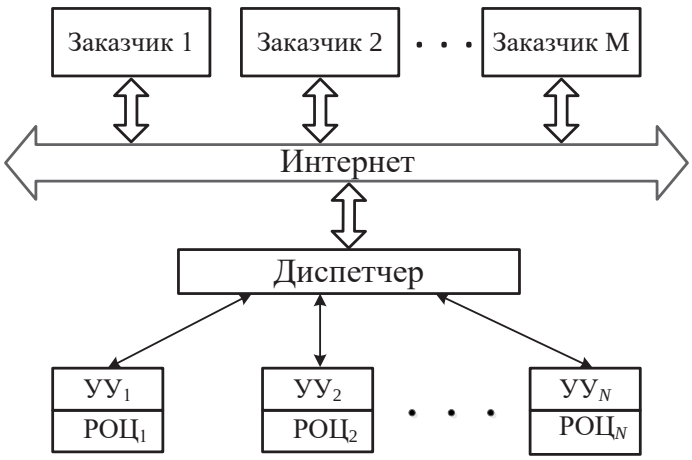


Рис. 4.1. Безлюдное роботизированное производство

Заказчики формируют и направляют через Интернет задания на свои изделия, которые поступают на диспетчер БРП (см. рис. 4.1), функции которого заключаются в формировании сообществ ресурсов БРП и временных графиков их взаимодействия для изготовления этих изделий. После того как такое сообщество сформировано, отдельные РОЦ, входящие в его состав, приступают к выполнению возложенных на них

операций в соответствии с временным графиком с помощью своих локальных систем управления.

Формально задачу самоорганизации БРП при выполнении потока заданий, поступающих от различных Заказчиков в произвольные моменты времени, можно представить следующим образом.

Будем считать, что в состав БРП входит некоторое множество РОЦ $R_1, R_2, ...R_N$, а также два склада – склад комплектующих и заготовок и склад готовых изделий (рис. 4.2). Все РОЦ и склады объединены общей транспортной линией, посредством которой комплектующие, заготовки и изделия могут передаваться между складами и РОЦ, а также между отдельными РОЦ (рис. 4.2).

Предположим, что каждый РОЦ R_i может выполнять некоторое множество операций $\boldsymbol{O}_i =< O_1^i, O_2^i, \dots, O_K^i > (i = 1,2, \dots, N)$, причем в общем случае $\boldsymbol{O}_i \neq \boldsymbol{O}_j \ \ (j = 1,2, \dots, i+1, \dots, N)$. Будем считать, что РОЦ R_p выполняет операцию $O_j^i \ \ (j = 1,2, \dots, L)$ за время $t_p(O_j^i)$, причем время выполнения идентичных операций различными РОЦ одинаково. Кроме того положим, что время транспортировки комплектующих и заготовок между отдельными РОЦ, а также между РОЦ и складами составляет $t_{\Pi}(S)$, где S – длина транспортной линии между ними.

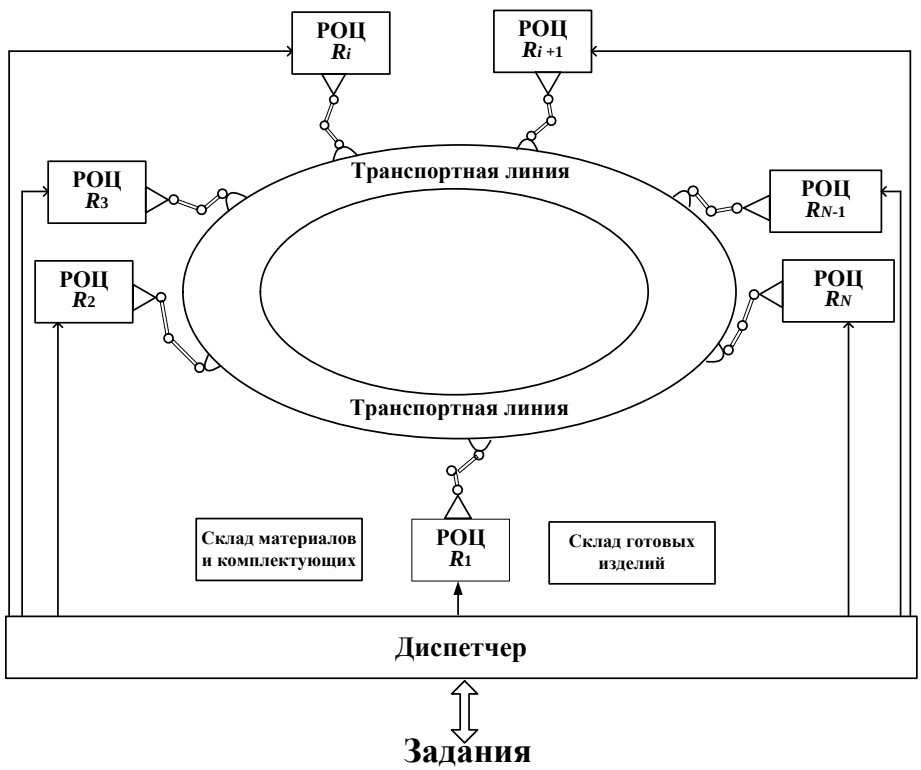


Рис. 4.2. Структура БРП

Будем считать, что в БРП в случайные моменты времени через Интернет от различных Заказчиков поступает некоторое множество (поток) различных заданий на изготовление изделий $\boldsymbol{Z} =< Z_1, Z_2, ..., Z_L >$, причем для каждого задания Z_l Заказчиком устанавливается момент времени $T^l_{\max}$, к которому он желает получить свое изделие. При этом задание на изготовление некоторого изделия $Z_l \in \boldsymbol{Z}$ представляется в виде ацикличного графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ (рис. 4.3), каждой вершине $q_j \in \boldsymbol{Q}_l$ которого приписана некоторая операция O_j, принадлежащая множеству операций $\boldsymbol{O} = \cup_{i=1}^{N} \boldsymbol{O}_i$, выполняемых различными РОЦ, причем если две вершины графа q_j и q_{j+1} соединены дугой $x(q_j, q_{j+1})$, то это означает, что операция O_{j+1}, приписанная вершине q_{j+1} должна выполняться по завершении операции O_j, приписанной вершине q_j. Входные вершины графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ определяют операции по доставке исходных комплектующих и заготовок, необходимых для изготовления изделия, со склада, а конечная вершина определяет операцию по размещению на складе готовой продукции конечного изделия, получаемого в результате выполнения всей программы его изготовления.

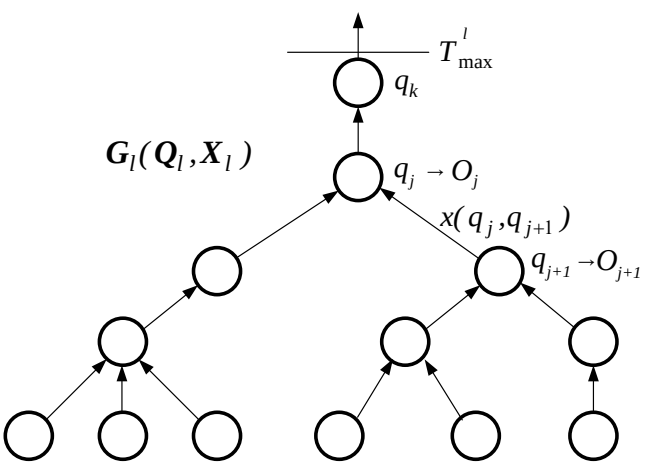


Рис. 4.3. Граф $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l

После того, как задание Z_l на изготовления изделия формализовано в виде графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, его дескриптор передается через Интернет в диспетчер БРП. Цель работы диспетчера заключаются в формирования сообществ РОЦ и построении временных графиков их взаимодействия для изготовления всех изделий $\boldsymbol{Z} = < Z_1, Z_2, \ldots, Z_L >$ таким образом, чтобы средняя задержка ΔT выполнения заданий Заказчиков относительно

установленного ими моментами времени $T_{\max}^{l}$ ($l = 1, 2, \ldots, L$) была минимальна.

Очевидно, что БРП, отвечающая сформированной выше задаче, относится к классу гетерогенных СРС первого типа (см. подраздел 1.1), что позволяет взять за основу при ее организации алгоритм мультиагентного диспетчирования ресурсов, предложенный в подразделе 2.3.

Будем считать, что каждый РОЦ R_i ($i = 1, 2, \ldots, N$), входящий в состав БРП, обладает своим программным агентом AR_i, представляющим его «интересы» в процессе диспетчирования, а взаимодействие агентов друг с другом осуществляется посредством некоторого информационного канала связи (рис. 4.4). Взаимодействие агентов с Заказчиками осуществляется посредством некоторого пассивного узла, подключенного к Интернет и выполняющего функции «доски объявлений» (ДО), на которой Заказчики размещают свои задания (рис. 4.4).

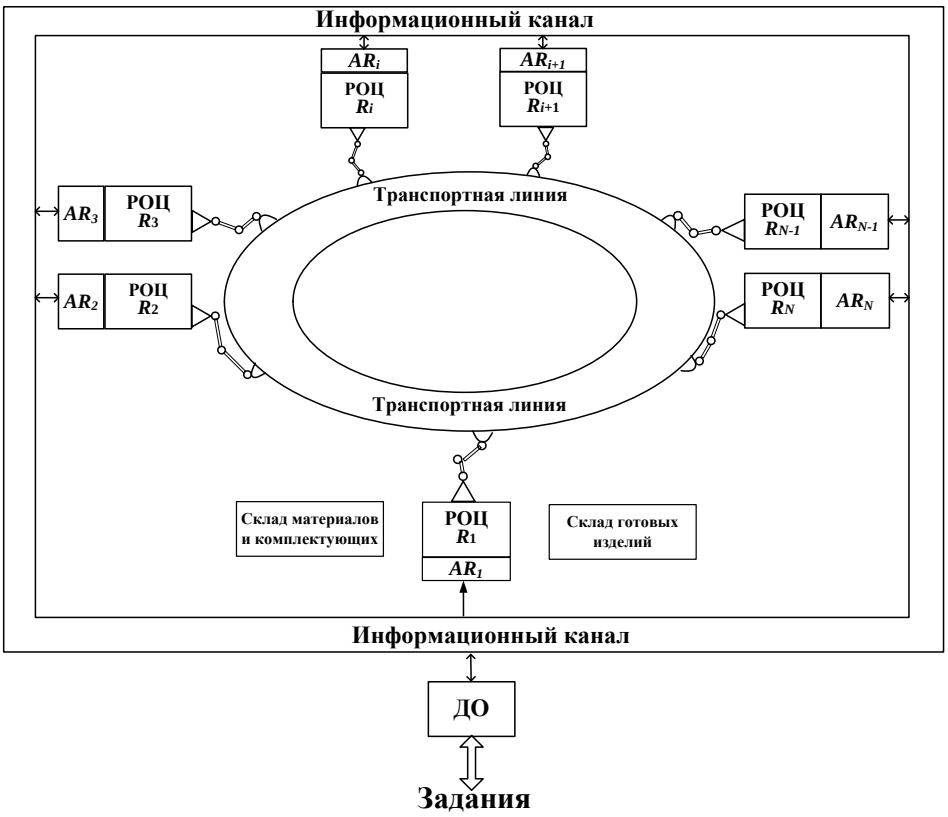


Рис. 4.4. Структура самоорганизующейся БРП с мультиагентным диспетчером

При этом дескриптор задания Z_l, размещаемого Заказчиком на ДО, должен содержать:

– граф $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания;

– список вершин множества $\boldsymbol{Q}_l$ и приписанных им операций;

– момент времени T^l_{max}, к которому потребитель желает получить готовое изделие.

Прежде чем приступить к разработке алгоритма мультиагентного диспетчирования БРП, конкретизируем понятие ветви задания для данного случая. Под ветвью будем понимать некоторую последовательность вершин $\boldsymbol{H}_f =< q^f_1, q^f_2, ..., q^f_k >$ графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания $Z_l \in \boldsymbol{Z}$, в которой вершины q^f_j и q^f_{j+1} $(j = 1,2,...,k-1)$ соединены дугой $x(q^f_j, q^f_{j+1})$ (рис. 4.5). Иными словами, ветвь определяет некоторый набор операций задания Z_l, которые должны выполняться последовательно.

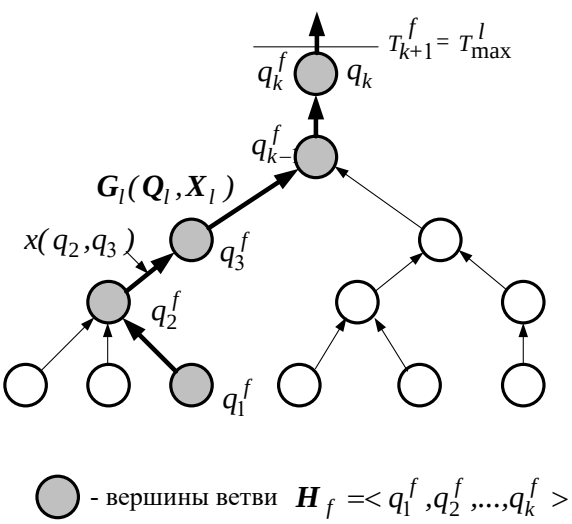


Рис. 4.5. Выделение ветви $\boldsymbol{H}_f$ в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l

При этом под длиной t_f ветви $\boldsymbol{H}_f$ будем понимать суммарное время, затрачиваемое на выполнение приписанных ее вершинам операций, определяемое как:

$$t_f = \sum_{i=1}^{k} \left(t_p\left(O^f_i\right) + t_{\text{п}}\left(S_{p,c}\right) \right),$$

где $t_p\left(O^f_i\right)$ – время, затрачиваемое РОЦ $R_p \in \boldsymbol{R}$ на выполнение операции O^f_i, приписанной вершине $q^f_i \in \boldsymbol{H}_f$ $(i = 1,2,...,k)$;

$t_{\text{п}}(S_{p,c})$ – время, затрачиваемое на транспортировку изделия от РОЦ $R_p \in \boldsymbol{R}$, выполняющего операцию O_i^f, РОЦ $R_c \in \boldsymbol{R}$, выполняющему следующую по очереди операцию O_{i+1}^f ветви $\boldsymbol{H}_f$;

$S_{p,c}$ – длина транспортной линии между РОЦ R_p и РОЦ R_c.

Если операции O_i^f и O_{i+1}^f выполняются одним и тем же РОЦ R_p, то $t_{\text{п}}(S_{p,p}) = 0$. Соответственно, если вся ветвь $\boldsymbol{H}_f$ выполняется одним и тем же ресурсом R_p, то ее длина будет составлять

$$t_f = \sum_{i=1}^{k} t_p\left(O_i^f\right) + t_{\text{п}}\left(S_{k,k+1}\right), \qquad (4.1)$$

где $t_{\text{п}}\left(S_{k,k+1}\right)$ – время транспортировки заготовки изделия между РОЦ R_p и РОЦ R_c, выполняющим смежную ветвь задания Z_l, или складом готовой продукции, если конечная вершина ветви $\boldsymbol{H}_f$ – это конечная вершина графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, т.е. $q_k^f = q_k$.

При этом если известен требуемый момент времени T_{k+1}^f исполнения всей ветви $\boldsymbol{H}_f$ (см. рис. 4.5), то можно определить допустимые моменты времени T_d^f начала выполнения всех операций O_d^f, приписанных ее вершинам $q_d^f \in \boldsymbol{H}_f$ $(d = 1,2,\ldots,k)$, при которых РОЦ R_p сможет выполнить всю ветвь $\boldsymbol{H}_f$ к требуемому моменту времени T_{k+1}^f, как

$$T_d^f = T_{k+1}^f - \sum_{i=d}^{k} t_p\left(O_i^f\right) - t_{\text{п}}\left(S_{k,k+1}\right) \ (d = 1,2,\ldots,k-1). \ (4.2)$$

Если при этом величина

$$\Delta T_f^l = T_{\text{тек}} - T_1^f > 0,$$

то это означает, что РОЦ R_p может выполнить ветвь $\boldsymbol{H}_f$ с задержкой ΔT^f относительно требуемого времени T_{k+1}^f.

Исходя из этих определений, можно предложить следующую процедуру мультиагентного диспетчирования работы БРП при выполнении потока заданий.

Заказчик формирует свое задание $Z_l \in \boldsymbol{Z}$ в виде графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ и определяет требуемый момент времени $T_{\max}^l$, к которому ее решение должно быть получено. Дескриптор представленного таким образом задания $Z_l \in \boldsymbol{Z}$ размещается на доске объявлений (см. рис. 4.4).

Агенты свободных РОЦ, которые не задействованы в выполнении каких-либо других заданий, обращаются к ДО в поисках работы. Если агент AR_p свободного РОЦ $R_p \in \boldsymbol{R}$ обнаруживает на ДО дескриптор

задания Z_l, содержащий граф $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$, то он делает попытку войти в сообщество $\boldsymbol{R}_l$ по его исполнению.

Поскольку, как мы приняли выше, каждый РОЦ имеет некоторую специализацию (т. е. может выполнять некоторый ограниченный набор операций), то в общем случае может оказаться, что РОЦ R_p способен выполнять далеко не все операции, приписанные вершинам графа $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ задания Z_l. Поэтому в графе $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ необходимо предварительно выделить подграф $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp}, \boldsymbol{X}_l^{jp})$, вершинам которого приписаны операции множества $\boldsymbol{O}_p$, выполняемые РОЦ R_p (рис. 4.6).

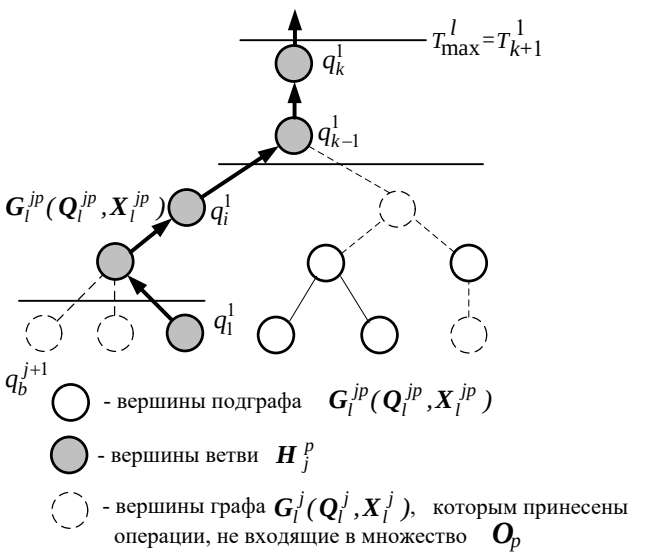


Рис. 4.6. Выделение подграфа $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp}, \boldsymbol{X}_l^{jp})$ в графе $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ задания Z_l

После этого необходимо проанализировать, есть ли в графе $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp}, \boldsymbol{X}_l^{jp})$ вершины, для которых установлено требуемое время их исполнения. Отметим, что изначально, в момент размещения дескриптора задания Z_l на ДО, требуемое время исполнения $T^l_{\max}$ приписано только конечной вершине q_k графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ (см. рис. 4.5). Если таковых вершин нет, то это говорит о том, что агент AR_p пока что не может вступить в сообщество $\boldsymbol{R}_l$ по выполнению задания Z_l и поэтому он вновь переходит к режиму опроса ДО с целью поиска других заданий.

В противном случае агент AR_p выделяет в графе $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp}, \boldsymbol{X}_l^{jp})$ с помощью выражения (2.1) наиболее длинную ветвь $\boldsymbol{H}_j^p = < q_1^{jp}, q_2^{jp}, ..., q_k^{jp} >$,

конечной вершине q_k^{jp} которой приписан требуемый момент времени исполнения T_{k+1}^{jp}, определяет момент времени T_1^{jp} начала исполнения ее первой операции O_1^{jp}, как

$$T_1^{jp} = T_{k+1}^{j} - \left(\sum_{i=1}^{k} t_p(O_i^{jp}) + t_{\text{п}}(S_{k,k+1}) \right), \text{ где } t_p(O_i^{jp}) = \frac{v(O_i^{jp})}{D_p(O_i^{jp})}.$$

Если $T_{\text{тек}} > T_1^{jp}$, то это означает, что ресурс R_p может выполнить данную ветвь $\boldsymbol{H}_j^p$ с задержкой $\Delta T_j^l = T_{\text{тек}} - T_1^{jp}$ относительно требуемого момента времени T_{k+1}^{jp}. Поскольку выше мы приняли, что все ресурсы множества $\boldsymbol{R}$ выполняют идентичные операции за одинаковое время, то это также означает, что никакой другой ресурс $R_c \in \boldsymbol{R}$ также не сможет выполнить данную ветвь $\boldsymbol{H}_j^p$ быстрее.

Далее агент AR_p входит в состав сообщества $\boldsymbol{R}_l$ по выполнению задания Z_l и выполняет процедуру модификации графа задания на ДО: исключает ветвь $\boldsymbol{H}_j^p$ из графа задания $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$, в результате чего формируется новый граф задания $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1}) = \boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) / \boldsymbol{H}_j^p$, а каждой вершине модифицированного графа $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1})$, инцидентной вершине ветви $\boldsymbol{H}_j^p$, приписывает требуемый момент времени ее исполнения, определяемый согласно выражения (2.3), а также номер РОЦ R_p, которому должны быть переданы результаты исполнения приписанной ей операции.

Далее аналогичный выбор делает следующий по нумерации агент AR_{p+1} свободного ресурса R_{p+1}.

Процесс распределения ветвей задания Z_l между агентами РОЦ заканчивается, когда после очередной модификации граф задания $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1})$ на ДО становится пустым, что означает, что все его ветви распределены между ресурсами множества $\boldsymbol{R}_l$.

После того, как некоторый агент AR_p выбрал для исполнения ветвь $\boldsymbol{H}_j^p = < q_1^{jp}, q_2^{jp}, ..., q_k^{jp} >$, он приступает к исполнению операций, приписанных ее вершинам с помощью «своего» РОЦ.

После успешного выполнения всех операций ветви $\boldsymbol{H}_j^p$ агент AR_p вновь переходит в режим опроса ДО с целью вхождения в новое сообщество по выполнению следующего задания.

Процесс выполнения задания Z_l продолжается до тех пор, пока не окажется, что список агентов-участников сообщества $\boldsymbol{R}_l$ по его

выполнению пуст. Это означает, что все ветви задания успешно выполнены. После этого задание снимается с ДО, а Заказчику отправляется сообщение об успешном выполнении его задания.

Исходя из приведенных выше соображений можно предложить следующий алгоритм работы агента AR_p РОЦ R_p при реализации процедуры мультиагентного диспетчирования БРП. При этом, как и в предыдущих случаях, полагаем, что при размещении Заказчиком задании Z_l на ДО счетчик итераций $j = 1$; $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) = \boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ и $\Delta T^l = 0$.

Алгоритм 4.1

1. Агент AR_p свободного РОЦ R_p опрашивает ДО в порядке своей очереди.

2. При обнаружении на ДО задания Z_l агент AR_p считывает его дескриптор, содержащий граф задания $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ и требуемый момент времени $T_{\max}^l$ его исполнения.

3. В графе $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ выделяется подграф $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp}, \boldsymbol{X}_l^{jp})$, вершинам которого приписаны операции множества $\boldsymbol{O}_p$, выполняемые ресурсом R_p.

4. Если $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp}, \boldsymbol{X}_l^{jp}) = \varnothing$ или подграф $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp}, \boldsymbol{X}_l^{jp})$ не содержит ни одной вершины, которой приписано требуемый момент времени T_{k+1}^j ее исполнения, то перейти к 1, иначе

5. Агент AR_p выделяет в подграфе $\boldsymbol{G}_l^{jp}(\boldsymbol{Q}_l^{jp}, \boldsymbol{X}_l^{jp})$ наиболее длинную ветвь $\boldsymbol{H}_j^p = < q_1^{jp}, q_2^{jp}, ..., q_k^{jp} >$, конечной вершине которой приписан требуемый момент времени ее исполнения T_{k+1}^j (в момент размещения задания Z_i на ДО требуемый момент времени исполнения $T_{k+1}^j = T_{max}^l$ приписано только конечной вершине q_k графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$), и определяет требуемый момент времени T_1^{jp}, когда необходимо начать ее выполнение как:

$$T_1^{jp} = T_{k+1}^j - \left(\sum_{i=1}^{k} t_p(O_1^{jp}) + t_{\text{п}}(S_{k,k+1}) \right).$$

6. Если $T_1^{jp} \geq T_{\text{тек}}$, где $T_{\text{тек}}$ – текущий момент времени, то $\Delta T^l = \Delta T^l + (T_{\text{тек}} - T_1^{jp})$.

7. Агент AR_p принимает на себя исполнение ветви $\boldsymbol{H}_j^p$, записывает в список участников сообщества $\boldsymbol{R}_l$ свой номер p и модифицирует граф, а именно:

– из графа задания $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$ исключаются вершины ветви $\boldsymbol{H}_j^p$, в результате чего формируется новый граф задания $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1}) = \boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) / \boldsymbol{H}_j^p$;

– каждой вершине $q_b^{j+1,p}$ модифицированного графа $\boldsymbol{G}_l^{j+1}(\boldsymbol{Q}_l^{j+1}, \boldsymbol{X}_l^{j+1})$, инцидентной вершине q_d^{jp} ветви $\boldsymbol{H}_j^p$, приписывается требуемый момент времени ее исполнения, определяемый как (см. рис. 2.11)

$$T_{b+1}^{j+1} = T_{k+1}^j - \left(\sum_{i=d}^{k} t_p\left(O_i^{jp}\right) + t_{\text{п}}\left(S_{k,k+1}\right)\right), \text{ где } t_p(O_i^{jp}) = \frac{v(O_i^{jp})}{D_p(O_i^{jp})}$$

а также номер ресурса R_p, которому необходимо передать результаты исполнения операции, приписанной вершине $q_b^{j+1,p}$;

– если вершина q_{k+1}^{jp} инцидента вершине $q_b^{j-1,p}$ ветви $\boldsymbol{H}_{j-1}^p$, то моменты времени начала исполнения, приписанные вершинам $q_b^{j-1,p}, b_{b+1}^{j-1,p}, \ldots, q_L^{j-1,p}$ ветви $\boldsymbol{H}_{j-1}^p$, увеличиваются на величину $(T_{\text{тек}} - T_1^{jp})$.

8. Счетчик итераций на ДО увеличивается на единицу, т. е. $j = j + 1$. Модифицированный граф $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j)$, а также новое значение ΔT^l размещаются на ДО. Если $\boldsymbol{G}_l^j(\boldsymbol{Q}_l^j, \boldsymbol{X}_l^j) \neq \varnothing$, то процесс распределения операций задания Z_l заканчивается, а Заказчику сообщается ожидаемое время задержки ΔT^l выполнения его задания.

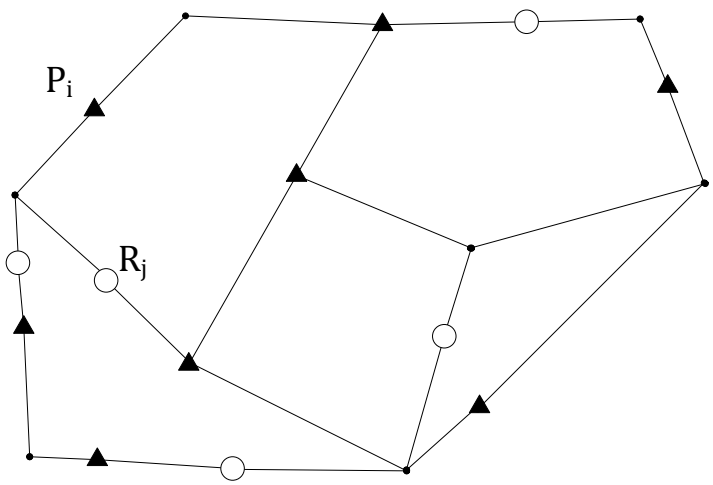
9. РОЦ R_p приступает к выполнению последовательности операций, приписанных вершинам ветви $\boldsymbol{H}_j$ согласно установленному временному графику (т. е. приписанным вершинам ветви $\boldsymbol{H}_j$ моментам времени начала их исполнения).

10. После завершения выполнения всех операций ветви $\boldsymbol{H}_j$ переход к 1.

4.2. Самоорганизующиеся транспортно-логистические системы

Еще одним примером самоорганизующихся распределенных систем, включающих в свой состав множество гетерогенных ресурсов, могут служить транспортно-логистические системы [28]. В общем виде задачу самоорганизации транспортно-логистической системы можно сформировать следующим образом.

Предположим, что существует некоторая дорожная сеть, на которой расположено D стационарных пунктов загрузки/выгрузки грузов $P_з^i$ $(i = 1,2, \dots, D)$ (рис. 4.7).



▲ – пункты загрузки/выгрузки $P_з^i$ $(i = 1,2, \dots, D)$;
○ – транспортные ресурсы R_j $(j = 1,2, \dots, N)$.

Рис. 4.7. Дорожная сеть

Одновременно на той же дорожной сети находятся N мобильных транспортных ресурсов R_j $(j = 1,2, \dots, N)$ (например, грузовых автомобилей) (см. рис. 4.7), которые служат для транспортировки грузов между пунктами загрузки/разгрузки. При этом каждый ресурс R_j описывается набором следующих параметров.

1. $P_т^j$ – точка текущего положения ресурса R_j на дорожной сети.
2. B_{max}^j – максимальная грузоподъемность.
3. $C(B^j)$ – максимально допустимая скорость движения при транспортировке груза с весом B^j.

Будем считать, что ресурс R_j $(j = 1,2, \dots, N)$ может выполнять операции трех типов.

1. $O_1(P_{\text{Т}}^j, P_3^i)$ – перемещение из точки текущего положения $P_{\text{Т}}^j$ в точку загрузки/выгрузки P_3^i.

2. $O_2(B^j)$ – загрузка груза весом B^j.

3. $O_3(B^j)$ – выгрузка груза весом B^j.

При этом трудоемкость $v(O_1)$ операции типа O_1 пропорциональна расстоянию $L(P_{\text{Т}}^j, P_3^i)$ между точками $P_{\text{Т}}^j$ и P_3^i по дорожной сети, т. е.

$$v(O_1) = K_1 \cdot L(P_{\text{Т}}^j, P_3^i),$$

где K_1 – некоторый коэффициент, а трудоемкости операций типа O_2 и O_3 пропорциональны весу загружаемого или выгружаемого груза, т.е.

$$v(O_2) = K_2 \cdot B^j \text{ и } v(O_3) = K_3 \cdot B^j ,$$

где K_2 и K_3 – некоторые коэффициенты.

Под производительностью $D_j(O_1)$ ресурса R_j $(j = 1,2,\dots,N)$ при выполнении операции типа O_1 будем понимать его максимально допустимую скорость движения C с грузом B^j, т. е.

$$D_j(O_1) = C(B^j),$$

а производительность $D_j(O_2)$ и $D_j(O_3)$ при выполнении операций типа O_2 и O_3 будем считать постоянной и равной D_B.

Предположим, что существует L Заказчиков, заинтересованных в транспортировке своих грузов между пунктами загрузки/выгрузки. При этом каждый Заказчик формирует свое задание Z_l $(l = 1,2,\dots,L)$ в виде графа (дерева) $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, состоящего из некоторого множества непересекающихся ветвей, каждая из которых описывает некоторое подзадание задания Z_l (рис. 4.8).

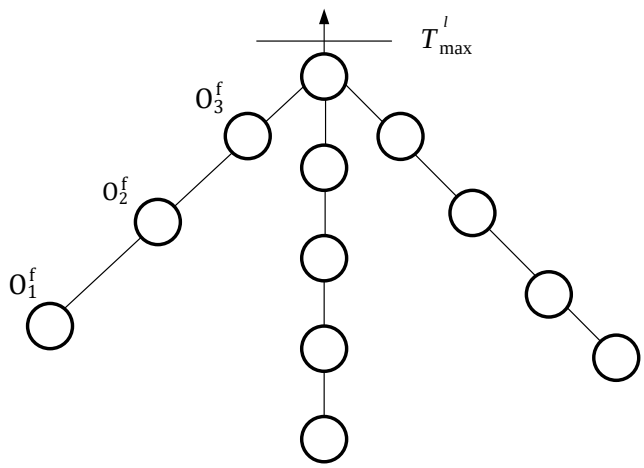


Рис. 4.8. Граф $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l

Например, подзаданию вида «забрать груз весом B в пункт P_3^1, доставить часть груза весом B^s в пункт P_3^2, а оставшуюся часть B^j в пункт P_3^3» будет

соответствовать ветвь $\boldsymbol{H}_f = < q_1^f, q_2^f, q_3^f, q_4^f, q_5^f, q_6^j >$ графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, вершинам которой приписаны следующие операции:

$O_1^f = O_1(P_{\text{T}}^j, P_3^1)$ – операция перемещения из точки текущего положения P_{T}^j в точку пункта загрузки/выгрузки P_3^1;

$O_2^f = O_2(B)$ – операция загрузки груза весом B;

$O_3^f = O_1(P_3^1, P_3^2)$ – операция перемещения груза из пункта P_3^1 в пункт загрузки/выгрузки P_3^2;

$O_4^f = O_3(B^s)$ – операция выгрузки груза весом $B^s < B$;

$O_5^f = O_1(P_3^2, P_3^3)$ – операция перемещения груза из пункта P_3^2 в пункт загрузки/выгрузки P_3^3;

$O_6^f = O_3(B^j)$ – операция выгрузки груза весом B^j $(B^j = B - B^s)$.

Заказчик также устанавливает момент времени $T_{\max}^l$, к которому он желает, чтобы его задание Z_l было выполнено (см. рис. 4.8), а также премию S^l за его успешное выполнение, причем величина премии S^l определяется следующим образом:

$$S^l = \begin{cases} S_{max}^l \cdot \dfrac{\Delta T_{max}^l - \Delta T^l}{\Delta T_{max}^l}, \text{если } \Delta T^l \le \Delta T_{max}^l, \\ 0, \text{если } \Delta T^l > \Delta T_{max}^l, \end{cases}$$

где S_{max}^l – максимальный размер премии, получаемый в случае успешного выполнения задания к установленному моменту времени T_{max}^l;

ΔT^l – задержка выполнения задания Z_l относительно требуемого момента времени T_{max}^l;

ΔT_{max}^l – максимально допустимая задержка.

При этом будем считать, что количество премиальных баллов S_j^l, начисляемое ресурсу R_j за успешное выполнение задания Z_l, определяется следующим образом:

$$S_j^l = \frac{S^l \cdot V_j}{V},$$

где $V_j = \sum_{i=1}^{k} v(O_i^j)$ – суммарная трудоемкость операций задания Z_l, выполненных ресурсом R_j;

$V = \sum_{i=1}^{M} v(O_i)$ – суммарная трудоемкость всех операций задания Z_l;

$M = |\boldsymbol{Q}_l|$– число вершин в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l.

При этом будем считать, что каждый ресурс R_j $(j = 1,2,\dots,N)$ заинтересован в получении максимального числа премиальных баллов за свою работу.

В сформулированной выше постановке задача самоорганизации в транспортно-логистической системе заключается в таком распределении ветвей (подзаданий) графа задания $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ $(l = 1,2,\dots,L)$ между

транспортными ресурсами R_j $(j = 1,2,\dots,N)$, при котором среднее время задержки $\Delta T = \frac{\sum_{l=1}^{M} \Delta T^l}{M}$ выполнения заданий множества $\boldsymbol{Z}$ будет минимально.

Описанная выше СРС относится к классу гетерогенных СРС первого типа, в которой предусмотрено премирование ресурсов за выполненную работу, и поэтому в основу ее функционирования можно положить алгоритм мультиагентного диспетчирования ресурсов, предложенный в подразделе 3.1.

Для этого каждый транспортный ресурс R_j (j = 1, 2, …, N) должен обладать своим программным агентом AR_j, имеющим доступ к доске объявлений (ДО), на которой Заказчики размещают дескрипторы своих заданий Z_l.

Агент AR_j свободного ресурса R_j должен в порядке своей очереди обращаться на ДО в поисках работы для «своего» ресурса. Если на ДО одновременно находится несколько заданий $\boldsymbol{Z} =< Z_1, Z_2, \dots, Z_L >$, то среди них агент AR_j должен в первую очередь выбрать то задание Z_l, для которого средняя стоимость единицы трудоемкости максимальна (см. подраздел 3.1), т. е. максимальна величина

$$S_{\text{ср}}^{l} = \frac{S_{max}^{l}}{V_l},$$

где $V_l = \sum_{i=1}^{M} v(O_i)$ – суммарная трудоемкость всех операций задания Z_l.

После того, как агент AR_j выбрал задание Z_l среди множества заданий $\boldsymbol{Z}$, размещенных на ДО, он выделяет ветвь этого задания, выполнение которой позволит ему получить максимальное число премиальных баллов. В подразделе 3.1 показано, что такая ветвь должна отвечать двум условиям: во-первых, ее трудоемкость должна быть как можно большей, а во-вторых, возникающая при ее исполнении задержка – как можно меньшей [29].

Формально данные требования (см. подраздел 3.1) можно сформулировать в следующем виде.

1. Ветви $\boldsymbol{H}_f$ должен быть приписан момент времени ее исполнения T_{k+1}^{f}. Здесь следует отметить, что поскольку граф $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l представляет собой дерево, то всем его ветвям будет приписан один и тот же требуемый момент времени их исполнения T_{max}^{l} (см. рис. 4.8). Поэтому $T_{k+1}^{f} = T_{max}^{l}$ для всех ветвей задания Z_l

2. Количество S_p^f потенциально получаемых премиальных баллов за выполнение ресурсом R_j ветви $\boldsymbol{H}_f$ должно быть максимально. При этом величину S_p^f можно оценить с помощью следующего выражения (см. подраздел 3.1).

$$S_p^f = \begin{cases} \frac{V_l^f}{V_l} \cdot S_{max}^l \cdot \frac{\Delta T_{max}^l - \Delta T_j^f}{\Delta T_{max}^l}, \text{если } \Delta T_j^f < \Delta T^l, \\ 0, \text{если } \Delta T_j^f \geq \Delta T^l, \end{cases} \quad (4.1)$$

где S_p^f – количество премиальных баллов, получаемых агентом AR_j за выполнение операций ветви $\boldsymbol{H_f}$ задания Z_l;

ΔT_j^f – задержка выполнения ветви $\boldsymbol{H_f}$ задания Z_l ресурсом R_j относительно установленного момента времени $T_{k+1}^f = T_{max}^l$;

ΔT_{max}^l – максимальная допустимая задержка, установленная Заказчиком задания Z_l;

V_l^f – суммарная трудоемкость операций ветви $\boldsymbol{H_f}$;

V_l – суммарная трудоемкость всех операций задания Z_l.

При этом значение ΔT_j^f определяется следующим образом

$$\Delta T_j^f = \begin{cases} T_{\text{тек}} - T_H^f, \text{если } T_{\text{тек}} > T_1^f \\ 0, \text{если } T_{\text{тек}} \leq T_1^f, \end{cases}, \quad (4.2)$$

где $T_1^f = T_{max}^l - \sum_{i=1}^{k} \frac{v(O_i^f)}{D_j(O_i^f)}$;

$v(O_i^f)$ – трудоемкость выполнения i-й операции O_i^f ветви $\boldsymbol{H}_f$ (i = 1, 2, …, k);

$D_j(O_i^f)$ – производительность ресурса R_j при выполнении операции O_i^f.

Удовлетворяющая данным условиям ветвь задания Z_l закрепляется за ресурсом R_j, и он приступает к ее исполнению. Кроме того, данная ветвь $\boldsymbol{H}_f$ исключается из графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l, поле чего модифицированный таким образом граф задания, если он еще не пустой, вновь выставляется на ДО для распределения между другими свободными ресурсами множества $\boldsymbol{R}$.

Учитывая приведенные выше соображения, алгоритм работы агента AR_j транспортного ресурса R_j ($j = 1,2, \dots, N$) можно представить в следующем виде.

Алгоритм 4.2

1. Агент AR_j свободного транспортного ресурса R_j считывает в порядке своей очереди дескрипторы заданий $\boldsymbol{Z} = < Z_1, Z_2, \dots, Z_L >$, размещенных на ДО, каждый из которых содержат информацию о графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задание Z_l и количестве премиальных баллов S_{max}^l, установленных Заказчиком за его успешное выполнение к моменту времени T_{max}^l.

2. Если $Z \neq 0$, то перейти к 1, иначе

3. Агент AR_j среди множества заданий $\boldsymbol{Z}$ выбирает то задание Z_l, для которого средняя стоимость трудоемкости максимальна, т. е. максимальна величина

$$S_{\text{ср}}^{l} = \frac{S_{max}^{l}}{\sum_{i=1}^{M} v(O_i)},$$

где $M = |\boldsymbol{Q}_l|$.

4. Агент AR_j выделяет в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ветвь $\boldsymbol{H}_f$, для которой значение величины S_p^f (4.1) максимально.

5. Если $S_p^f = 0$, то $\boldsymbol{Z} = \boldsymbol{Z}/Z_l$ перейти к 1, иначе

6. Агент AR_j входит в состав сообщества $\boldsymbol{R}_l$ по выполнении задания Z_l, после чего модифицирует граф $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l на ДО, исключая из него ветвь $\boldsymbol{H}_f$.

7. Ресурс R_j приступает к исполнению операций ветви $\boldsymbol{H}_f$. По завершении выполнения всех операций ветви $\boldsymbol{H}_f$ агент AR_j выходит из сообщества $\boldsymbol{R}_l$ и ему начисляются премиальные баллы в размере

$$S_j^l = \frac{S^l \cdot V_l^j}{V_l},$$

причем $S^l = \begin{cases} S_{max}^{l} \cdot \frac{\Delta T_{max}^{l} - \Delta T^l}{\Delta T_{max}^{l}}, \text{если } \Delta T^l < \Delta T_{max}^{l} \\ 0, \text{если } \Delta T^l \geq \Delta T_{max}^{l}, \end{cases}$

где V_l^j – суммарная трудоемкость операций задания Z_l, выполненных ресурсом R_j;

V_l – суммарная трудоемкость всех операций задания Z_l;

ΔT^l – задержка выполнения задания Z_l относительно установленного Заказчиком момента времени $T_{\max}^{l}$.

4.3. Самоорганизующиеся облачные вычислительные среды

В настоящее время уровень развития телекоммуникационных технологий открывает новые возможности организации распределенных вычислений на основе множества пространственно удаленных вычислительных ресурсов с помощью сервис-ориентированной инфраструктуры, обеспечивающей «вычисления по требованию». Эти возможности породили новую парадигму облачных вычислений, т. е. крупномасштабных распределенных вычислений на основе пула абстрактных, виртуализованных, динамически перераспределяемых вычислительных ресурсов, предоставляемых по запросу внешним пользователям через Интернет [30, 31]. В настоящее время облачные вычислительные среды (ОВС) находят все более широкое применение для решения крупномасштабных научных и технических задач различных предметных областей, например таких как физика высоких энергий, науки о земле, химия и биология, космология и астрофизика, экологическая и техногенная безопасность, промышленность, фармакология и фармацевтика, материаловедение, нефтегазодобыча, медицина и т. п. Особенностью таких крупномасштабных научных задач является их сложная внутренняя структура, состоящая, как правило, из ряда информационно взаимосвязанных подзадач, причем зачастую эффективность решения этих подзадач в сильной степени зависит от типа используемого вычислительного ресурса.

Одним из основных преимуществ концепции облачных вычислений является возможность использования в составе ОВС разнотипных (гетерогенных) вычислительных ресурсов, которые имеют различную реальную производительность при решении тех или иных задач. Это обстоятельство, с одной стороны, открывает возможность повышения эффективности (сокращения времени) выполнения крупномасштабных пользовательских задач за счет использования тех вычислительных ресурсов ОВС, которые обеспечивают максимальную реальную производительность при их решении (или решении их подзадач), но с другой стороны, порождает проблему оптимального распределения (диспетчирования) решаемых задач (или их подзадач) по разнотипным вычислительным ресурсам, входящим в состав ОВС. Данная проблема многократно усложняется в случае, если ОВС должна обеспечивать решение не одиночной задачи, а некоторого множества (потока) разнородных пользовательских задач, поступающих в произвольные моменты времени.

Решение задачи распределения ресурсов ОВС по поступающим задачам классическими методами, например методами линейного или динамического программирования [13,14], затруднено, во-первых, вследствие того, что число вычислительных ресурсов ОВС и различных

задач может быть велико, что ведет к экспоненциальному росту вариантов перебора, а во-вторых, поскольку задачи поступают в заранее неизвестные моменты времени, что не позволяет заранее сформировать временной график их исполнения, а требует динамического распределения подзадач различных задач потока между ресурсами ОВС с учетом их функциональных возможностей и производительности на той или иной подзадаче. Все это требует разработки новых подходов к проблеме диспетчирования ОВС при обработке потока крупномасштабных задач, основанных на методах и алгоритмах самоорганизации распределенных систем, предложенных выше.

Формально задачу самоорганизации ОВС можно представить следующим образом.

Предположим, что ОВС должна решать некоторое множество (поток) различных крупномасштабных задач $\boldsymbol{Z} =< Z_1, Z_2, ..., Z_L >$, которые могут поступать от Заказчиков в случайные моменты времени. При этом под крупномасштабной задачей будем понимать задачу, состоящую из некоторого множества информационно зависимых (взаимосвязанных) подзадач, каждая из которых имеет значительную вычислительную трудоемкость. Формально каждая такая крупномасштабная задача $Z_l \in \boldsymbol{Z}$ может быть представлена в виде некоторого графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ (рис. 4.9), вершины $q_j \in \boldsymbol{Q}_l$ которого соответствуют некоторым подзадачам O_j, принадлежащим множеству подзадач $\boldsymbol{O} =< O_1, O_2, ..., O_c >$, а дуги $x(q_j, q_{j+1}) \in \boldsymbol{X}_l$ определяют информационные взаимосвязи между подзадачами, т. е. если две вершины q_j и q_{j+1} соединены дугой $x(q_j, q_{j+1})$, то это означает, что данные, полученные в результате решения подзадачи O_j, приписанной вершине q_j, необходимы для выполнения подзадачи O_{j+1}, приписанной вершине q_{j+1}. Будем считать, что каждой вершине $q_j \in \boldsymbol{Q}_l$ приписан тип решаемой подзадачи $O_j \in \boldsymbol{O}$, а также ее вычислительная трудоемкость $v(O_j)$, оцениваемая числом элементарных вычислительных операций, выполняемых при ее решении, а дуге $x(q_j, q_{j+1}) \in \boldsymbol{X}_l$ приписан объем данных $W(O_j)$, передаваемых от подзадачи O_j, приписанной вершине q_j, следующей подзадаче O_{j+1}, приписанной вершине q_{j+1} (рис. 4.9). Кроме того, положим, что Заказчик задает и приписывает конечной вершине q_k графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ момент времени T^l_{max}, к которому он желает получить результат решения своей задачи Z_l (рис. 4.9), а также устанавливает количество S^l премиальных баллов (бонусов), которое он готов заплатить за это, причем

$$S^l = \begin{cases} S^l_{max} \cdot \dfrac{\Delta T^l_{max} - \Delta T^l}{\Delta T^l_{max}}, \text{если } \Delta T^l \le \Delta T^l_{max}, \\ 0, \text{если } \Delta T^l > \Delta T^l_{max}, \end{cases}$$

где S^l_{max} – максимальное количество премиальных баллов, которое Заказчик готов заплатить за решение его задачи к моменту времени T^l_{max};

ΔT^l_{max} – максимально допустимая задержка решения задачи Z_l относительно момента T^l_{max};

ΔT^l – реально получаемая задержка решения задачи Z_l относительно момента T^l_{max}.

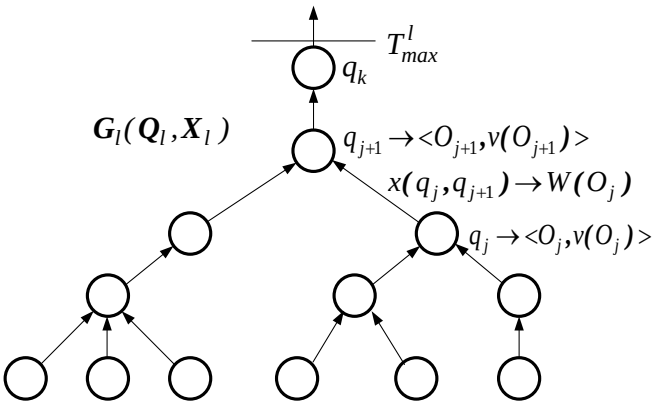


Рис. 4.9. Граф $G_l(Q_l, X_l)$ крупномасштабной вычислительной задачи Z_l

Предположим, что в состав ОВС входит множество вычислительных ресурсов $R =< R_1, R_2, ...R_N >$. При этом под вычислительным ресурсом будем понимать некоторое вычислительное устройство (ПК, сервер, многопроцессорная вычислительная система и т. п.), подключенное к облачной инфраструктуре (рис. 4.10). Будем считать, что каждый ресурс $R_i \in R$ может решать некоторый набор (подмножество) типов подзадач $O_i =< O^i_1, O^i_2, ..., O^i_K >\subseteq O\ (i = 1,2,..., N)$, причем производительность ресурса R_i при решении подзадачи $O^i_j \in O_i$ составляет $D_i(O^i_j)\ (j = 1,2,..., K)$ операций в секунду. Кроме того, будем считать, что пропускная способность канала связи ресурса $R_i \in R$ с облачной инфраструктурой составляет Y_i байт/сек.

Очевидно, что такая ОВС может быть отнесена к классу гетерогенных СРС второго типа (см. подраздел 1.1).

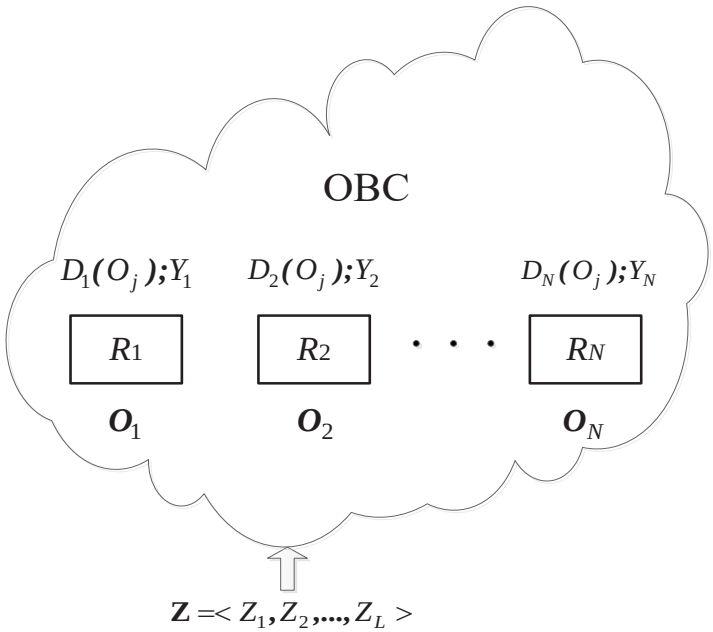


Рис. 4.10. Облачная вычислительная среда

Целью работы диспетчера ОВС является такое распределение подзадач потока задач $\boldsymbol{Z}$ по вычислительным ресурсам $R_1, R_2, ..., R_N$, при котором будет достигаться минимум величины ΔT средней задержки решения задач потока $\boldsymbol{Z} =< Z_1, Z_2, \dots, Z_L >$ относительно установленных Заказчиками моментов времени T_{max}^l $(l = 1,2, \dots, L)$, причем

$$\Delta T = \frac{\sum_{i=1}^{L} \Delta T^l}{L}.$$

До недавнего времени проблема диспетчирования ОВС решалась, как правило, посредством специально выделенных серверных узлов, функции которых заключались в приеме пользовательских задач и их распределении по вычислительным ресурсам ОВС [32, 33] (в качестве примера можно привести проект «Корпоративное облако СО РАН» [34]). Однако такая централизованная организация диспетчера имеет ряд недостатков. Во-первых, при большом числе разнородных ресурсов в ОВС, отличающихся производительностью и пропускной способностью каналов связи, их оперативное распределение по задачам (а тем более по отдельным информационно связанным подзадачам) с помощью одного центрального диспетчера (или дерева диспетчеров) является крайне проблематичным, причем данная проблема многократно усложняется в случае, если ОВС должна решать не одиночную задачу, а поток пользовательских задач, поступающих в заранее неизвестные моменты времени. Во-вторых, существенно затрудняется масштабирование ОВС (т. е. добавление в ее состав новых вычислительных

ресурсов), что требует внесения изменений в алгоритмы работы диспетчера. И наконец, в-третьих, ОВС с централизованным диспетчером имеет низкую отказоустойчивость, поскольку выход из строя служебного серверного узла, реализующего функции диспетчера, приводит к катастрофическим последствиям для всей ОВС в целом.

Указанные недостатки устраняются при использовании принципов самоорганизации распределенных систем, предложенных в предыдущих разделах. При этом в качестве основных активных элементов следует использовать программных агентов AR_i, физически реализуемых на каждом вычислительном ресурсе $R_i \in \boldsymbol{R}\,(i=1,N)$, входящем в состав ОВС и представляющем его «интересы» в процессе диспетчирования. При этом взаимодействие Заказчиков и агентов должно осуществляется посредством некоторого множества специально выделенных пассивных узлов, составляющих служебный уровень облачной инфраструктуры и выполняющих функции досок объявлений (ДО) (рис. 4.11).

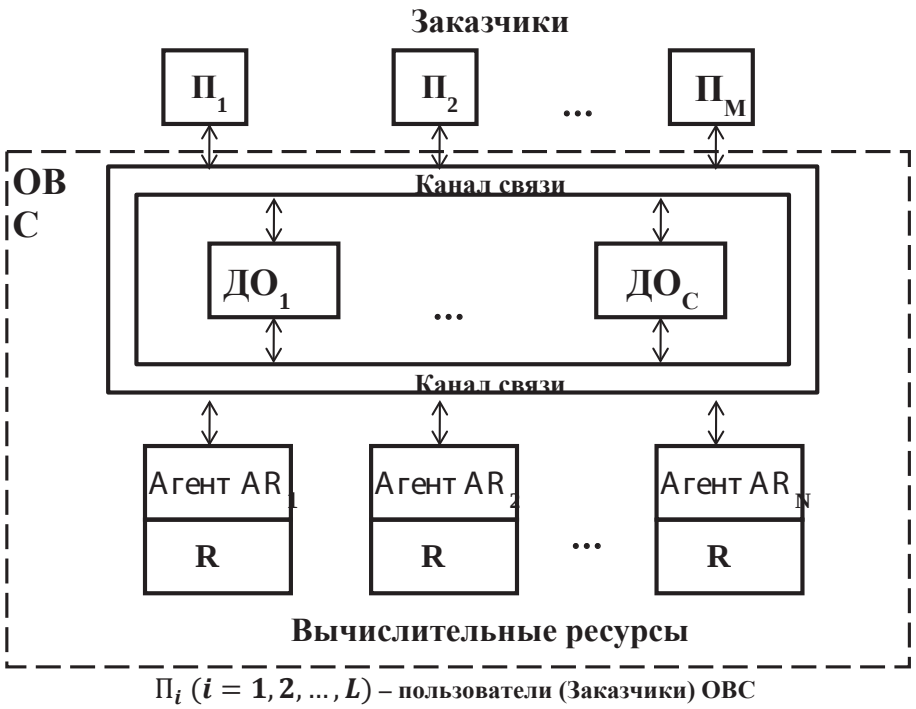


Рис. 4.11. Структура самоорганизующейся ОВС с мультиагентным диспетчером

В укрупненном виде работу такой самоорганизующейся ОВС с мультиагентным диспетчером можно представить в следующем виде.

1. Заказчик (пользователь) формирует свою задачу $Z_l \in \boldsymbol{Z}$ в виде графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, устанавливает требуемый момент времени T^l_{max}, к

которому он желает получить ее решение, и величину премиальных баллов S^l_{max}, которую он готов заплатить за ее решение к данному моменту времени T^l_{max}, а также величину максимально допустимой задержки ΔT^l_{max}. Дескриптор представленной таким образом задачи $Z_l \in \boldsymbol{Z}$ размещается на одной из досок объявлений (см. рис. 4.11).

2. Агент AR_i ресурса R_i , не задействованного при решении других задач, опрашивает доски объявлений в поисках работы для «своего» ресурса. В случае обнаружения на ДО некоторого множества пользовательских задач $\boldsymbol{Z} =< Z_1, Z_2, \ldots, Z_M >$ агент AR_i выбирает ту задачу Z_l, для которой средняя стоимость единицы трудоемкости вычислений $S^l_{\text{ср}}$ (т. е. соотношение величины премиальных баллов, предлагаемых пользователем за решение данной задачи, к ее суммарной вычислительной трудоемкости) максимальна.

3. Агент AR_i выделяет в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задачи Z_l фрагмент (ветвь), состоящий из последовательности подзадач, не закрепленных за другими агентами, за выполнение которого он потенциально может получить максимальное число премиальных баллов S^l_i, после чего агент AR_i приступает к его решению.

4. После успешного завершения решения задачи Z_l всем агентам сообщества $\boldsymbol{R}_l$, принимавшим участие в ее решении, начисляются премиальные баллы, зависящие от величины общей задержки ΔT^l решения задачи Z_l относительно установленного Заказчиком момента времени ΔT^l_{max}, а также суммарной вычислительной трудоемкости подзадач задачи Z_l, решенных ресурсом R_i.

Основным преимуществом предлагаемого подхода к созданию самоорганизующейся ОВС является то, что вычислительный процесс будет адаптироваться к вычислительным возможностям ресурсов, задействованным в ее составе. Кроме того, по сравнению с классической, централизованной организацией диспетчера облачной среды в данном случае упрощаются требования к служебным серверам (доскам объявлений), что позволяет существенно снизить накладные расходы на организацию облачной среды и, как следствие, снизить стоимость облачных вычислений, а также упростить процесс масштабирования облачной среды.

Прежде чем приступить к разработке алгоритма мультиагентного диспетчирования ОВС, уточним для рассматриваемого случая введенное в подразделе 1.3 понятие ветви графа задачи. Под ветвью будем понимать некоторую последовательность вершин $\boldsymbol{H}_f =< q^f_1, q^f_2, \ldots, q^f_k >$ графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задачи $Z_l \in \boldsymbol{Z}$, в которой вершины q^f_j и q^f_{j+1} $(j = 1,2,\ldots,k-1)$ соединены дугой $x(q^f_j, q^f_{j+1})$ (рис. 4.12). Иными словами, ветвь определяет

некоторую последовательность подзадач задачи Z_l, в которой каждая последующая подзадача использует в качестве исходных данных результат выполнения предыдущей подзадачи. Очевидно, что подзадачи, приписанные вершинам ветви $\boldsymbol{H}_f$, могут выполняться только последовательно.

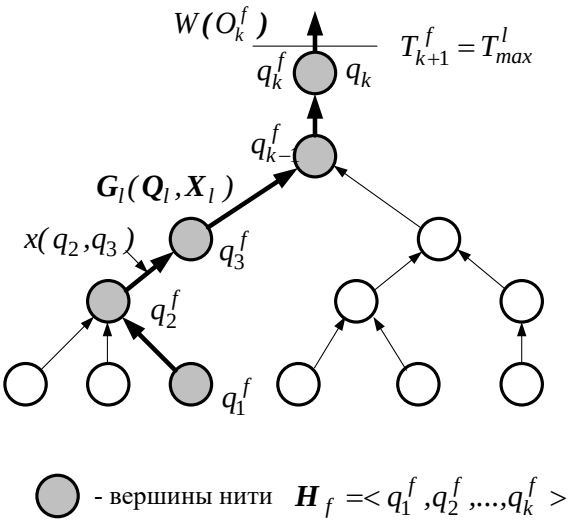


Рис. 4.12. Выделение ветви $\boldsymbol{H}_f$ в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задачи Z_l

При этом под длиной t_f ветви $\boldsymbol{H}_f$ будем понимать суммарное время, затрачиваемое на ее решение, определяемое как:

$$t_f = \sum_{i=1}^{k} \left(\frac{v(O_i^f)}{D_p(O_i^f)} + t_{i,i+1}^f \right),$$

где $v(O_i^f)$ – вычислительная трудоемкость подзадачи O_i^f, приписанной вершине q_i^f $(i = 1,2,...,k)$ ветви $\boldsymbol{H}_f$;

$D_p(O_i^f)$ – производительность ресурса R_p, решающего подзадачу O_i^f;

$t_{i,i+1}^f$ – время передачи данных, полученных в результате выполнения подзадачи O_i^f, приписанной вершине q_i^f, подзадаче O_{i+1}^f, приписанной смежной вершине q_{i+1}^f, причем будем считать, что

$$t_{i,i+1}^{f}=\begin{cases}0, \text{если подзадачи } O_i^f \text{ и } O_{i+1}^f \text{ решаются одним и тте же} \\ \quad \text{ресурсом } R_p \in \mathbf{R}, \\ \dfrac{W(O_i^f)}{Y_p}, \text{если подзадачи } O_i^f \text{ и } O_{i+1}^f \text{ решаются различными} \\ \quad \text{ресурсами } R_p \text{ и } R_c, \end{cases}$$

где $W(O_i^f)$ – объем данных, приписанных дуге $x(q_j,q_{j+1}) \in \boldsymbol{X}_l$ и, соответственно, передаваемых от подзадачи O_i^f следующей подзадаче O_{i+1}^f;

Y_p – пропускная способность канала связи ресурса R_p с облачной инфраструктурой.

Тогда очевидно, если вся ветвь $\boldsymbol{H}_f$ выполняется одним ресурсом R_p, то ее длина будет составлять

$$t_f^p=\sum_{i=1}^{k}\frac{v(O_i^f)}{D_p(O_i^f)}+\frac{W(O_k^f)}{Y_p}, \tag{4.3}$$

где $W(O_k^f)$ – объем данных, приписанный исходящей (конечной) дуге ветви $\boldsymbol{H}_f$ (см. рис. 4.12).

При этом если известен требуемый момент времени T_{k+1}^f исполнения всей ветви $\boldsymbol{H}_f$, то можно определить допустимые моменты времени T_d^f начала выполнения всех подзадач O_d^f, приписанных ее вершинам $q_d^f \in \boldsymbol{H}_f$ $(d=1,2,...,k)$ (при которых ресурс R_p успевает выполнить всю ветвь $\boldsymbol{H}_f$ к требуемому моменту времени T_{k+1}^f), как

$$T_d^f = T_{k+1}^f - \left(\sum_{i=d}^{k}\frac{v(O_i^f)}{D_p(O_i^f)}+\frac{W(O_k^f)}{Y_p}\right), \ (d=1,2,...,k-1). \tag{4.4}$$

Если при этом оказывается, что $T_1^f < T_{\text{тек}}$, то это означает, что ресурс R_p может выполнить ветвь $\boldsymbol{H}_f$ только с задержкой

$$\Delta T^f = T_{\text{тек}} - T_1^f \tag{4.5}$$

относительно установленного момента времени T_{k+1}^f исполнения последовательности подзадач ветви $\boldsymbol{H}_f$.

Исходя из этих соображений, можно предложить следующую процедуру мультиагентного диспетчирования ресурсов $R_1,R_2,...,R_N$, входящих в состав гетерогенной ОВС, с учетом их вычислительных и коммуникационных возможностей при решении потока задач $\boldsymbol{Z}$.

Заказчик формирует свою задачу $Z_l \in \boldsymbol{Z}$ в виде графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ и определяет требуемый момент времени T^l_{max}, к которому ее решение должно быть получено, величину премиальных баллов S^l_{max}, которую он готов заплатить за ее решение к установленному моменту времени T^l_{max}, а также максимально допустимую задержку ΔT^l_{max}. Дескриптор представленной таким образом задачи $Z_l \in \boldsymbol{Z}$ размещается на одной из досок объявлений (см. рис. 4.11).

Агенты, ресурсы которых не задействованные в решении каких либо задач, обращаются к ДО в поисках работы. Если агент свободного ресурса $R_p \in \boldsymbol{R}$ обнаруживает на ДО некоторое множество задач $\boldsymbol{Z} =< Z_1, Z_2, ..., Z_L >$, то он выбирает среди них ту задачу Z_l, которая имеет максимальное значение стоимости единицы трудоемкости

$$S^l_{\text{ср}} = \frac{S^l_{max}}{\sum_{i=1}^{M} v(O_i)},$$

где $\sum_{i=1}^{M} v(O_i)$ – суммарная вычислительная трудоемкость всех подзадач задачи $Z_l, M = |\boldsymbol{Q}_l|$, после чего делает попытку войти в сообщество (подмножество) ресурсов $\boldsymbol{R}_l \subseteq \boldsymbol{R}$, задействованных в ее решении.

Поскольку, как мы приняли выше, каждый ресурс ОВС имеет некоторую специализацию (т. е. может решить ограниченный набор подзадач), то в общем случае может оказаться, что ресурс R_p способен выполнять далеко не все подзадачи, приписанные вершинам графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задачи Z_l. Поэтому в графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ необходимо предварительно выделить подграф $\boldsymbol{G}^p_l(\boldsymbol{Q}^p_l, \boldsymbol{X}^p_l)$, вершинам которого приписаны подзадачи множества $\boldsymbol{O}_p$, выполняемые ресурсом R_p (рис. 4.13).

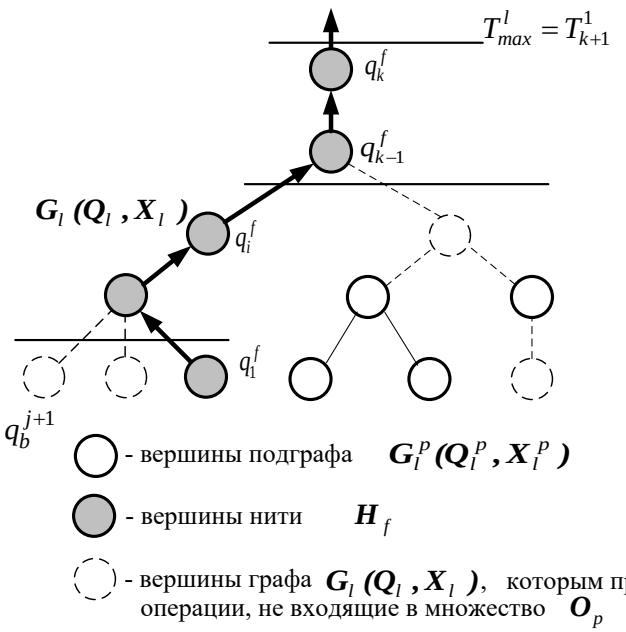


Рис. 4.13. Выделение подграфа $G^p_l\,(Q^p_l\,, X^p_l\,)$ в графе $G_l(Q_l\,, X_l\,)$ задачи Z_l

После этого агент AR_p должен выбрать в графе $G^p_l\,(Q^p_l\,, X^p_l\,)$ ветвь H_f, для которой выполняются следующие условия (см. подраздел 3.1):

– во-первых, установлен момент времени ее исполнения T^f_{k+1} (в момент размещения Заказчиком задания Z_l на ДО такое время установлено только для конечной вершины графа $G_l(Q_l\,, X_l\,)$, т. е. в этом случае $T^f_{k+1} = T^l_{max}$) (см. рис. 4.13);

– во-вторых, величина

$$S^f_p = \begin{cases} \dfrac{\sum_{i=1}^{k} v(O^f_i)}{\sum_{j=1}^{M} v(O_j)} \cdot S^l_{max} \cdot \dfrac{\Delta T^l - \Delta T^f_p}{\Delta T^l}, \text{если } \Delta T^l_p < \Delta T^l \\ 0, \text{если } \Delta T^l_p \geq \Delta T^l \end{cases} \qquad (4.6)$$

максимальна, где

S^f_p – количество премиальных баллов, получаемых агентом AR_p за выполнение операций ветви H_f задания Z_l;

ΔT^l_p – задержка выполнения ветви H_f задания Z_l ресурсом R_p;

$\Delta T^l = \Delta T^l_{max} - \sum_{i=1}^{j-1} \Delta T^l_i$;

$j - 1$ – число агентов, уже вступивших ранее в сообщество R_l;

ΔT^l_i – задержка, вносимая в выполнение задания Z_l ресурсом R_i (i = 1, 2, …, $j - 1$), ранее вступившим в сообщество R_l.

$\sum_{i=1}^{k} v(O_i^f)$ – суммарная вычислительная трудоемкость операций ветви $\boldsymbol{H}_f$.

Учитывая, что величину задержки ΔT_p^f можно определить с помощью следующего выражения (см. выражения (4.4) и (4.5)).

$$\Delta T_p^l = \begin{cases} T_{\text{тек}} - T_1^f, \text{если } T_{\text{тек}} > T_{1,}^f \\ 0, \text{если } T_{\text{тек}} \le T_1^f, \end{cases}$$

где
$$T_1^f = T_{k+1}^f - \left(\sum_{i=1}^{k} \frac{v(O_i^f)}{D_p(O_i^f)} + \frac{W(o_k^f)}{Y_p} \right) \quad (4.7)$$

– момент времени начала выполнения первой подзадачи O_1^f ветви $\boldsymbol{H}_f$ ресурсом R_p, при котором последний успевает завершить ее исполнение к требуемому моменту T_{k+1}^f.

Подставляя последнее выражение в (4.5), получаем

$$S_p^f = \begin{cases} \sum_{i=1}^{k} v(O_i^f) \cdot S_{max}^l \cdot \left(\Delta T^l - T_{\text{тек}} - \left(T_{k+1}^f - \sum_{i=1}^{k} \frac{v(O_i^f)}{D_p(O_i^f)} + \frac{W(o_k^f)}{Y_p} \right) \right), \text{если } \Delta T_p^l < \Delta T^l; \\ 0, \text{если } \Delta T_p^l \ge \Delta T^l. \end{cases} \quad (4.8)$$

Таким образом, агент AR_p должен выбрать для исполнения такую ветвь $\boldsymbol{H}_f$ задания Z_l, для которой значение величины (4.8) максимально.

После того, как агент AR_p выделил некоторую ветвь $\boldsymbol{H}_f = < q_1^f, q_2^f, ..., q_k^f >$, удовлетворяющую перечисленным выше условиям, и принял ее к исполнению, он должен произвести модификацию графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задания Z_l на ДО. При этом:

1. Идентификатор агента AR_p записывается в список членов сообщества $\boldsymbol{R}_l$ по решению задачи Z_l.

2. Вершины, входящие в ветвь $\boldsymbol{H}_f$, исключаются из графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задачи Z_l, в результате чего формируется новый граф $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l) = \boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l) / \boldsymbol{H}_f$ (рис. 4.14).

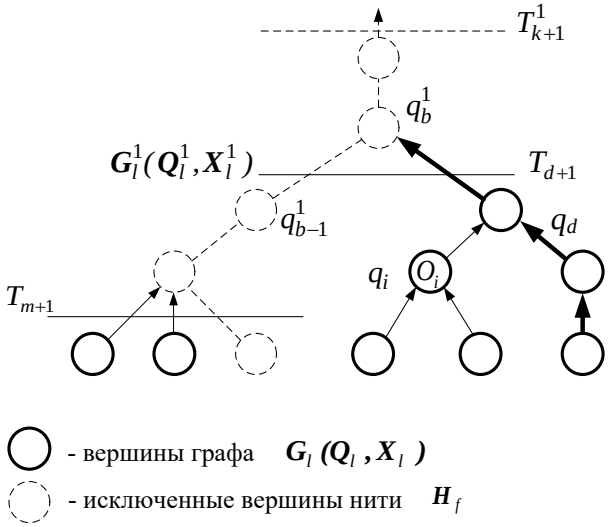


Рис. 4.14. Граф $G_l(Q_l, X_l)$ задачи Z_l, модифицированный агентом R_p

3. $\Delta T^l = \Delta T^l - \Delta T^l_p$,

где $\Delta T^l_p = T_{\text{тек}} - T^f_1$ (см. выражение (4.7) (в начальный момент времени размещения задания Z_l на ДО $\Delta T^l = \Delta T^l_{max}$)).

4. Всем вершинам графа $G_l(Q_l, X_l)$, инцидентным вершинам ветви H_f, приписываются требуемые моменты времени их исполнения, которые определяются исходя из следующих соображений.

Допустим, что некоторая вершина q_d модифицированного графа $G_l(Q_l, X_l)$ инцидентна вершине q^f_b, принадлежащей ветви H_f (см. рис. 4.14). Это означает, что для выполнения подзадачи, приписанной вершине q^f_b, необходимы результаты выполнения подзадачи, приписанной вершине q_d. Поэтому очевидно, что результаты выполнения подзадачи, приписанной вершине q_d, должны быть получены и переданы агенту R_p, выполняющему операции ветви H_f, не позже, чем к требуемому моменту времени T^f_b начала выполнения агентом R_p подзадачи O^f_b, приписанной вершине q^f_b, определяемому согласно выражению (4.4) как:

$$T^f_b = T^f_{k+1} - \left(\sum_{i=b}^{k} \frac{v(O^f_i)}{D_p(O^f_i)} + \frac{W(O^f_k)}{Y_p} \right). \qquad (4.9)$$

В противном случае ресурс R_p не успеет закончить исполнение взятой на себя ветви H_f к требуемому моменту времени T^f_{k+1}.

Поэтому вершине q_d графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ приписывается требуемое время ее исполнения $T_{d+1} = T_b^f$, а также идентификатор агента AR_p, которому результаты исполнения подзадачи, приписанной вершине q_d, должны быть переданы (см. рис. 4.14).

Аналогичным образом определяются требуемые моменты T_{m+1} исполнения всех остальных вершин графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, инцидентных вершинам ветви $\boldsymbol{H}_f$ (см. рис. 4.14).

Модифицированный таким образом граф $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задачи Z_l размещается агентом AR_p на ДО, и если граф еще не пустой, т. е. $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l) \neq \varnothing$, то процесс создания сообщества $\boldsymbol{R}_l$ для выполнения задачи Z_l продолжается далее путем включения в него новых агентов свободных ресурсов из множества $\boldsymbol{R}$.

После того, как агент R_p выбрал для исполнения ветвь $\boldsymbol{H}_f = < q_1^f, q_2^f, ..., q_k^f >$, он приступает к реализации подзадач, приписанных ее вершинам. При этом перед началом выполнения очередной подзадачи O_d^f, приписанной вершине $q_d^f \in \boldsymbol{H}_f$ $(d = 1,2,...,k)$, агент R_p должен проверить, наличие всех исходных данных, необходимых для ее выполнения.

Поскольку для выполнения подзадачи O_d^f могут требоваться исходные данные, получаемые в результате выполнения смежной ветви другим агентом R_c, то к моменту начала выполнения агентом R_p подзадачи O_d^f может оказаться, что эти данные еще не поступили. В этом случае агент R_p должен перейти в режим ожидания поступления необходимых исходных данных.

По завершении решения всех подзадач $O_1^f, O_2^f, \ldots, O_k^f$ ветви $\boldsymbol{H}_f$ агент AR_p должен проверить соблюдение временного графика ее выполнения, для чего он сравнивает момент времени T_{k+1}^f, приписанный последней вершине q_k^f ветви $\boldsymbol{H}_f$, с текущим временем $T_{\text{тек}}$.

Если $T_{\text{тек}} > T_{k+1}^f$, то это означает, что произошла задержка выполнения ветви $\boldsymbol{H}_f$, которая, соответственно, вносит дополнительную задержку $\Delta T = T_{\text{тек}} - T_{k+1}^f$ в общее время решения задачи Z_l.

В этом случае агент AR_p должен модифицировать дескриптор задачи Z_l на ДО, изменив значение величины допустимой задержки $\Delta T^l = \Delta T^l - \Delta T$, а также увеличить значения ранее приписанных вершинам графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ требуемых моментов времени их исполнения на величину ΔT.

В случае успешного решения задачи Z_l не позже, чем к моменту времени $(T^l_{max} + \Delta T^l_{max})$, агенту AR_p начисляются премиальные баллы в размере

$$S^l_p = \frac{S^l_{max} \cdot (\Delta T^l_{max} - \Delta T^l) \cdot V^p_l}{\Delta T^l_{max} \cdot V_l},$$

где ΔT^l – задержка решения задачи Z_l относительно установленного Заказчиком момента $T^l_{\max}$;

$V^p_l = \sum_{i=1}^{k} v(O^p_i)$ – суммарная вычислительная трудоемкость подзадач, выполненных ресурсом R_p;

$V_l = \sum_{i=1}^{M} v(O_i)$ – суммарная вычислительная трудоемкость всех подзадач задачи Z_l.

Описанному выше процессу отвечает следующий укрупненный алгоритм функционирования агента, представляющего ресурс R_p в ОВС.

Алгоритм 4.3

1. Агент свободного ресурса R_p опрашивает ДО.

2. Среди множества $\boldsymbol{Z} =< Z_1, Z_2, ..., Z_L >$, обнаруженных на ДО задач, агент AR_p выбирает задачу Z_l, для которой значение

$$S^l_{\text{ср}} = \frac{S^l_{max}}{\sum_{i=1}^{M} v(O_i)}, \text{ где } M = |\boldsymbol{Q}_l|$$

максимально.

3. В графе $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ задачи Z_l выделяется подграф $\boldsymbol{G}^p_l(\boldsymbol{Q}^p_l, \boldsymbol{X}^p_l)$, вершинам которого приписаны подзадачи множества $\boldsymbol{O}_p$, выполняемые ресурсом R_p.

Если $\boldsymbol{G}^p_l(\boldsymbol{Q}^p_l, \boldsymbol{X}^p_l)=\varnothing$, то $\boldsymbol{Z} = \boldsymbol{Z}/Z_l$ перейти к п. 2, иначе

4. Агент AR_p выделяет в графе $\boldsymbol{G}^p_l(\boldsymbol{Q}^p_l, \boldsymbol{X}^p_l)$ ветвь $\boldsymbol{H}_f =< q^f_1, q^f_2, ..., q^f_k >$, удовлетворяющую условиям:

– конечной вершине q^f_k ветви $\boldsymbol{H}_f$ приписан требуемый момент времени исполнения T^f_{k+1};

– величина

$$S^f_p = \begin{cases} \frac{\sum_{i=1}^{k} v(O^f_i)}{\sum_{i=1}^{M} v(O_i)} \cdot S^l_{max} \cdot \frac{\Delta T^l - \Delta T^l_p}{\Delta T^l}, \text{если } \Delta T^l_p < \Delta T^l, \\ 0, \text{если } \Delta T^l_p \geq \Delta T^l, \end{cases}$$

где $\Delta T^l = \Delta T^l_{max} - \sum_{i=1}^{p-1} \Delta T^l_i$.

4. Агент AR_p принимает ветвь $\boldsymbol{H}_f$ к исполнению и модифицирует дескриптор задачи Z_l на ДО:

– ветвь $\boldsymbol{H}_f$ исключает из графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, т.е. $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l) = \boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l) / \boldsymbol{H}_f$;

– всем вершинам графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, инцидентным вершинам ветви $\boldsymbol{H}_f$ приписываются требуемые моменты времени из исполнения, определяемые как

$$T_{d+1} = T_{k+1}^f - \left(\sum_{i=b}^{k} \frac{v(O_i^f)}{D_p(O_i^f)} + \frac{W(O_k^f)}{Y_p} \right),$$

где q_d вершина графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$ инцидентная вершине q_b^f ветви $\boldsymbol{H}_f$;

$$\Delta T^l = \Delta T^l - \Delta T_p^l,$$

где $\Delta T_p^l = T_{\text{тек}} - \left(T_{k+1}^f - \left(\sum_{i=1}^{k} \frac{v(O_i^f)}{D_p(O_i^f)} + \frac{W(O_k^f)}{Y_p} \right) \right)$.

5. Агент AR_p приступает к решению последовательности подзадач $O_1^f, O_2^f, \ldots, O_k^f$, приписанных вершинам ветви $\boldsymbol{H}_f$: $i = 1$:

6. Агент AR_p проверяет наличие исходных данных, необходимых для решения подзадач O_i^f. Если исходные данные еще не поступили, то перейти к 6, иначе

7. Агент AR_p инициирует выполнение операции O_i^f, приписанной вершине $q_i^f \in \boldsymbol{H}_f$, с помощью локального УУ$_p$ «своего» ресурса R_p.

8. $i = i + 1$, если $i \le k$, то перейти к п. 6, иначе

9. Если $T_{\text{тек}} \le T_{k+1}^f$, то перейти к п. 11, иначе

10. $\Delta T^l = \Delta T^l - \left(T_{\text{тек}} - T_{k+1}^f\right)$ и всем вершинам q_d графа $\boldsymbol{G}_l(\boldsymbol{Q}_l, \boldsymbol{X}_l)$, которым приписано требуемое время исполнения T_{d+1}, устанавливается новое время исполнения $T_{d+1} = T_{d+1} - \left(T_{\text{тек}} - T_{k+1}^f\right)$.

11. Если при завершении решения всей задачи Z_l в целом выполняется условие

$$T_{\text{тек}} \le \Delta T_{max}^l + \Delta T_{max}^l,$$

то агенту AR_p начисляются премиальные баллы в размере

$$S_p^l = \frac{S_{max}^l \cdot (\Delta \text{п} T_{max}^l - \Delta T^l) \cdot V_l^p}{\Delta T_{max}^l \cdot V_l},$$

где $V_l^p = \sum_{i=1}^{k} v(O_i^f)$ – трудоемкость подзадач, выполненных ресурсом R_p;

$V_l = \sum_{i=1}^{M} v(O_i)$ – суммарная трудоемкость всех подзадач задачи Z_l.

4.4. Самоорганизующиеся системы управления воздушных судов

4.4.1. Эволюция архитектуры систем управления воздушных судов

Система управления является важным компонентом современного воздушного судна, во многом определяющим его безопасность и эффективность. На разных этапах развития авиации на воздушных судах применялись системы управления с различными архитектурами, что было обусловлено номенклатурой реализуемых функций, имеющейся в наличии элементной базой, существующими методами и подходами к построению управляющих систем. Можно выделить следующие основные виды архитектур систем управления воздушных судов (СУ ВС) [35-37]:

– конфедеративная архитектура;

– федеративная архитектура;

– архитектура на базе концепции интегрированной модульной авионики (ИМА).

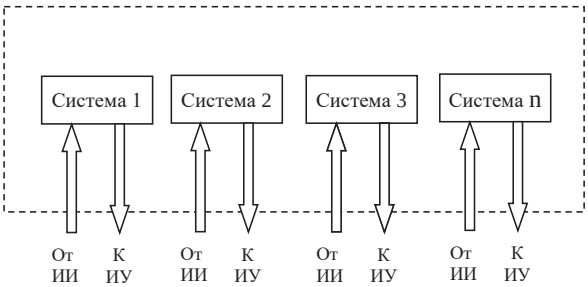
Конфедеративная архитектура (рис. 4.15) подразумевает наличие в составе системы управления воздушным судном множества независимых (не связанных друг с другом) систем, каждая из которых реализует определенную функцию самолетовождения [38]). К недостаткам такой архитектуры следует отнести:

– практически полное отсутствие унификации оборудования, т. к. каждая из подсистем разрабатывается и создается отдельным производителем, вследствие этого увеличивается время разработки системы управления в целом, снижается ее ремонтопригодность, затрудняется модернизация, увеличивается номенклатура комплектующих;

– отсутствие возможности разделения ресурсов между отдельными подсистемами, поскольку каждая из них комплектуется своими уникальными датчиками, вычислителями и другими устройствами, вследствие этого ухудшаются массо-габаритные характеристики системы управления в целом и увеличивается ее стоимость;

– необходимость использования в каждой системе отдельного дополнительного оборудования для обеспечения ее отказоустойчивости, что ухудшает массо-габаритные характеристики, значительно увеличиваются энергопотребление и стоимость системы управления в целом;

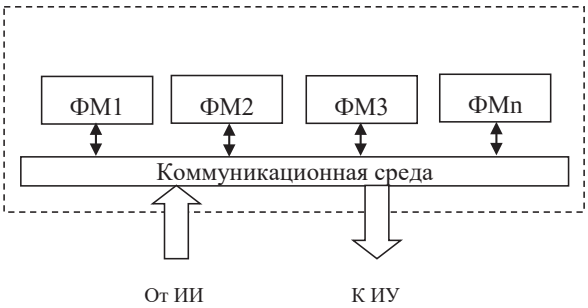
– расширение номенклатуры функций, решаемых системой управления, увеличивает число входящих в ее состав отдельных систем, что также негативно отражается на массо-габаритных характеристиках.



ИИ – источники информации ИУ – исполнительные устройства

Рис. 4.15. Система управления с конфедеративной архитектурой

Федеративная архитектура в системах управления воздушных судов появилась в начале 80-х годов XX века вследствие расширения номенклатуры реализуемых функций самолетовождения. Ее главным отличием является распределение функций между взаимодействующими подсистемами, связанных между собой посредством мультиплексных каналов и других линий передач информации (рис. 4.16) [39,40]. При этом, как и в конфедеративной архитектуре, каждая из подсистем представляет собой отдельный функциональный модуль, ориентированный на выполнение одной из функций самолетовождения.



ИИ – источники информации. ИУ – исполнительные устройства.
ФМ – функциональный модуль.

Рис. 4.16. Система управления с федеративной архитектурой

По сравнению с конфедеративной архитектурой федеративная архитектура позволила за счет комплексной обработки бортовой информации и применения мультиплексированных линий связи улучшить массо-габаритные характеристики систем управления воздушных судов. Однако при этом системы управления, построенные на базе данной архитектуры, обладают следующими недостатками [38]:

– отсутствие унификации различных блоков, в частности функциональных модулей (ФМ) и коммуникационного оборудования;

– программное обеспечение неразрывно связано с конструкцией функционального модуля и поэтому не может быть вторично использовано в последующих модификациях системы управления или при наращивании ее дополнительных функций без повторной верификации и валидации;

– для обеспечения устойчивости к отказам необходимо осуществлять резервирование каждого функционального модуля по отдельности, что требует больших дополнительных аппаратных затрат.

С начала 2000-х годов начала развиваться принципиально новая концепция построения архитектуры систем управления воздушных судов, так называемая концепция интегрированной модульной авионики. Данная концепция предполагает, что различные функции самолетовождения выделяются в логические разделы, которые могут физически реализовываться на любом унифицированном вычислительном устройстве (процессорном узле), входящем в состав СУ ВС (рис. 4.17). При этом все процессорные узлы (ПУ) соединяются посредством коммуникационной среды на базе высокоскоростных бортовых сетевых интерфейсов (коммуникационной сети (КС)) и являются высокоинтегрированными устройствами с общим программным уровнем, типичным для спецификации ARINC 653 APEX. Функции подсистем комплекса в этом случае выполняют программные приложения, разделяющие общие вычислительные ресурсы [38,41].

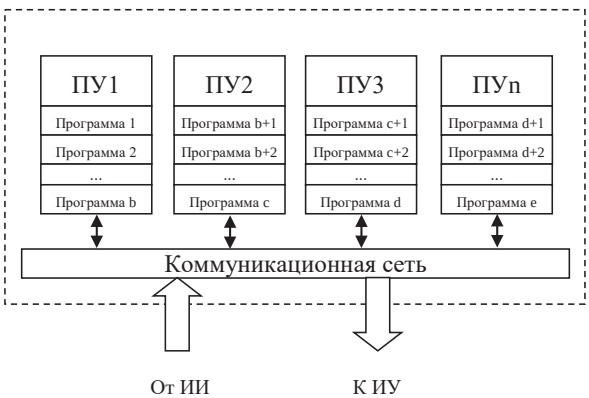


Рис. 4.17. Система управления ВС с архитектурой ИМА

Архитектура СУ ВС на базе концепции ИМА является открытой, ориентированной на широкое применение унифицированных решений, аппаратуры и программ. Данная архитектура позволяет реализовать несколько функций на одних и тех же аппаратных средствах, выполнять раздельную разработку аппаратных и программных средств. Все это позволяет сократить время и снизить стоимость разработки систем управления воздушных судов.

Еще одним преимуществом архитектуры СУ ВС на базе концепции ИМА является возможность использования описанных выше методов самоорганизации для повышения ее отказоустойчивости. Суть предлагаемого подхода заключается в следующем.

Одним из важнейших требований, предъявляемых к СУ ВС, является их высокая надежность и отказоустойчивость. В настоящее время наиболее распространенным способом повышения надежности технических систем является способ структурного резервирования [42]. При использовании данного способа вероятность безотказной работы вычислительной части СУ ВС повышается за счет введения в ее состав дополнительных резервных ПУ, которые непосредственно не участвуют в решении задачи управления, но могут заменить в случае необходимости вышедшие из строя (отказавшие) основные ПУ. Однако этот способ имеет серьезные недостатки, затрудняющие, а порой и делающие невозможным его применение при создании бортовых СУ ВС с архитектурой ИМА, а именно: введение в состав СУ ВС дополнительных ПУ, которые большую часть времени могут простаивать (т. е. не быть задействованными в полезном вычислительном процессе до момента выхода из строя одного из основных ПУ), приводит к увеличению массогабаритных характеристик СУ ВС и ее энергопотребления.

Можно предложить иной подход к проблеме повышения надежности СУ ВС, лишенный указанных выше недостатков.

Допустим, что СУ ВС предназначена для решения некоторой задачи управления (в случае СУ ВС под задачей управления понимается множество программ, необходимых для реализации заданных функций самолетовождения, причем каждую отдельную программу можно рассматривать как подзадачу), которая в общем виде может быть представлена в виде некоторого графа $\boldsymbol{G}(\boldsymbol{Q},\boldsymbol{X})$, как это показано на рис. 4.17. При этом будем считать, что каждой вершине $q_j \in \boldsymbol{Q}_1$ графа $\boldsymbol{G}(\boldsymbol{Q},\boldsymbol{X})$ приписана некоторая подзадача O_j, принадлежащая множеству решаемых подзадач $\boldsymbol{O}$, а дуга $x(q_j, q_{j+1})$ устанавливает взаимосвязь между подзадачами, приписанными вершинам q_j и q_{j+1}, т. е. определяет, что результат решения подзадачи O_j, приписанной вершине q_j, необходим для выполнения подзадачи O_{j+1}, приписанной вершине q_{j+1}.

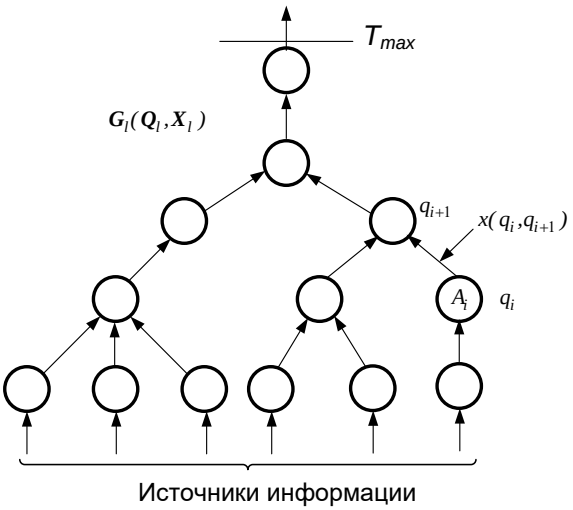


Рис. 4.17. Граф задачи управления

При этом будем считать, что подзадача O_j имеет некоторую трудоемкость $v(O_j)$, которая определяется числом элементарных действий, которые необходимо осуществить для ее выполнения, а дуге $x(q_j, q_{j+1}) \in \boldsymbol{X}$ приписан объем данных $W(O_j)$, передаваемых от подзадачи O_j, приписанной вершине q_j, следующей подзадаче O_{j+1}, приписанной вершине q_{j+1}.

При этом для того, чтобы обеспечить режим реального времени при управлении воздушным судном, задача управления должна решаться за ограниченный промежуток времениT_{max}.

Примем, что в состав СУ ВС входит N одинаковых ПУ$_i$ $(i = 1,2,...,N)$ (см. рис. 4.17), которые могут решать все множество подзадач $\boldsymbol{O}$, приписанных вершинам графа задачи управления $\boldsymbol{G}(\boldsymbol{Q}, \boldsymbol{X})$, причем все ПУ $(i = 1,2,...,N)$ имеют одинаковую производительность $D_{\text{ПУ}}$, а пропускная способность канала связи ПУ$_i$ $(i = 1,2,...,N)$ с другими ПУ составляет Y байт/с.

Очевидно, что СУ ВС с такой архитектурой можно отнести к классу гомогенных самоорганизующихся систем первого типа (см. подраздел 1.1).

Допустим, что подзадачи задачи управления, приписанные вершинам графа задачи управления $\boldsymbol{G}(\boldsymbol{Q}, \boldsymbol{X})$, распределены по ПУ $(i = 1,2,...,N)$, входящим в состав СУ ВС, таким образом, чтобы обеспечивался режим реального времени при решении задачи управления, т. е. чтобы время решения задачи управления не превышало величины T_{max}. При этом на каждое ПУ возлагается решение некоторого подмножества подзадач $\boldsymbol{O_i}$, причем опять-

таки для обеспечения требований реального времени ПУ$_i$ должен решать возложенные на него множество подзадач $\boldsymbol{O_i}$ за время $T^i_{max} \le T_{max}$.

Допустим, что ПУ$_i$ $(i = 1,2,...,N)$ имеют некоторые резервы производительности, т. е. за отведенное время T^i_{max} могут решить большее число подзадач, чем в закрепленным за ним подмножестве $\boldsymbol{O_i}$ (рис. 4.18.). Тогда в случае выхода из строя некоторого ПУ$_i$ множество подзадач $\boldsymbol{O_i}$, закрепленное за ним, можно перераспределить между другими ПУ, имеющими достаточный для их выполнения резерв производительности (рис. 4.18). Предлагаемый подход имеет следующие важное преимущество: для достижения заданного уровня надежности не нужно вводить в состав СУ ВС дополнительные резервные ПУ, а достаточно создать резерв производительности на уже имеющихся ПУ, что достигается либо путем его «закладки» на этапе создания СУ ВС, либо просто заменой процессоров на более производительные.

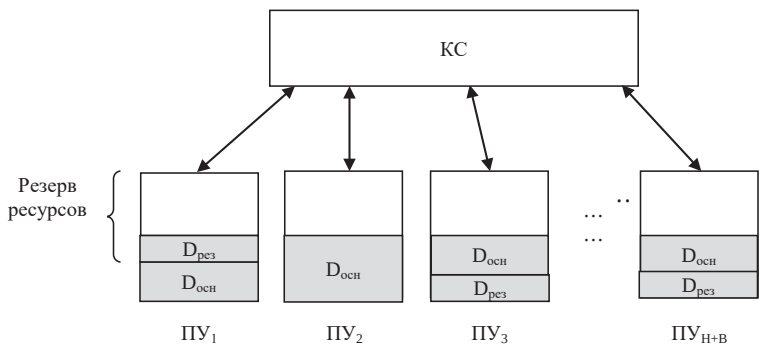


Рис. 4.18. Распределение вычислительной нагрузки между процессорами СУ ВС с резервированием производительности

Рассмотрим основные принципы применения способа резервирования производительности для повышения надежности СУ ВС более подробно. Для наглядности и простоты дальнейших рассуждений рассмотрим сначала частные случаи.

Допустим, что в состав СУ ВС входят два ПУ с производительностью $D_{\text{ПУ}} = S$, которые обеспечивают решение всех подзадач задачи управления $\boldsymbol{G(Q,X)}$ за время

$$T_B = \frac{\sum_{i=1}^{M} v(O_i)}{2S},$$

где M – число вершин в графе $\boldsymbol{G(Q,X)}$,

$v(O_i)$ $(i = 1, 2, \dots, M)$ – вычислительная сложность подзадачи, приписанной i-й вершине графа $\boldsymbol{G(Q,X)}$.

При этом вероятность безотказной работы такого СУ ВС, состоящего из двух ПУ, будет равна

$$P_{\text{В}} = P^2_{\text{ПУ}}.$$

Заменим в устройстве ПУ с производительностью $D_{\text{ПУ}}$ на ПУ с производительностью $D_{\text{ПУ}} = 2S$ (рис. 4.19). В этом случае при выходе из строя одного из ПУ второй сможет обеспечить решение задачи $\boldsymbol{G(Q,X)}$ за то же время $T_{\text{в}}$, поскольку его производительности для этого будет достаточно. Поэтому вероятность безотказной работы двух ПУ с производительностью $D_{\text{ПУ}} = 2S$ будет уже равна

$$P_{\text{в}} = 1 - (1 - P_{\text{ПУ}})^2 = 1 - (1 - 2P_{\text{ПУ}} + P_{\text{ПУ}}^2) = 2P_{\text{ПУ}} - P_{\text{ПУ}}^2,$$

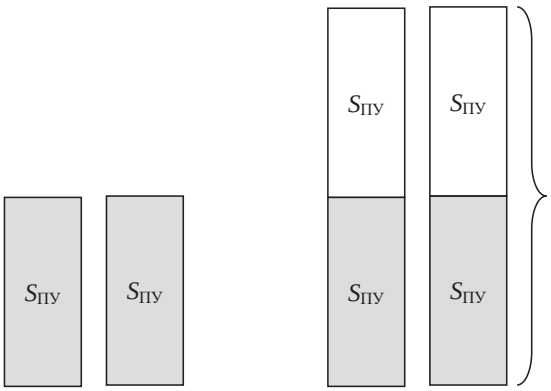
что на величину $2(P_{\text{ПУ}} - P_{\text{ПУ}}^2)$ больше, чем в случае, когда производительность ПУ равна $D_{\text{ПУ}} = S$.

Рассмотрим теперь случай, когда в решении задачи $\boldsymbol{G(Q,X)}$ задействовано три ПУ с производительностью $D_{\text{ПУ}} = S$. При этом время решения подзадач графа $\boldsymbol{G(Q,X)}$ на таком СУ ВС приблизительно составит

$$T_B = \frac{\sum_{i=1}^{M} v(O_i)}{3S_{\text{ПУ}}},$$

а вероятность безотказной работы ВУ будет равна

$$P_{\text{в}} = P_{\text{ПУ}}^3.$$



СУ ВС, содержащая 2 ПУ с производительностью $D_{\text{ПУ}} = S$

СУ ВС, содержащая 2 ПУ с производительностью $D_{\text{ПУ}} = 2S$

Рис. 4.19. Повышение отказоустойчивости за счет увеличения производительности ПУ

Но если мы заменим в таком устройстве ПУ с производительностью $D_{\text{ПУ}} = S$ на ПУ с производительностью $D_{\text{ПУ}} = 2S$, то при сохранении того же времени вычислений T_{max} вероятность безотказной работы СУ ВС повышается и будет уже в данном случае составлять

$$P_{\text{в}} = 1 - (1 - P_{\text{ПУ}})^3 - 3 \cdot P_{\text{ПУ}}(1 - P_{\text{ПУ}})^2 = 3 \cdot P_{\text{ПУ}}^2 - 2 \cdot P_{\text{ПУ}}^3.$$

При этом СУ ВС, состоящее из трех ПУ с производительностью $D_{\text{ПУ}} = 2S$, обеспечит парирование отказа одного ПУ, поскольку два оставшихся смогут обеспечить решение своих подзадач, а также подзадач

вышедшего из строя ПУ за то же время T_{max} (рис. 4.20). При этом вероятность безотказной работы такой системы составит

$$P_{\text{В}} = P_{\text{ПУ}}^3 + 3P_{\text{ПУ}}^2(1 - P_{\text{ПУ}}) = P_{\text{ПУ}}^2(3 - 2 \cdot P_{\text{ПУ}}).$$

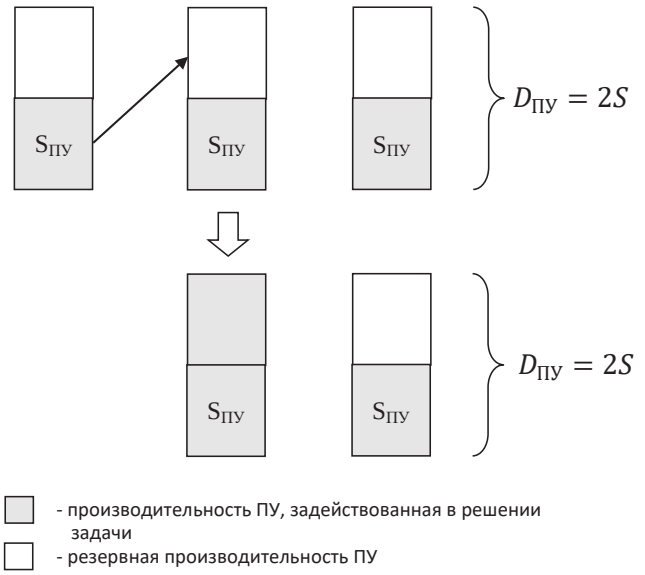


Рис. 4.20. Перераспределение задач в случае отказа одного ПУ

Заметим, что в данном случае для обеспечения парирования одного отказа производительность ПУ не обязательно нужно повышать в два раза. Достаточно повысить производительность ПУ до величины (рис.4.21)

$$D_{\text{ПУ}} = \frac{3}{2}S.$$

Если продолжить рассуждения далее, то можно показать, что СУ ВС, содержащая N ПУ с производительностью

$$D_{\text{ПУ}} = S + \frac{1}{N-1}S = \frac{N}{N-1}S,$$

будет парировать один отказ и обеспечивать вероятность безотказной работы

$$P_{\text{В}} = P_{\text{ПУ}}^H + HP_{\text{ПУ}}^{H-1} \cdot (1 - P_{\text{ПУ}}).$$

Для того чтобы парировать B отказов, производительность всех N ПУ необходимо повысить до величины

$$D_{\text{ПУ}} = S + \frac{B}{N-B}S = \frac{N}{N-B}S.$$

При этом вероятность безотказной работы системы будет равна

$$P_{\text{В}} = \sum_{k=0}^{B} C_H^k P_{\text{ПУ}}^{H-k}(1 - P_{\text{ПУ}})^k.$$

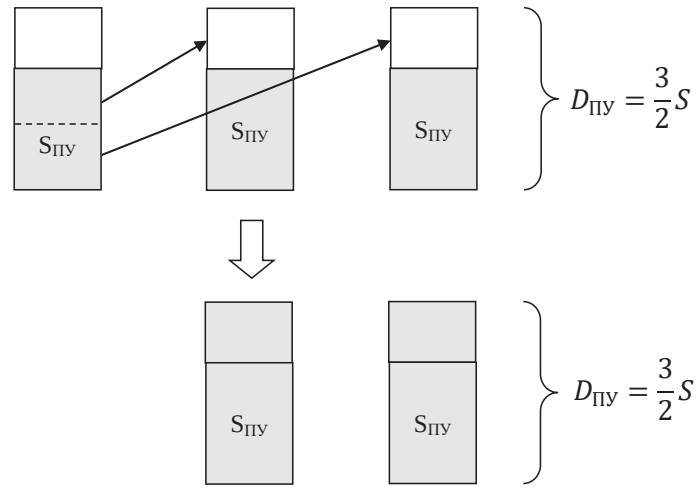


Рис. 4.21. Перераспределение задач в случае отказа одного ПУ

Необходимо отметить, что приведенные выше рассуждения несколько идеализированы, поскольку предполагается, что задачи, решаемые на отдельных ПУ, можно «дробить» на отдельные подзадачи с произвольным «квантом». Однако несмотря на это допущение, сам подход к повышению надежности за счет резервной (избыточной) производительности ПУ применим и в общем случае.

Однако очевидно, что при использовании данного подхода значительно усложняется процедура восстановления вычислительного процесса в СУ ВС, поскольку в случае возникновения отказа какого либо ПУ необходимо каким-то образом осуществить перераспределение решаемых им подзадач задачи управления на работоспособные ПУ, а не просто перенести их с отказавшего ПУ на резервное, как это осуществляется при использовании способа структурного резервирования. Поскольку, как правило, на время $T_{ввп}$ восстановления вычислительного процесса наложены ограничения, то время выполнения процедуры перераспределения задач по работоспособным ПУ (реконфигурации СУ ВС) также должно быть ограничено, т. е. $T_{рек} \leq T_{ввп}$, где $T_{рек}$ – время реконфигурации СУ ВС при возникновении отказа, $T_{ввп}$ – заданное время восстановления вычислительного процесса в СУ ВС. Поскольку, как показано выше, СУ ВС на базе архитектуры ИМА относится к классу гомогенных распределенных систем, то данное обстоятельство открывает возможности использования методов и алгоритмов мультиагентного диспетчирования ее ресурсов в процессе реализации процедуры реконфигурации, которые, как показано в подразделе 1.2, позволяют

существенно снизить временные затраты по сравнению с другими способами диспетчирования.

Для этого на каждом ПУ $(i = 1,2,...,N)$, входящем в состав СУ ВС, должен быть реализован агент AR_i (рис. 4.22), выполняющий следующие основные функции:

1) функцию тестирования;
2) функцию реконфигурации;
3) функцию решения задачи управления.

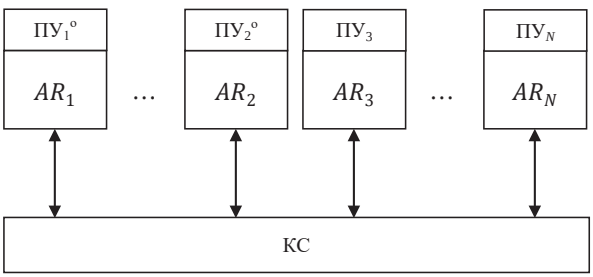


Рис. 4.22. СУ ВС с мультиагентным диспетчером

При реализации функции тестирования агент AR_i $(i = 1, 2, \dots, N)$ должен выполнять следующие действия.

1. Проводить диагностирование работоспособности «своего» ПУ$_i$.

2. Пересылать диагностическое сообщение о работоспособности «своего» ПУ$_i$ всем остальным ПУ$_j$ $(j = 1, 2, \dots, i-1, i+1, \dots, N)$, входящим в состав СУ ВС.

3. Принимать сообщения от агентов других ПУ$_j$ $(j = 1, 2, \dots, i-1, i+1, \dots, N)$, подтверждающие их работоспособность.

4. В случае если агент некоторого ПУ, задействованного в решении задачи управления, не подтвердил свою работоспособность, переходить в режим реконфигурации.

5. Если же агенты всех ПУ подтвердили свою работоспособность, то переходить в режим решения задачи управления.

При реализации функции реконфигурации агент AR_i $(i = 1,2,...,N)$ должен выполнять следующие основные действия.

1. Принимать участие в процедуре перераспределения (размещения) операционных вершин графа $\boldsymbol{G(Q, X)}$ по работоспособным ПУ.

2. По завершении процедуры перераспределения (размещения) переходить в режим решения задачи управления.

При реализации функции решения задачи управления агент AR_i $(i = 1,2,...,N)$ должен выполнять следующие основные действия.

1. Получать от других ПУ или информационных подсистем исходные данные, необходимые для решения множества подзадач $\boldsymbol{O}_i$, закрепленных за ПУ$_i$.

2. Решать подзадачи множества $\boldsymbol{O}_i$.

3. Передавать данные, полученные в результате решения подзадач множества $\boldsymbol{O}_i$, на другие ПУ или на исполнительные устройства в соответствии с топологией связей вершин графа $\boldsymbol{G}(\boldsymbol{Q},\boldsymbol{X})$.

4. Переходить в режим тестирования.

Очевидно, что наибольшую сложность при этом представляет работа агента AR_i $(i=\overline{1,N})$ в режиме реконфигурации. Как уже сказано выше, от времени выполнения процедуры реконфигурации зависит время $T_{\text{ввп}}$ восстановления вычислительного процесса в СУ ВС. Поскольку, как правило, на время $T_{\text{ввп}}$ наложены ограничения, то время выполнения процедуры реконфигурации также должно быть ограничено, т. е. $T_{\text{рек}} \le T_{\text{ввп}}$, где $T_{\text{ввп}}$ – заданное время восстановления вычислительного процесса в СУ ВС. С учетом этого рассмотрим методы самоорганизации СУ ВС при реализации процедуры реконфигурации более подробно.

Допустим, что в некоторый момент времени из строя выходят L ПУ ($L < N$). Тогда задача реконфигурации СУ ВС сводится к перераспределению подзадач, приписанных вершинам графа $\boldsymbol{G}(\boldsymbol{Q},\boldsymbol{X})$ между $N-L$ работоспособными ПУ, таким образом, чтобы обеспечить режим реального времени при решении задачи управления, т. е. обеспечить решение задачи управления за время T_{max}.

Поскольку, как мы приняли, все ПУ$_i$ $(i=1,2,...,N)$ имеют одинаковую производительность $D_{\text{ПУ}}$, то выполнение всех операций, приписанных вершинам ветви $\boldsymbol{H}_f$ графа $\boldsymbol{G}(\boldsymbol{Q},\boldsymbol{X})$ задачи управления СУ ВС, целесообразно осуществлять с помощью одного и того же ПУ$_i$. При этом, если требуемый момент времени T_{k+1}^f исполнения последней вершины q_k^f ветви $\boldsymbol{H}_f$ известен, то зная период времени $t_p\left(O_i^f\right)$ $(i=1,2,...,k)$ выполнения ПУ$_i$ отдельных операций, приписанных вершинам ветви $\boldsymbol{H}_f$, можно, соответственно, определить требуемый момент времени начала выполнения операции O_d^f, приписанной любой вершине $q_d^f \in \boldsymbol{H}_f$ $(d=1,2,...,k)$ как (рис. 4.22):

$$T_d^f = T_{k+1}^f - \left(\sum_{j=d}^{k} t_p(O_j^f) + \frac{W\left(o_k^f\right)}{Y} \right), \qquad (4.10)$$

где $t_p\left(O_j^f\right) = \frac{v\left(o_j^f\right)}{D_{\text{ПУ}}}$;

$W\left(O_k^f\right)$ – объем данных, приписанных конечной дуге ветви $\boldsymbol{H}_f$;

Y – пропускная способность канала связи ПУ$_i$.

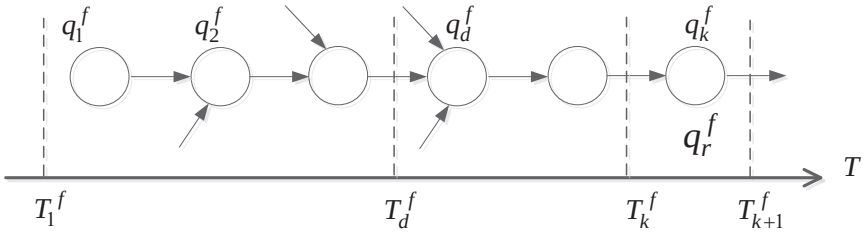


Рис. 4.22. Моменты времени начала выполнения операций, приписанных вершинам ветви $\boldsymbol{H}_f$

Отметим, что изначально требуемый момент времени T_{max} приписан только конечной вершине $q_k \in \boldsymbol{Q}$ графа $\boldsymbol{G}(\boldsymbol{Q},\boldsymbol{X})$, причем этот момент времени определяется требованиями режима реального времени при решении задач управления.

Исходя из этих соображений, можно предложить следующую процедуру реконфигурации СУ ВС при возникновении отказа некоторых ПУ.

После обнаружения отказа L ПУ агенты работоспособных ПУp $(p = 1, 2, \dots, N - L)$ в некотором порядке (например, в порядке нумерации) оценивают возможность своего участия в сообществе $\boldsymbol{R}$ по решению задачи управления. Для этого агент AR_p выделяет в графе задачи управления $\boldsymbol{G^1}(\boldsymbol{Q^1},\boldsymbol{X^1}) = \boldsymbol{G}(\boldsymbol{Q},\boldsymbol{X})$ наиболее длинную ветвь $\boldsymbol{H}_1 =< q_1^1, q_2^1, \dots, q_k^1 >$, конечной вершине которой приписан момент времени ее исполнения T_{k+1}^f . Отметим, поскольку в начальный момент времени реализации процедуры реконфигурации требуемый момент времени T_{max} исполнения приписан только конечной вершине q_k графа $\boldsymbol{G^1}(\boldsymbol{Q^1},\boldsymbol{X^1})$, то ветвь $\boldsymbol{H}_1 =< q_1^1, q_2^1, \dots, q_k^1 >$ должна заканчиваться на конечной вершине графа $\boldsymbol{G^1}(\boldsymbol{Q^1},\boldsymbol{X^1})$, т. е. $q_k^1 = q_k$.

На основании данных о периодах времени $t_p(O_i^1)$ $(i = 1,2,\dots,k)$ выполнения отдельных операций, входящих в выделенную ветвь $\boldsymbol{H}_1$, агент AR_p ПУp определяет, согласно (4.10), момент времени T_1^1, когда необходимо приступить к выполнению первой операции ветви $\boldsymbol{H}_1$ (т. е. операции, приписанной вершине $q_1^1 \in \boldsymbol{H}_1$), чтобы успеть завершить исполнение всей ветви $\boldsymbol{H}_1$ к заданному моменту времени T_{k+1}^1, как:

$$T_1^1 = T_{k+1}^1 - \left(\sum_{i=d}^{k} t_p(O_i^1) + \frac{w(o_k^f)}{Y} \right),$$

где $t_p(O_i^1) = \frac{v(O_i^1)}{D_p(O_i^1)}$ $(d = 1,2,\dots,k)$.

Если при этом оказывается, что $T_1^1 < T_{\text{тек}}$, где $T_{\text{тек}}$ – текущий момент времени, T_1^1 – требуемый момент времени начала выполнения первой операции ветви $\boldsymbol{H}_1$, то это означает, что ПУ$_p$ не может обеспечить выполнение всей последовательности операций ветви $\boldsymbol{H}_1 =< q_1^1, q_2^1, ..., q_k^1 >$ к установленному моменту времени T_{k+1}^1. Поскольку полагается, что все ПУ одинаковы и выполняют идентичные операции за одинаковое время, то это также означает, что никакой другой ресурс $R_j \in \boldsymbol{R}$ $(j = 1,2,..., p-1, p+1,..., N)$ также не сможет выполнить данную ветвь $\boldsymbol{H}_1$ к установленному моменту времени T_{k+1}^1. Это означает, что задача управления не может быть решена за время T_{max}, и поэтому надо либо каким-то образом «исключать» второстепенные подзадачи, входящие в общую задачу управления ВС, либо отправлять СУ ВС в ремонт.

Если же условие $T_1^l \geq T_{\text{тек}}$ выполняется, то агент AR_p ПУ$_p$ принимает на себя исполнение последовательности операций, приписанных вершинам ветви $\boldsymbol{H}_1$. При этом агент AR_p осуществляет модификацию графа задания $\boldsymbol{G^1(Q^1, X^1)}$, а именно:

1. Всем вершинам $q_i^1 \in \boldsymbol{H}_1$ ветви $\boldsymbol{H}_1$ приписываются моменты времени T_d^1 начала их исполнения, определяемые как

$$T_d^1 = T_{k+1}^1 - \left(\sum_{i=d}^{k} t_p(O_i^1) + \frac{W(o_k^f)}{Y} \right) (d = 1,2,...,k),$$

где O_i^1 – операция, приписываемая вершине $q_i^1 \in \boldsymbol{H}_1$.

2. Вершины, входящие в ветвь $\boldsymbol{H}_1$, исключаются из графа $\boldsymbol{G^1(Q^1, X^1)}$, в результате чего формируется новый граф $\boldsymbol{G^2(Q^2, X^2) = G^1(Q^1, X^1)/H}_1$ (рис. 2.4).

3. Всем вершинам модифицированного графа $\boldsymbol{G^2(Q^2, X^2)}$, инцидентным вершинам ветви $\boldsymbol{H}_1$, приписываются требуемые моменты времени их исполнения, которые определяются исходя из следующих соображений. Допустим, что некоторая вершина q_f^2 графа $\boldsymbol{G^2(Q^2, X^2)}$ инцидентна вершине q_b^1, принадлежащей ветви $\boldsymbol{H}_1$. Это означает, что для выполнения операции, приписанной вершине q_b^1, необходимы результаты выполнения подзадач, приписанных вершине q_f^2 графа $\boldsymbol{G^2(Q^2, X^2)}$. Поэтому, очевидно, что результаты выполнения операции, приписанной вершине q_f^2, должны быть получены и переданы ПУ$_p$, выполняющему подзадачи ветви $\boldsymbol{H}_1$, не позже чем к требуемому моменту времени окончания выполнения вершины q_{b-1}^1, т. е.

$$T_{f+1}^2 = T_{k+1}^1 - \left(\sum_{i=b}^{k} t_p(O_i^1) + \frac{W(O_k^1)}{Y} \right). \qquad (4.11)$$

Поэтому вершине q_f^2 графа $\boldsymbol{G^2(Q^2, X^2)}$ приписывается требуемое время ее исполнения $T_{f+1}^2 = T_b^1$, а также номер процессора ПУ$_p$, которому результаты исполнения операции q_f^2 должны быть переданы . Аналогичным образом определяются и приписываются всем остальным вершинам графа $\boldsymbol{G^2(Q^2, X^2)}$, инцидентным вершинам ветви $\boldsymbol{H}_1$, требуемые моменты времени T_{m+1}^2 их исполнения.

Если после модификации новый граф $\boldsymbol{G^2(Q^2, X^2)}$ задачи управления еще не пустой, т. е. $\boldsymbol{G^2(Q^2, X^2)} \neq \emptyset$, то процесс реконфигурации продолжается далее. Следующий по очереди (по номеру) агент AR_c работоспособного ПУ$_c$ выделяет в графе $\boldsymbol{G^2(Q^2, X^2)}$ наиболее длинную ветвь $\boldsymbol{H}_2 =< q_1^2, q_2^2, ..., q_f^2 >$, конечной вершине q_f^2 которой приписано требуемое время исполнения T_{f+1}^2, и определяет, согласно выражению (4.10), момент времени, когда необходимо приступить к ее выполнению, как:

$$T_1^2 = T_{f+1}^1 - \left(\sum_{i=1}^{f} t_c\left(O_i^2\right) + \frac{W(O_k^2)}{Y} \right),$$

где T_{f+1}^2 – момент времени, приписанный конечной вершине q_f^2 ветви $\boldsymbol{H}_2$;

O_i^2 – операция, приписанная вершине $q_i^2 \in \boldsymbol{H}_2$ $\boldsymbol{(}i = 1,2,..., f\boldsymbol{)}$.

Если $T_1^2 < T_{\text{тек}}$, где $T_{\text{тек}}$ – текущий момент времени, то это означает, что данная ветвь не может быть выполнена к установленному моменту времени T_{f+1}^2, а вся задача управления не может быть решена с помощью оставшихся $(N - L)$ работоспособных ПУ за время T_{max}. В этом случае необходимо либо исключить второстепенные подзадачи из графа задачи управления и осуществить процедуру реконфигурации заново с учетом «усеченного» таким образом графа $\boldsymbol{G(Q, X)}$, либо осуществить ремонт СУ ВС, заменив отказавшие ПУ на работоспособные.

В противном случае агент AR_c принимает на себя исполнение операций, приписанных вершинам нити $\boldsymbol{H}_2$, и осуществляет очередную модификацию графа задачи управления $\boldsymbol{G^2(Q^2, X^2)}$.

1. Всем вершинам $q_d^2 \in \boldsymbol{H}_2$ нити $\boldsymbol{H}_2$ приписываются моменты времени T_d^2 начала их исполнения, определяемые как

$$T_d^2 = T_{f+1}^2 - \left(\sum_{i=d}^{f} t_c(O_i^2) + \frac{W(O_k^2)}{Y} \right) \boldsymbol{(}d = 1,2,..., f\boldsymbol{)}.$$

2. Вершины, входящие в ветвь $\boldsymbol{H}_2$, исключаются из графа $\boldsymbol{G^2(Q^2, X^2)}$, в результате чего образуется новый граф $\boldsymbol{G^3(Q^3, X^3) = G^2(Q^2, X^2)/H}_2$.

3. Каждой вершине q_p^3 графа $\boldsymbol{G^3(Q^3, X^3)}$, инцидентной вершине q_d^2 ветви $\boldsymbol{H}_2$, приписывается требуемый момент времени ее исполнения, определяемый как

$$T_{p+1}^3 = T_{f+1}^2 - \left(\sum_{i=d}^{f} t_c(O_i^2) + \frac{W(O_k^2)}{Y} \right),$$

а также номер ПУ, которому необходимо передать результаты исполнения данной операции.

Далее в процесс распределения подзадач задачи управления включается следующий по очереди агент работоспособного ПУ$_c$, который выделяет на модифицированном агентом AR_c графе $\boldsymbol{G^3(Q^3, X^3)}$ самую длинную ветвь $\boldsymbol{H}_3$ и т. д. до тех пор, пока не окажется, что после очередной модификации граф $\boldsymbol{G^j(Q^j, X^j)}$ стал пустым, что означает, что все подзадачи задачи управления разобраны агентами ПУ.

В результате описанной выше процедуры самоорганизации (реконфигурации) все подзадачи задачи управления ВС будут перераспределены по $N - L$ работоспособным ПУ, причем время решения задачи управления не будет превышать предельно допустимого T_{max}.

Описанной выше процедуре реконфигурации отвечает следующий укрупненный алгоритм работы агентов ПУ.

Алгоритм 4.4

1. $j = 1$; $\boldsymbol{G^j(Q^j, X^j)} = \boldsymbol{G(Q, X)}$.

2. Агент AR_j работоспособного ПУ$_j$ выделяет в графе $\boldsymbol{G^j(Q^j, X^j)}$ наиболее длинную ветвь $\boldsymbol{H}_j =< q_1^j, q_2^j, ..., q_k^j >$, конечной вершине q_k^j которой приписан требуемый момент времени ее исполнения T_{k+1}^j ($j = 1$, то $T_{k+1}^j = T_{max}^l$), и определяет допустимый момент времени, когда необходимо начать ее выполнение как

$$T_1^j = T_{k+1}^j - \left(\sum_{i=1}^{k} t_p(O_i^j) + t_{\text{п}} \right), \text{ где } t_p(O_i^j) = \frac{v(O_i^j)}{D_p(O_i^j)}.$$

3. Если $T_1^j < T_{\text{тек}}$, где $T_{\text{тек}}$ – текущий момент времени, то перейти к 8.

4. Агент AR_j принимает на себя выполнение операций ветви $\boldsymbol{H}_j$ и модифицирует граф $\boldsymbol{G^j(Q^j, X^j)}$ задачи управления:

– всем вершинам $q_d^j \in \boldsymbol{H}_j$ $(d = 1, 2, ..., k)$ приписывается момент времени T_d начала их исполнения

$$T_{d+1} = T_{k+1}^f - \left(\sum_{i=d}^{f} \frac{v(o_i^f)}{D_p(o_i^f)} + \frac{W(o_k^f)}{Y_p} \right);$$

– вершины ветви $\boldsymbol{H}_j$ исключаются из графа $\boldsymbol{G^j(Q^j, X^j)}$, в результате чего формируется новый граф $\boldsymbol{G^{j+1}(Q^{j+1}, X^{j+1}) = G^j(Q^j, X^j)/H_j}$;

– каждой вершине q_b^{j+1} модифицированного графа $\boldsymbol{G^{j+1}(Q^{j+1}, X^{j+1})}$, инцидентной вершине q_d^j ветви $\boldsymbol{H}_j$, требуемый момент времени ее исполнения, определяемый как

$$T_{d+1} = T_{k+1}^f - \left(\sum_{i=b}^{k} \frac{v(O_i^f)}{D_p(O_i^f)} + \frac{W(O_k^f)}{Y_p} \right),$$

где q_d – вершина графа $\boldsymbol{G(Q, X)}$ инцидентная вершине q_b^f ветви $\boldsymbol{H}_f$ а также номер ПУ$_j$, которому необходимо передать результаты исполнения операции, приписанной вершине q_b^{i+1}.

5. Если $\boldsymbol{G^{j+1}(Q^{j+1}, X^{j+1}) \neq \emptyset}$, то процесс распределения операций задания Z_l заканчивается.

6. Агент AR_j запускает на ПУ$_j$ процедуру выполнения последовательности операций, приписанных вершинам ветви $\boldsymbol{H}_j$ согласно установленному временному графику (т. е. приписанным вершинам ветви $\boldsymbol{H}_j$ моментам времени начала их исполнения).

7. $j = j + 1$. Если $j \leq N - L$, то перейти к 2, иначе

8. Формируется сообщение о необходимости ремонта СУ ВС или исключения из задачи управления некоторого множества второстепенных подзадач.

9. Конец.

Реализация представленного выше алгоритма реконфигурации подразумевает полное перераспределение всех ветвей графа задачи управления $\boldsymbol{G(Q, X)}$ между $N - L$ работоспособными ПУ, что в свою очередь может потребовать значительного времени, превышающего допустимое время $T_{\text{ввп}}$ восстановления вычислительно процесса в СУ ВС. Время реконфигурации СУ ВС можно существенно сократить, если перераспределять заново не все ветви графа задачи управления $\boldsymbol{G(Q, X)}$ по работоспособными ПУ, а только те ветви, которые ранее исполнялись отказавшими ПУ.

Для этого, прежде всего, необходимо сформировать список $\{\boldsymbol{H^p}\} = \{\boldsymbol{H_1^p, H_2^p, \dots, H_L^p}\}$ ветвей графа $\boldsymbol{G(Q, X)}$, которые выполнялись отказавшими ПУ и требуют перераспределения на $N - L$ работоспособных ПУ. Далее каждый из работоспособных ПУ$_i$ $(i = 1, 2, \dots, N - L)$ должен в порядке некоторой очереди (например, в порядке нумерации) оценить возможность включения в список $\{\boldsymbol{H^i}\}$ выполняемых им ветвей графа $\boldsymbol{G(Q, X)}$ дополнительной ветви $\boldsymbol{H_j^p} \subseteq \{\boldsymbol{H^p}\}$. При этом ветвь $\boldsymbol{H_j^p}$ может быть включена в список $\{\boldsymbol{H^i}\}$ ветвей, исполняемых процессором ПУ$_i$, если имеющийся у него резерв производительности позволяет решить все

подзадачи этой ветви $\boldsymbol{H}_j^p$ за время, не превышающее T_{max}, т. е. должно выполняться условие

$$\frac{\sum_{k=1}^{K} V(\boldsymbol{H}_k^i)+V(\boldsymbol{H}_j^p)}{D_{\text{пу}}} \le T_{max}, \qquad (4.12)$$

где $V(\boldsymbol{H}_k^i)$ – суммарная трудоемкость подзадач, приписанных вершинам ветви $\boldsymbol{H}_k^i$, ранее включенной в список$\{\boldsymbol{H}^i\}$;

K – число ветвей в списке $\{\boldsymbol{H}^i\}$;

$V(\boldsymbol{H}_j^p)$ – суммарная трудоемкость подзадач, приписанных вершинам ветви $\boldsymbol{H}_j^p$;

$D_{\text{пу}}$ – производительность ПУ$_i$.

Исходя из этих соображений, можно предложить следующий алгоритм реконфигурации СУ ВС при отказе L ПУ.

Алгоритм 4.5

1. $i = 1; j = 1$.

2. Агент AR_i ПУ$_i$ выбирает из списка $\{\boldsymbol{H}^p\}$ ветвей графа $\boldsymbol{G}(\boldsymbol{Q},\boldsymbol{X})$, требующих перераспределения, ветвь $\boldsymbol{H}_j^p$.

3. Если условие (4.12) не выполняется, то перейти к 7, иначе

4. Ветвь $\boldsymbol{H}_i^p$ включается в список $\{\boldsymbol{H}^i\}$, выполняемых ПУ$_i$ ветвей графа $\boldsymbol{G}(\boldsymbol{Q},\boldsymbol{X})$, т. е. $\{\boldsymbol{H}^i\} = \{\boldsymbol{H}^i\} \cup \boldsymbol{H}_i^p$, и одновременно исключается из списка $\{\boldsymbol{H}^p\}$ ветвей требующих перераспределения, т. е. $\{\boldsymbol{H}^p\} = \{\boldsymbol{H}^p\}/\boldsymbol{H}_i^p$.

5. Если $\{\boldsymbol{H}^p\} \neq \emptyset$, то перейти в 9 иначе

6. $j = j + 1$, если $j \le L$, то перейти к 2 иначе

7. $i = i + 1$, если $i \le N - L$, то перейти к 2, иначе

8. Если $\{\boldsymbol{H}^p\} \neq \emptyset$, то реконфигурация СУ ВС невозможна. При этом необходимо либо исключить из задачи управления второстепенные подзадачи и запустить процесс реконфигурации заново, либо отправить СУ ВС в ремонт. Перейти к 10.

9. Процесс реконфигурации успешно завершен.

10. Конец.

Список литературы

1. *George Coulouris, Jean Dollimore, Tim Kindberg,* "Distributed Systems Concepts and Design" 3rd edition, Addison-Wesley. Blaze M. et al. The role of trust management in distributed systems security // Secure Internet Programming. Springer Berlin Heidelberg, 1999.

2. *Лопота А.В., Юревич Е.И.* Самоорганизация в кибернетике и робототехнике // Робототехника и техническая кибернетика. – 2014. – №. 4. – С. 4–5.

3. *Граничин О.Н.* Круглый стол «Самоорганизация и искусственный интеллект в группе автономных роботов: методология, теория, практика» // Стохастическая оптимизация в информатике. – 2020. – Т. 16. – №. 1. – С. 5–12.

5. *Кузнецова В. Л., Раков М. А.* Самоорганизация в технических системах. – Наукова думка, 1987. – 200 С.

6. *Смелянский Р.Л.* Модель функционирования распределенных вычислительных систем //Вестн. Моск. Ун-та. сер. 1990. Т. 15. – С. 3–21.

7. *Li K. et al.* Scheduling precedence constrained stochastic tasks on heterogeneous cluster systems // IEEE Trans. Comput. 2015. Vol. 64, № 1. P. 191–204.

8. *Мельник Эдуард Всеволодович, Клименко Анна Борисовна.* «Комплексный метод организации управления отказоустойчивой информационно-управляющей системой на основе многоагентного взаимодействия». – Известия Тульского государственного университета. Технические науки, 9-1, 2017, с. 136–149.

9. *Anatoly Kalyaev; Korovin I.,* New Method to Use Idle Personal Computers for Solving Coherent Tasks // 2014 AASRI Conference on Circuit and Signal Processing (CSP 2014) Volume 9, 2014, P. 131–137.

10. *W. R. Ashby* Principles of the Self-Organizing Dynamic System, «Journal of General Psychology», 1947, vol. 37, p. 125–128.

11. *Виленкин Н.Я.* Глава III. Комбинаторика кортежей и множеств. Размещения с повторениями // Популярная комбинаторика. – М.: Наука, 1975. – С. 80. – 208 с.

12. *Куприянов М.С., Кочетков А.В.* Мультиагентная модель самоорганизующейся распределенной системы // Известия СПбГЭТУ «ЛЭТИ». – 2016. – №. 2. – С. 12.

13. *М.И. Гераськин, Л.С. Клентак* Линейное программирование. – Самара: Изд-во СГАУ, 2014. – 104 с.: ISBN 978-5-7883-0979-8

14. *Беллман Р.* Динамическое программирование. – М.: Издательство иностранной литературы, 1960.

15. *Танаев В.С., Шкурба В.В.* Введение в теорию расписаний. – Москва: Главная редакция физико-математической литературы изд-ва «Наука», 1975.

16. *Семенистый С.А., Каляев А.И.* Алгоритм работы досок объявлений распределенной автоматизированной системы прогнозирования отказов нефтепромыслового оборудования // Наука и современность. Сборник материалов V-й международной научно-практической конференции. 2016. – С. 55–56.

17. *David Karger, Rajeev Motwani,* G.D.S. Ramkumar. On approximating the longest path in a graph // Algorithmica. – 1997. – Т. 18, вып. 1. – С. 82–98. –doi:10.1007/BF02523689.

18. *Каляев А.И., Каляев И.А., Коровин Я.С.* Распределение ресурсов в облачной среде при выполнении потока произвольно поступающих задач // Робототехника и техническая кибернетика. 2016. № 1 (10). – С. 18–26.

19. *Капустян С., Каляев И., Гайдук А.* Модели и алгоритмы коллективного управления в группах роботов. – Litres, 2018. – 289 с.

20. *Каляев А.И., Каляев И.А.* Метод мультиагентного диспетчирования ресурсов в облачных вычислительных средах // Известия Российской академии наук. Теория и системы управления. 2016. № 2. – С. 51.

21. *Anatoly Kalyaev* Multiagent Approach for Building Distributed Adaptive Computing System // Procedia Computer Science, Volume 18, 2013, Pages 2193-2202 ISSN: 1877-0509

22. *Каляев А.И., Каляев И.А.* Метод децентрализованного управления распределенной системой при выполнении множества заданий // Мехатроника, автоматизация, управление. 2015. Т. 16. № 9. – С. 585–598.

23. *Каляев А.И., Каляев И.А., Коровин Я.С.* Метод мультиагентного диспетчирования ресурсов в гетерогенной облачной среде при выполнении множества задач // Вестник компьютерных и информационных технологий. 2015. № 11 (137). – С. 31–40.

24. *Дунаев Д.Н.*. «Виды задач линейного программирования, способы их решения». Наука и современность, № 21, 2013, с. 121–125.

25. *Kalyaev A.I., Kalyaev I.A.* Method Of Multiagent Scheduling Of Resources In Cloud Computing Environments // Journal of Computer and Systems Sciences International. 2016. Т. 55. № 2. С. 211–221.

26. *Каляев И.А., Каляев А.И., Коровин Я.С.* Принципы организации и функционирования безлюдного роботизированного производства с децентрализованным диспетчером // Мехатроника, автоматизация, управление. – 2016. – Т. 17. № 11. – С. 741–749.

27. *Аскаров Даурен Тулегенович, and Бакытжан Даулет Алимуратулы.* «Экономические аспекты внедрения концепции безлюдных производств». –Наука без границ, no. 3 (8), 2017. – С. 16–21.

28. *Ничипорук А.О.* «ОПЫТ И ПРОБЛЕМЫ ПОСТРОЕНИЯ ТРАНСПОРТНО-ЛОГИСТИЧЕСКИХ СИСТЕМ ДОСТАВКИ ГРУЗОВ». – Научные проблемы водного транспорта, № 50, 2017. – С. 212–218.

29. *Каляев И.А., Каляев А.И., Коровин Я.С.* Алгоритм мультиагентного диспетчирования ресурсов в гетерогенной облачной среде // Вычислительные технологии. – 2016. – Т. 21. № 5. – С. 38–53.

30. *Белоножко Павел Петрович, Белоус Валентина Владимировна, Куцевич Надежда Александровна, Храмов Дмитрий Александрович.* «Свободные облачные аппаратно-программные платформы. Аналитический обзор». –Вестник евразийской науки, vol. 8, no. 6 (37), 2016, p. 61.

31. *Балашов Н.А., Баранов А.В., Кадочников И.С., Кореньков В.В., Кутовский Н.А., Нечаевский А.В., Пелеванюк И.С.* Программный комплекс интеллектуального диспетчирования и адаптивной самоорганизации виртуальных вычислительных ресурсов на базе облачного центра ЛИТ ОИЯИ // Известия Южного федерального университета. Технические науки. – 2016. – №. 12 (185). – С. 92–103.

32. *Смирнов Ю.В.* Облачные вычисления: история и влияние на будущее библиотек. Научные и технические библиотеки. 2016; (6):62–73.

33. *Богданов Александр Владимирович, and Е Мьинт Найнг.* «Сравнение нескольких платформ облачных вычислений». – Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления, no. 2, 2013, pp. 102–110.

34. *Гуськов А.Е., Косяков Д.В.* Опыт применения облачных технологий в научной деятельности на примере сибирского отделения РАН // Информационные технологии, системы и приборы в АПК: Материалы 5-й Международной научно-практической конференции «АГРОИНФО-2012» / Сибирский физико-технический институт аграрных проблем Россельхозакадемии; 2012. – Т.:Часть 1. – Краснообск: Государственное научное учреждение Сибирский физико-технический институт аграрных проблем Россельхозакадемии. – С.370–374. – ISBN: 978-5-906143-02-0.

35. *Антимиров В.М.* Поколения бортовых цифровых вычислительных систем [Текст] / *В.М. Антимиров, М.Б. Трапезников* // Вопросы атомной науки и техники.

Серия: Физика радиационного воздействия на радиоэлектронную аппаратуру. – 2012. – №2. – С. 38–46.

36. *Антимиров Владимир Михайлович, Уманский Алексей Борисович, Шалимов Леонид Николаевич.* «Бортовые цифровые вычислительные системы семейства “Малахит” для работы в экстремальных условиях». –Вестник Самарского государственного аэрокосмического университета им. академика С.П. Королёва (национального исследовательского университета), № 4 (42), 2013, с.19–27.

37. *Колпаков, К.М.* История развития бортовых цифровых вычислительных машин в России [Текст] // ITweek. 1999. №32. [Электронный ресурс, режим доступа: https://www.itweek.ru/industrial/article/detail.php?ID=51809, открытый. Дата обращения: 21.07.2022]]:

38. *Федосов, Е., Косьянчук, В., Сельвесюк, Н.* Интегрированная модульная авионика / РАДИОЭЛЕКТРОННЫЕ ТЕХНОЛОГИИ, №1, 2015, С. 66 –71.

39. UAV Navigation AP04 autopilot [Электронный ресурс, режим доступа: https://www.uavnavigation.com/support/kb/autopilots/ap04, открытый. Дата обращения: 21.07.2022],

40. V*aglienti, B.* Piccolo System User’s Guide [Текст]/ *B. Vaglienti, R. Hoag, M. Niculescu,* // Cloud Cap Technologies [Электронный ресурс, режим доступа:http://cloudcaptech.com, открытый. Датаобращения: 21.07.2022]]

41. *Федосов, Е.А.* Проект создания нового поколения интегрированной модульной авионики с открытой архитектурой [Текст] // Полет. – 2008. – №8. – С. 15–22.

42. *Труханов, В.М.* Новый подход к обеспечению надежности сложных систем: монография / *В.М.Труханов.* – М.: Издательский дом «Спектр». – 2010. – 247с.

Научное издание

РОССИЙСКАЯ АКАДЕМИЯ НАУК
Южный Федеральный Университет

А.И. Каляев, И.А. Каляев

САМООРГАНИЗУЮЩИЕСЯ РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ

Издается по решению Научно-издательского совета Российской академии наук (НИСО РАН) от 31.03.2023 г. и распространяется бесплатно

Издатель – Российская академия наук

Публикуется в авторской редакции

Подписано в печать 15.11.2023. Формат 70x100/16. Объем 9,75 п. л.
Гарнитура Times New Roman.
Печать офсетная. Бумага офсетная. Тираж 300 экз. Заказ 23-08761

Отпечатано в ООО «КОНСТАНТА»
308519, Белгородская обл., Белгородский р-н, пос. Северный, ул. Березовая, 1/12.
Тел./факс (4722) 300-720, www.konstanta-print.ru