Информационные процессы, Том 20, № 2, 2020, стр. 116–126

ПЕРЕДАЧА ИНФОРМАЦИИ В КОМПЬЮТЕРНЫХ СЕТЯХ

Оптимизация расходов оператора при распределении вычислительной нагрузки между серверами в интеллектуальных транспортных системах[1]

А.А. Белогаев[*,**], **А.А. Елохин**[***], **А.Н. Красилов**[*,**], **Е.М. Хоров**[*,**]

[*] *Московский физико-технический институт (НИУ), Москва*
[**] *Институт проблем передачи информации им. А.А. Харкевича РАН, Москва*
[***] *Национальный исследовательский университет Высшая школа экономики, Москва*

Поступила в редколлегию 08.06.2020 г.

Аннотация—В данной работе рассматривается задача распределения вычислительной нагрузки между серверами в интеллектуальных транспортных системах. Поставлена задача минимизации расходов оператора при выполнении заданных ограничений по времени передачи по сети и выполнения вычислительных задач на серверах. Предложен алгоритм решения данной задачи оптимизации, основанный на сведении ее к задаче целочисленной линейной оптимизации. Результаты моделирования показывают, что использование предложенного алгоритма позволяет существенно снизить расходы оператора по сравнению с существующими подходами и при этом выполнить строгие ограничения на задержку различных приложений.

КЛЮЧЕВЫЕ СЛОВА: V2X, автомобильные самоорганизующиеся сети, целочисленное линейное программирование, интеллектуальные транспортные системы, периферийные вычисления, распределение вычислительной нагрузки.

1. ВВЕДЕНИЕ

Рынок Интеллектуальных транспортных систем (англ.: Intelligent Transport Systems, ITS) стремительно растёт. Согласно отчету американской консалтинговой компании Grand View Research [1] размер рынка превысил 25 миллиардов долларов в 2018 году и к 2025 году вырастет до 51 миллиарда. В ближайшем будущем нельзя будет представить ни одного города без подобных систем. В их основе лежит работа множества приложений, выполняющих различные функции, такие как: обеспечение безопасности на дороге, координация дорожного трафика, управление транспортными колоннами, сокращение вредных выбросов в атмосферу и другие. Многие приложения подразумевают возможность взаимодействия автомобилей между собой и с окружающей их сетевой инфраструктурой, поэтому их называют приложениями V2X (англ.: vehicle-to-everything). Консорциум 3rd Generation Partnership Project (3GPP), разрабатывающий технологии построения сотовых сетей, в техническом отчёте [2] рассматривает ряд приложений для поддержки в сетях пятого поколения (5G). В данной работе рассматривается три приложения:

– *Управление транспортными колоннами* — приложение, которое позволяет автомобилям, движущимся в одном направлении, объединяться в транспортные колонны. Внутри колонны автомобили могут существенно уменьшить интервалы между друг другом и двигаться как единая система за счет точной координации.

[1] Исследование выполнено в ИППИ РАН за счет гранта Правительства Российской Федерации (Договор No 14.W03.31.0019)

Рис. 1. Архитектура интеллектуальной транспортной системы.

– *Безопасный проезд перекрёстков* — приложение, которое позволяет избежать аварий, возникающих на перекрёстках. В частности, оно может определять очередность проезда, скорость проезда, полосу и т.д.
– *Коллективное восприятие обстановки на дороге* — приложение, позволяющее в реальном времени обнаруживать различные объекты на дороге (например, пешеходов) и выполнять соответствующие манёвры (смена полосы, опережение, объезд, и т.д.) за счёт быстрого обмена информацией между автомобилями и инфраструктурой.

В интеллектуальных транспортных системах взаимодействие автомобилей с инфраструктурой осуществляется с помощью придорожных модулей (англ.: Road Side Unit, RSU), размещаемых вдоль дорог на достаточно близком расстоянии друг от друга и осуществляющих обмен данными с транспортными средствами по беспроводному каналу. Как автомобили, так и придорожные модули могут быть оборудованы камерами, радарами, лидарами и другими приборами, позволяющими получать информацию об обстановке на дороге непосредственно. Для того, чтобы на основе имеющихся данных принять то или иное решение, приложениям V2X требуется провести множество вычислений, которое к тому же должно быть выполнено в кратчайшие сроки. Для того, чтобы это было возможно, организация European Telecommunications Standards Institute (ETSI) предложила концепцию мобильных периферийных вычислений MEC (англ.: Mobile Edge Computing, позже переименованна в Multi-access Edge Computing) [3], позволяющую выполнять вычисления на серверах, расположенных в сети радиодоступа (англ.: Radio Access Network, RAN), например, на придорожных модулях или базовых станциях. На

рис. 1 изображена архитектура интеллектуальной транспортной системы, поддерживающей периферийных вычислений.

Поскольку обслуживание серверов влечет дополнительные расходы, возникает необходимость в оптимальном распределении вычислительной нагрузки между ними с учетом требований к качеству обслуживания для различных приложений. За последние три года был опубликован ряд работ [4–8], посвященных этой тематике. Однако все эти работы решают оптимизационную задачу в краткосрочной перспективе (т.е. по отдельности распределяют каждую вычислительную задачу, возникающую в процессе работы приложений V2X), что не позволяет использовать предложенные решения в случае большого числа автомобилей и, как следствие, большой интенсивности возникающих вычислительных задач. В данной работе предполагается, что решение о распределении вычислительной нагрузки принимается на долгую перспективу и изменяется в течение дня в зависимости от интенсивности автомобильного трафика на дорогах, а также генерируемой ими вычислительной нагрузки.

Дальнейшее изложение материала построено следующим образом. В разделе 2 формулируется математическая постановка задачи. В разделе 3 описывается предлагаемый в работе алгоритм распределения вычислительной нагрузки. В разделе 4 приводятся описание экспериментов и численные результаты. В разделе 5 содержатся заключительные выводы.

2. ПОСТАНОВКА ЗАДАЧИ

В данной работе топология сети интеллектуальной транспортной системы описывается неориентированным графом $G(\mathbb{V}, \mathbb{E})$, где $\mathbb{V}$ обозначает множество вершин, а $\mathbb{E}$ — множество ребер. В качестве вершины графа может выступать любой узел сети: придорожный модуль или базовая станция. Будем считать, что вычислительные серверы установлены на некотором подмножестве $\mathbb{K} \subset \mathbb{V}$ вершин. Для удобства все используемые переменные перечислены в Таблице 1.

Будем моделировать работу V2X-приложений следующим образом. Каждое приложение, запущенное на придорожном модуле, генерирует вычислительные задачи, входными данными которых является собранная информация о текущей обстановке на дороге, а выходными — рекомендации об изменении параметров движения, предназначенные для водителей и/или автономных модулей автомобилей. Пусть $\mathbb{F}$ определяет множество типов вычислительных задач, генерируемых V2X-приложениями. При генерации задачи принимается решение о том, какие ресурсы использовать для вычислений. Это могут быть как серверы, установленные непосредственно на том придорожном модуле, на котором она была сгенерирована, так и серверы, расположенные на других узлах сети. Для удалённого вычисления формируется запрос, содержащий её входные данные, и отправляется на сервер, который будет использоваться для вычисления. По окончании вычислений сервер отправляет ответ, содержащий решение поставленной вычислительной задачи.

V2X-приложения накладывают различные ограничения на задержку при выполнении соответствующих задач. Под задержкой в данном случае подразумевается полное время, затраченное на выполнение задачи, т.е. промежуток времени между генерацией задачи на узле сети i и получением необходимых выходных данных на этом узле. Для задачи типа f, сгенерированной на узле i и вычисляемой с помощью сервера на узле k (далее на сервере k), задержка d_{ifk} состоит из удвоенной задержки передачи данных между узлом сети и сервером t_{ik}, времени ожидания в очереди на сервере W_k и времени выполнения задачи S_k:

$$d_{ifk} = 2t_{ik} + W_k + S_k.$$

В данной работе предполагается, что стохастический процесс генерации приложением на узле сети i потока задач каждого типа $f \in \mathbb{F}$ является потоком Пуассона с интенсивностью

Таблица 1. Переменные

Параметр	Значение
$\mathbb{V}$	множество вершин графа сети
$\mathbb{E}$	множество ребер графа (соединений) сети
$\mathbb{K}$	подмножество вершин графа, на которых установлены серверы
$\mathbb{F}$	множество типов задач
d_{ifk}	полное время, затрачиваемое на выполнение задачи типа f, сгенерированной узлом i и выполняемой на сервере k
t_{ik}	задержка передачи данных между узлом i и сервером k
t_e	задержка передачи данных по соединению e
W_k	время ожидания в очереди на сервере k
S_k	время выполнения одной задачи на сервере k
T_k	время, которое одна задача проводит на сервере k
σ_{if}	интенсивность потока задач типа f, сгенерированных узлом i
μ_k	производительность сервера k
z_{ifk}	индикатор $I\{$поток задач типа f, сгенерированных узлом i, вычисляется на сервере $k\}$
$x_{ifi'f'k}$	вспомогательные переменные $z_{ifk}z_{i'f'k}$
y_k	индикатор $I\{$сервер k используется для вычислений$\}$
h_k	интенсивность запросов на вычисление задач, поступающих на сервер k
$\mathbb{E}_{ik}$	множество соединений на кратчайшем пути между узлом i и сервером k
D_f	ограничение на задержку для задач типа f
q	требуемая вероятность выполнения ограничения на задержку
C_k	расходы оператора на обслуживание сервера k
β_k	расходы на обслуживание сервера k в ненагруженном состоянии
γ_k	расходы на обслуживание сервера k в нагруженном состоянии
α_k	$\gamma_k - \beta_k$

σ_{if}, а время выполнения одной задачи S_k является экспоненциально распределённой случайной величиной, зависящей от производительности сервера μ_k, на котором производятся вычисления. Если задачи типа f не генерируются на узле i, соответствующее значение σ_{if} равно нулю. Поскольку сумма независимых потоков Пуассона является потоком Пуассона, выполнение задач на сервере может быть описано с помощью модели системы массового обслуживания M/M/1 [9].

Распределение потоков задач по серверам определяется матрицей $\boldsymbol{Z} = \{z_{ifk}\}$: если поток задач типа f, генерирующихся на узле i, вычисляется на сервере k, то соответствующий элемент матрицы z_{ifk} равен единице. В противном случае он равен нулю. Введём также вектор $\boldsymbol{Y} = \{y_k\}$, в котором каждый элемент соответствует статусу сервера k: если сервер k используется для вычислений, то соответствующий элемент y_k равен единице, в противном случае он равен нулю.

Переменные $\boldsymbol{Z}$ и $\boldsymbol{Y}$ связаны следующими неравенствами:

$$y_k \geq z_{ifk}, \qquad \forall i \in \mathbb{V}, \forall f \in \mathbb{F}, \forall k \in \mathbb{K}. \tag{1}$$

Так как задачи одного типа f могут использовать одну и ту же контекстную информацию, которую удобно хранить на одном сервере, то будем полагать, что все задачи одного типа f,

сгенерированные одним и тем же узлом i, вычисляются на одном сервере.

$$\sum_k z_{ifk} = 1, \qquad \forall i \in \mathbb{V}, \forall f \in \mathbb{F}. \tag{2}$$

Заметим, что во избежание перегрузки сервера k число запросов на выполнение задач, поступающих на него в единицу времени, не должно превышать μ_k.

$$\sum_{i,f} z_{ifk}\sigma_{if} < \mu_k, \qquad \forall k \in \mathbb{K}. \tag{3}$$

Будем считать, что объем трафика, связанного с выполнением задач, значительно меньше объема остального трафика, генерируемого в сети. Тогда для того, чтобы минимизировать задержку выполнения задач, положим, что трафик V2X-приложений имеет абсолютный приоритет при передаче по каждому соединению. В этом случае можно не рассматривать задачу балансировки нагрузки и считать, что для передачи данных всегда используется кратчайший путь. Отсюда можно оценить:

$$t_{ik} = \sum_{e \in \mathbb{E}_{ik}} t_e,$$

где $\mathbb{E}_{ik}$ — множество соединений на кратчайшем пути от узла i до сервера k, а t_e — задержка передачи данных по соединению e.

Полное время $T_k = W_k + S_k$, проведённое задачей на сервере, является случайной величиной, имеющей экспоненциальное распределение со средним [9, Глава 3]

$$E[T_k] = \frac{1}{\mu_k - h_k} = \frac{1}{\mu_k - \sum\limits_{i',f'} z_{i'f'k}\sigma_{i'f'}}.$$

Будем считать, что ограничение на задержку D_f для задачи типа f выполнено, если полная задержка не превосходит его с вероятностью q:

$$P(d_{ifk} < D_f) \geq q. \tag{4}$$

Так как q-квантиль экспоненциального распределения $p(x) = \lambda exp[-\lambda x]$ со средним $1/\lambda$ равен $(1/\lambda)\ln[1/(1-q)]$, ограничение (4) преобразуется к виду

$$2t_{ik}z_{ifk} + \frac{1}{\mu_k - \sum\limits_{i',f'} z_{i'f'k}\sigma_{i',f'}} \ln\left[\frac{1}{1-q}\right] \leq D_f,$$
$$\forall i \in \mathbb{V}, \forall k \in \mathbb{K}, \forall f \in \mathbb{F}. \tag{5}$$

В данной работе мы рассматриваем семейство функций расходов оператора на обслуживание конкретного сервера, которые линейно зависят от его нагрузки. Функцию расходов можно представить в виде $C_k = \alpha_k h_k/\mu_k + \beta_k$, где $h_k = \sum_{i,f} z_{ifk}\sigma_{if}$ — интенсивность поступающих запросов на выполнение задач, а $\alpha_k > 0$ и $\beta_k > 0$ — коэффициенты, зависящие от расположения и характеристик сервера. Например, потребление энергии каждого сервера линейно зависит от его загруженности [10]. Доля времени, когда сервер k выполняет вычисления, равна h_k/μ_k. Если ввести обозначения β_k и γ_k, соответствующие энергопотреблению сервера k в ненагруженном состоянии и во время вычислений соответственно, то среднее энергопотребление сервера k можно вычислить следующим образом:

$$C_k = \gamma_k\frac{h_k}{\mu_k} + \beta_k(1 - \frac{h_k}{\mu_k}) = \beta_k + (\gamma_k - \beta_k)\frac{h_k}{\mu_k} = \beta_k + \alpha_k\frac{h_k}{\mu_k},$$

где $\alpha_k = \gamma_k - \beta_k$.

Таким образом, задача о распределении вычислительной нагрузки между серверами может быть сформулирована как следующая задача целочисленной оптимизации:

$$\begin{cases} \sum\limits_k \left[y_k \beta_k + \frac{\alpha_k}{\mu_k} \sum\limits_{i,f} z_{ifk} \sigma_{i,f} \right] \to \min \\ \text{При условии } (1) - (3),\ (5). \end{cases} \tag{6}$$

3. ОПИСАНИЕ АЛГОРИТМА РАСПРЕДЕЛЕНИЯ ВЫЧИСЛИТЕЛЬНОЙ НАГРУЗКИ

Задача целочисленной оптимизации (6) не является линейной из-за ограничения (5). Для того, чтобы привести это ограничение к линейному виду, умножим обе части неравенства (5) на $(\mu_k - \sum_{i',f'} z_{i'f'k}\sigma_{i'f'})$:

$$2\mu_k t_{ik} z_{ifk} - 2t_{ik} \sum_{i',f'} z_{ifk} z_{i'f'k} \sigma_{i'f'} + D_f \sum_{i',f'} z_{i'f'k} \sigma_{i'f'} + \ln \left[\frac{1}{1-q} \right] \le \mu_k D_f$$
$$\forall k \in \mathbb{K}, \forall i \in \mathbb{V}, \forall f \in \mathbb{F}.$$

Введем замену переменных:

$$z_{ifk} z_{i'f'k} = x_{ifi'f'k},$$

где новые переменные $x_{ifi'f'k}$ могут принимать значения 0 или 1.

Корректность замены обеспечивается выполнением следующих неравенств:

$$x_{ifi'f'k} \le z_{ifk}, \forall k \in \mathbb{K}, \forall i, i' \in \mathbb{V}, \forall f, f' \in \mathbb{F}, \tag{7}$$

$$x_{ifi'f'k} \le z_{i'f'k}, \forall k \in \mathbb{K}, \forall i, i' \in \mathbb{V}, \forall f, f' \in \mathbb{F}, \tag{8}$$

$$\begin{aligned} x_{ifi'f'k} \ge & z_{i'f'k} + z_{ifk} - 1, \\ & \forall k \in \mathbb{K}, \forall i, i' \in \mathbb{V}, \forall f, f' \in \mathbb{F}. \end{aligned} \tag{9}$$

После замены ограничение (5) имеет следующий линейный вид:

$$2\mu_k t_{ik} z_{ifk} - 2t_{ik} \sum_{i',f'} x_{ifi'f'k} \sigma_{i'f'} + D_f \sum_{i',f'} z_{i'f'k} \sigma_{i'f'} + \ln \left[\frac{1}{1-q} \right] \le \mu_k D_f \ \forall k \in \mathbb{K}, \forall i \in \mathbb{V}, \forall f \in \mathbb{F}. \tag{10}$$

Таким образом, исходная задача (6) может быть сведена к следующей задаче линейной целочисленной оптимизации:

$$\begin{cases} \sum\limits_k \left[y_k \beta_k + \frac{\alpha_k}{\mu_k} \sum\limits_{i,f} z_{ifk} \sigma_{if} \right] \to \min, \\ \text{При условии } (1) - (3),\ (7) - (10). \end{cases} \tag{11}$$

Полученная задача может быть решена классическими методами целочисленного программирования, например, методом ветвей и отсечений (англ.: Branch-and-Cut) [11]. Для получения решения в данной работе используется реализация метода ветвей и отсечений в библиотеке PuLP языка Python 3 [12].

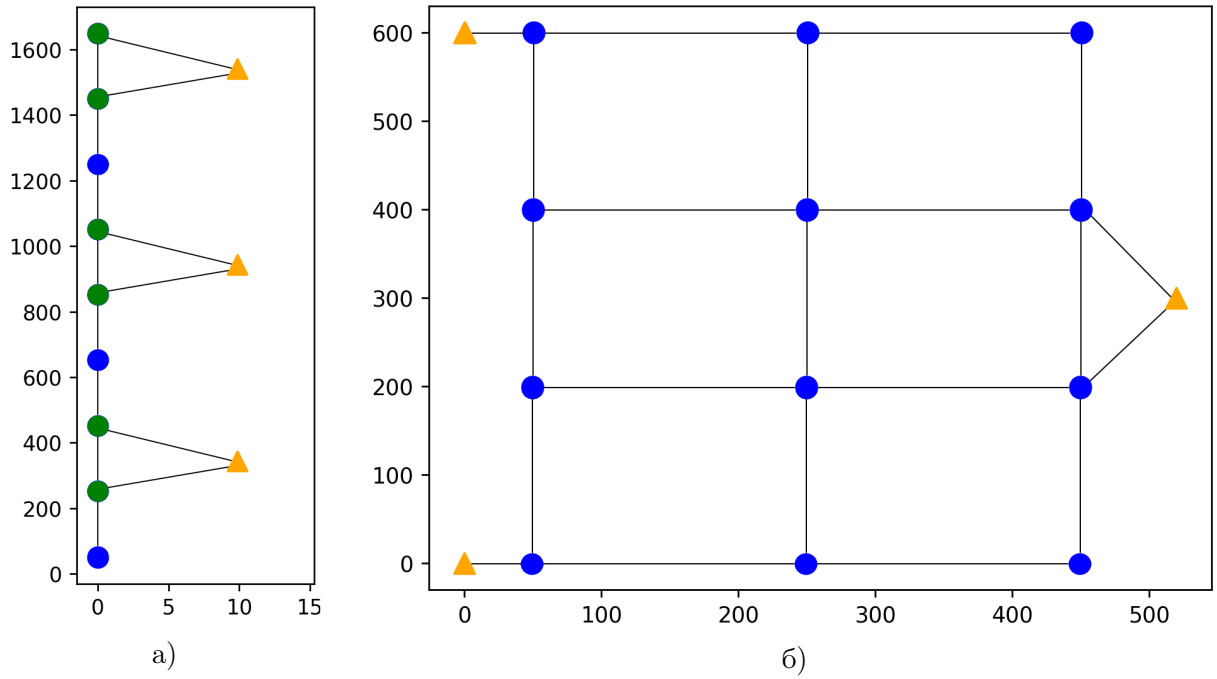

Рис. 2. Сценарии: (а) «Шоссе»; (б) «Манхэттен».

4. АНАЛИЗ ЭФФЕКТИВНОСТИ ПРЕДЛОЖЕННОГО АЛГОРИТМА

4.1. Описание экспериментов

В данной работе рассматривается два сценария: «Шоссе» и «Манхэттен» (одноимённо городу, в котором топология улиц представляет собой регулярную решетчатую структуру). На рис. 2 изображены координаты придорожных модулей (круглые маркеры) и базовых станций (треугольные маркеры) в каждом из сценариев. Рассмотрим каждый сценарий более подробно.

1. *Сценарий «Шоссе»* представляет собой длинный прямой участок дороги со съездами на второстепенные дороги. Вдоль этого участка каждые 200 метров расположены девять придорожных модулей, соединенных друг с другом проводными соединениями. Базовые станции расположены в треугольной сетке с шагом сетки 600 метров и соединены с ближайшими придорожными модулями также проводными соединениями. Серверы, расположенные на придорожных модулях и трёх ближайших к дороге базовых станциях, могут использоваться для выполнения задач V2X-приложений.
2. *Сценарий «Манхэттен»* представляет собой регулярную прямоугольную сетку дорог, похожую на небольшой участок острова «Манхэттен». Двенадцать придорожных модулей расположены в узлах этой решетки и соединены друг с другом вдоль дорог проводными соединениями. Аналогично сценарию «Шоссе», в дополнение к серверам, расположенным на придорожных модулях, для вычислений могут использоваться серверы, расположенные на трёх базовых станциях. Поскольку все придорожные модули расположены вблизи перекрёстков, то на них генерируются задачи всех трёх типов.

Приложения, запущенные на придорожных модулях, генерируют вычислительные задачи различных типов (см. [2]) со следующими значениями их параметров:

– Тип 1: управление транспортными колоннами, $\nu_1 = 40$ Гц, $d_1 = 20$ мс;
– Тип 2: коллективное восприятие обстановки на дороге, $\nu_2 = 10$ Гц, $d_2 = 3$ ms;
– Тип 3: безопасный проезд перекрёстков, $\nu_3 = 50$ Гц, $d_3 = 10$ мс.

Параметры ν и d соответствуют числу генерируемых каждым из приложений вычислительных задач в секунду и ограничению на задержку передачи этих задач из конца в конец. Задачи третьего типа генерируются только теми приложениями, которые запущены на придорожных модулях вблизи перекрёстков (выделены синим цветом на рис. 2).

Исходя из предположения, что задержка в обе стороны на беспроводном соединении между автомобилем и придорожным модулем не превышает $\tau = 4$ мс [13], будем оценивать ограничение на полное время выполнения одной задачи как $D_f = 2d_f - \tau$. Тогда получим: $D_1 = 36$ мс, $D_2 = 2$ мс, $D_3 = 16$ мс. Задержка передачи данных t_e по каждому из соединений e в экспериментах выбирается случайно из интервала $[200, 500]$ мкс.

Число задач σ_{if}, генерируемых за секунду каждым приложением, оценивается следующим образом. Будем предполагать, что каждый придорожный модуль обслуживает участок дороги длиной $l = 200$ м, и при этом придорожные модули расположены вдоль дороги с фиксированным расстоянием l между ними. Каждый автомобиль на этом участке дороги занимает пространство с эффективной длиной $w = 10$ м (с учетом интервалов между соседними автомобилями). Загруженность дороги определяется параметром $\xi \in [0, 1]$. Если дорога на участке, где расположен придорожный модуль i, имеет n полос движения, то интенсивность потока задач типа f, генерируемых на придорожном модуле i, равна:

$$\sigma_{i,f} = I\{f \in \mathbb{F}_i\}\xi n \frac{l}{w}\nu_f,$$

где $I\{f \in \mathbb{F}_i\}$ - индикаторная функция, равная единице, если задачи типа f генерируются на рассматриваемом участке дороги, и нулю в противном случае.

На всех придорожных модулях и некоторых базовых станциях установлены серверы, которые могут использоваться для вычислений. Каждый сервер характеризуется тремя параметрами: его производительностью μ_k и парой (β_k, γ_k), описывающей потребление энергии в ненагруженном состоянии и во время проведения вычислений соответственно. Будем рассматривать три типа серверов (см. Таблицу 2).

Таблица 2. Параметры серверов.

Тип сервера	μ_k, задач/с	β_k, Вт	γ_k, Вт
Сервер придорожного модуля	23000	5	30
Сервер на базовой станции типа 1	92000	3,5	70
Сервер на базовой станции типа 2	161000	5	100

На базовых станциях с вероятностью $0,5$ может быть установлен сервер одного из двух типов (с вероятностью $0,25$ каждый).

4.2. Анализ численных результатов

В данном разделе приводятся численные результаты, полученные в двух сценариях для предложенного алгоритма ILP (англ.: Integer Linear Programming — целочисленное линейное программирование), описанного в разделе 3, и алгоритма RANDOM (англ.: случайный) [8], который распределяет задачи между серверами случайным образом с учетом их ограничений на задержку.

Сценарий «Шоссе»

На рис. 3 представлены результаты, полученные в сценарии «Шоссе». Как видно из рис. 3(б)-(г), оба алгоритма позволяют выполнить ограничение на задержку для всех рассматриваемых

типов вычислительных задач. Однако, согласно рис. 3(а), алгоритм ILP позволяет снизить энергопотребление более чем в два раза по сравнению с алгоритмом RANDOM.

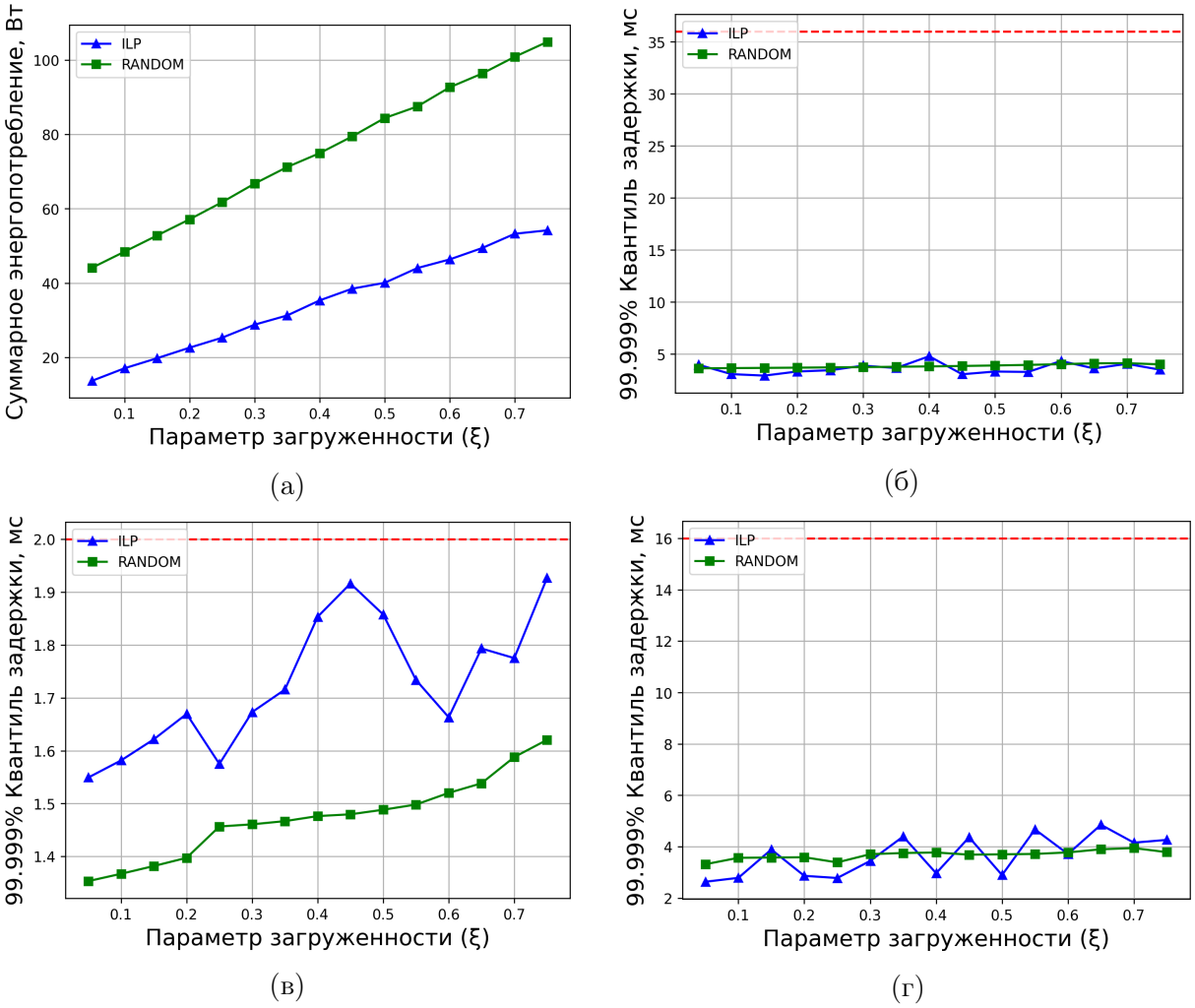

Рис. 3. Сравнение алгоритмов в сценарии «Шоссе»: (а) Суммарное энергопотребление; (б) Квантиль задержки для задач Типа 1; (в) Квантиль задержки для задач Типа 2; (г) Квантиль задержки для задач Типа 3.

Сценарий «Манхэттен»

На рис. 4 представлены результаты, полученные в сценарии «Манхэттен». Все наблюдаемые эффекты аналогичны эффектам, наблюдаемым в сценарии «Шоссе». В частности, алгоритм ILP позволяет снизить расходы оператора до 2,5 раз.

Оценим сокращение расходов оператора за год при использовании предложенного алгоритма на примере крупного мегаполиса, например, Москвы. Предполагая, что среднее число автомобилей N, одновременно находящихся в движении по улицам города, равно $3 \cdot 10^5$, получим, что за год оператор сэкономит $\Delta C \cdot N \cdot 365 \cdot 24 / N_{scenario}$, где ΔC обозначает экономию оператора в рассматриваемом в данном разделе сценарии, а $N_{scenario}$ — число автомобилей в этом сценарии. Согласно полученной оценке алгоритм ILP позволяет сэкономить порядка 10^6 кВт·ч.

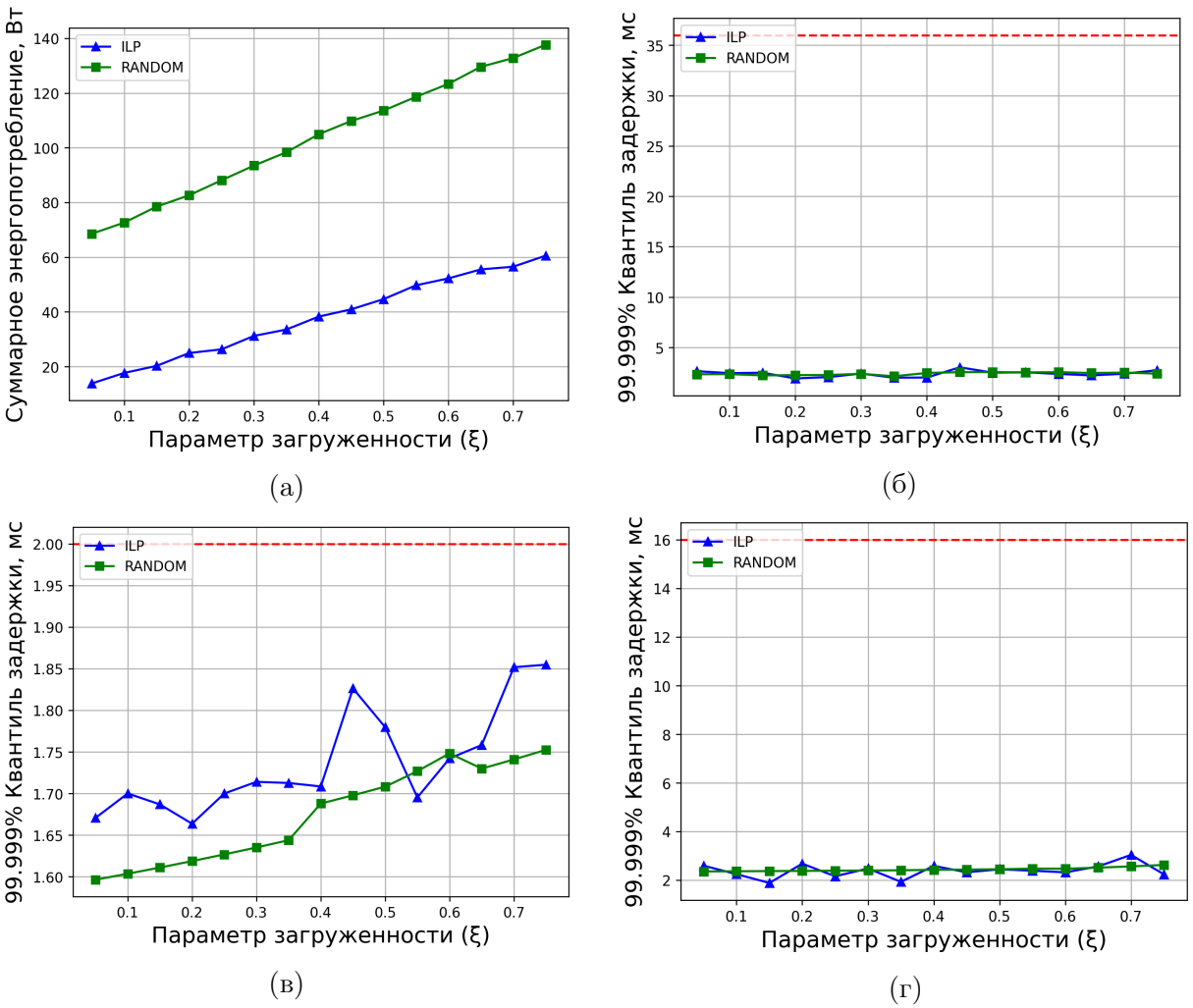

Рис. 4. Сравнение алгоритмов в сценарии «Манхэттен»: (а) Суммарное энергопотребление; (б) Квантиль задержки для задач Типа 1; (в) Квантиль задержки для задач Типа 2; (г) Квантиль задержки для задач Типа 3.

5. ЗАКЛЮЧЕНИЕ

В работе рассмотрена задача распределения вычислительной нагрузки между серверами в интеллектуальных транспортных системах с поддержкой мобильных периферийных вычислений. Представлена математическая формулировка задачи в виде задачи нелинейной целочисленной оптимизации. В качестве функции расходов оператора в работе рассматривается энергопотребление серверов. Предложен метод её линеаризации с помощью введения дополнительных переменных и разработан алгоритм на основе классического метода ветвей и отсечений. Численные результаты показывают, что предложенный алгоритм позволяет существенно (до 2,5 раз) снизить расходы оператора и при этом выполнить строгие ограничения на задержку, накладываемые различными V2X-приложениями.

СПИСОК ЛИТЕРАТУРЫ

1. Research Grand View. Intelligent transportation system market size, share & trends analysis report by type (ATIS, ATMS, ATPS, APTS, EMS), by application (road safety & security, public transport), and segment forecasts, 2019 - 2025. — 2019. — 04. — no. 978-1-68038-036-1. — URL: `https://www.grandviewresearch.com/industry-analysis/intelligent-transportation-systems-industry`.

2. Study on enhancement of 3GPP support for 5G V2X services : Technical Report (TR) : TR 22.886 / 3GPP ; Executor: 3GPP : 2018. — 12. — URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3108. — V16.2.0.

3. Multi-access edge computing (MEC); study on MEC support for V2X use cases : Group Report (GR) : GR MEC 022 / ETSI ; Executor: ETSI : 2018. — 09. — URL: https://www.etsi.org/deliver/etsi_gr/MEC/001_099/022/02.01.01_60/gr_MEC022v020101p.pdf. — V2.1.1.

4. Scalable edge computing deployment for reliable service provisioning in vehicular networks / Federico Tonini, Bahare M Khorsandi, Elisabetta Amato, Carla Raffaelli // Journal of Sensor and Actuator Networks. — 2019. — Vol. 8, no. 4. — P. 51.

5. Mobile-edge computing for vehicular networks: a promising network paradigm with predictive offloading / Ke Zhang, Yuming Mao, Supeng Leng et al. // IEEE Vehicular Technology Magazine. — 2017. — Vol. 12, no. 2. — P. 36–44.

6. Minimum-cost offloading for collaborative task execution of MEC-assisted platooning / Xiayan Fan, Taiping Cui, Chunyan Cao et al. // Sensors. — 2019. — Vol. 19, no. 4. — P. 847.

7. Efficient mobility-aware task offloading for vehicular edge computing networks / Chao Yang, Yi Liu, Xin Chen et al. // IEEE Access. — 2019. — Vol. 7. — P. 26652–26664.

8. Guo Hongzhi, Zhang Jie, Liu Jiajia. FiWi-enhanced vehicular edge computing networks: collaborative task offloading // IEEE Vehicular Technology Magazine. — 2018. — Vol. 14, no. 1. — P. 45–53.

9. Kleinrock Leonard. Queueing systems. Volume 1: Theory. — Wiley New York, 1975.

10. Barroso Luiz André, Hölzle Urs. The case for energy-proportional computing // Computer. — 2007. — Vol. 40, no. 12. — P. 33–37.

11. Mitchell John E. Branch-and-cut algorithms for combinatorial optimization problems // Handbook of applied optimization. — 2002. — Vol. 1. — P. 65–77.

12. Mitchell Stuart, OSullivan Michael, Dunning Iain. PuLP: a linear programming toolkit for python // The University of Auckland, Auckland, New Zealand. — 2011.

13. Latency of cellular-based V2X: Perspectives on TTI-proportional latency and TTI-independent latency / Kwonjong Lee, Joonki Kim, Yosub Park et al. // IEEE Access. — 2017. — Vol. 5. — P. 15800–15809.

Cost Optimization for Computing Resources Management in Intelligent Transportation Systems

Belogaev A.A., Elokhin A.A., Krasilov A.N., Khorov E.M.

This paper considers a computing resources management problem in the context of intelligent transportation systems. We formulate an operator's expenses minimization problem subject to delay constraints that include transmission and computational delays. To solve the problem, we propose an algorithm based on a method to reduce the problem to an integer linear programming problem. The simulation results show that the proposed algorithm significantly lowers operator's expenses compared with the existing approaches while satisfying the strict delay requirements of various applications.

KEYWORDS: V2X, VANET, intelligent transportation systems, integer linear programming, multi-access edge computing, computing resources management