Distributed Task Splitting and Offloading in Mobile Edge Computing

Yanling Ren[1], Zhihui Weng[1], Yuanjiang Li[1], Zhibin Xie[1(✉)], Kening Song[2], and Xiaolei Sun[3]

[1] Jiangsu University of Science and Technology, Zhenjiang, China
xiezhibin@just.edu.cn
[2] PLA AF 95829, Xiaogan, China
skning@163.com
[3] PLA Navy Submarine Academy, Qingdao, China
msyl1211@163.com

Abstract. With the rapid development of the mobile internet, many emerging compute-intensive and data-intensive tasks are extremely sensitive to latency and cannot be implemented on mobile devices (MDs). To solve this problem, mobile edge computing (MEC) appears to be a promising solution. In this paper, we propose a distributed task splitting and offloading algorithm (DSOA) for the scenario of multi-device and multi-MEC servers in ultra-dense networks (UDN). In the proposed scheme, the MDs can perform their tasks locally or offload suitable percentage of tasks to the MEC server. The optimization goal is to minimize the overall task computation time. Since the MDs are selfish, we propose a game theory approach to achieve optimal global computation time. Finally, the numerical simulation results verify that the algorithm can effectively reduce global computation time.

Keywords: Mobile edge computing · Offload strategy · Game theory

1 Introduction

In recent years, the mobile internet have developed rapidly, which has promoted the popularity of mobile devices (MDs) and the exponential growth of internet traffic [1]. Ultra-dense network (UDN) is considered to be one of the key technologies of the future mobile network [2]. However, many applications are computationally intensive and data intensive with strict time requirements such as virtual reality, face recognition, smart traffic and interactive games. The computing resources of existing MDs are often limited, and the processing capability is difficult to meet the requirements of these applications. Therefore, the deployment of UDN faces unprecedented challenges [3].

Cloud computing is an optional solution [4, 5]. However, the remotely offloading task to the cloud has some limitations that will still consume some unexpected excessive time and energy. Subsequently, some researchers began to pay attention to provide cloud computing ability at the edge of the wireless access network near MDs, which is

H. Gao et al. (Eds.): ChinaCom 2019, LNICST 312, pp. 33–42, 2020.
https://doi.org/10.1007/978-3-030-41114-5_3

named Mobile Edge Computing (MEC). MEC not only solves the problem of insufficient resources of the MDs, but also makes up for the shortcomings of long delays in cloud computing. The issues related to deployment, resource allocation, load balancing and fairness among multiple MDs in a mobile network are discussed in [6]. Some works have been done on the problem of offloading decision-making and resource allocation. The key technology is how to achieve low latency caused by communication and computation. In [7], cloud and wireless resource allocation are considered, and a convex optimization solution method is adopted. The model established in [7, 8] is a single model of multi-device and single MEC server. Guo et al. discussed the case of multiple servers and used potential games to solve the problem of offloading strategies, but the situation of task splitting was not taken into account [9]. The distributed joint computing offloading and resource allocation optimization problem in heterogeneous networks is considered in [10], but it is also directed to the case of single server offloading. Reference [11] combines the offloading task and the scheduling execution sequence to form a dynamic task offloading and scheduling problem, and use a logic-based bending decomposition technique to solve the problem.

Although, there are some excellent works in terms of offloading decisions and resource allocation, few literature has considered multi-device and multi-server offload scenarios to our knowledge. Most of the works are binary offloading choice, either the MDs performs computational tasks locally on its CPU or offloads its computationally intensive tasks to the MEC server [7]. In this paper, we propose an effective distributed task splitting and offloading algorithm (DSOA) for multi-device and multi-MEC server scenarios in UDN. In the proposed scheme, the change of the MEC server computing resources and the distance are considered. Based on these factors, the optimal server and the offload ratio are searched for realizing the goal of overall minimum computation time. Since the MDs are selfish, a game approach is proposed to find the optimal strategy. The numerical simulation results show the effectiveness of DSOA.

2 System Model

2.1 Network Model

As shown in Fig. 1, we consider a distributed UDN including multi-device and multi-MEC servers. In this network, there are $j \in \mathcal{M}=\{1, 2,\ldots,M\}$ small base stations (BS), $i \in \mathcal{N}=\{1, 2,\ldots, N\}$ MDs, the small BS and MDs are evenly distributed. The small BSs is connected to the core network through wired fibers. A lightweight MEC server is deployed around the small BS, so that the small BS has MEC capability. Assume that each MD has an intensive computing task to perform, they will choose to offload to the MEC server subject to their own resources and latency requirements. The small BSs can communicate with each other, and determine the MD's offloading decisions that minimize the overall computation time.

2.2 Communication Model

The data transmission rate r_i of MD i can be expressed as:

$$r_i = B \log_2(1 + \frac{p_i h_i}{\sigma^2}), \tag{1}$$

where B represents the bandwidth of the selected channel when the task is offloaded, p_i represents the transmission power of MD i, h_i is the channel gain between MD i and the BS, and σ^2 represents the thermal noise power.

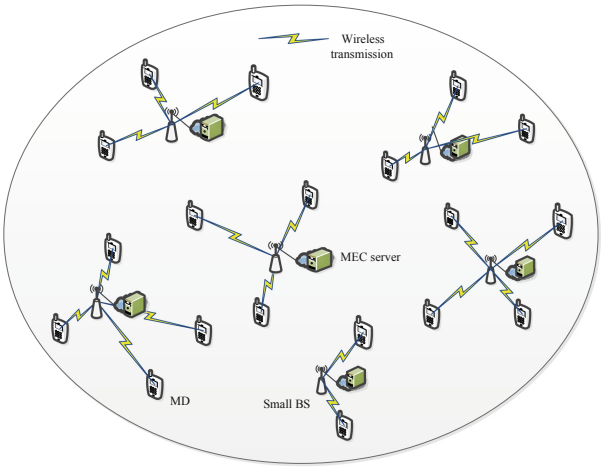

Fig. 1. Mobile edge computing model in ultra-dense network

2.3 Computation Model

If MD i selects MEC server j to offload, then the task transmission time is divided into three parts: uplink time, channel transmission time and downlink time. For each MD i, the size of task input data is b_i. Compared with b_i, the size of output data that calculation result from MEC servers is less, so the downlink time is set to be a constant α [7]. So the task transmission time is

$$t_i^{\text{trans}} = \frac{b_i}{r_i} + \frac{d_{i,j}}{v} + \alpha, \tag{2}$$

where $d_{i,j}$ is the distance from the MD i to the MEC server j, and v is the speed at which the electromagnetic wave propagates.

Assume that s_i is the required CPU cycles for finishing the task b_i, and f_i is the computing capability of the device, then the computation execution time can be expressed as

$$t = \frac{s_i}{f_i}. \tag{3}$$

3 Problem Formulation

Assume that each MD i is running a computing task, there are two possibilities for the execution location of these tasks. One is performed locally by the MD, and the other

is offloaded to the MEC server for execution. Limited by their computation resources, the MDs will choose to offload computing tasks. However, if a lot of MDs offload its task, it will cause a certain amount of congestion and increase the task calculation time. The MD can make a decision whether to split the computation task according to its own needs and the characteristics of the task.

If the computing task is a detachable task, a part of the task is executed locally, and the rest is offloaded to the MEC server for calculation. That is, each MD can only select one MEC server to offload the computing task, and the MEC server j can provide computing service for multiple MDs at the same time. Let $\lambda_{i,j} \in \{0,1\}$ denote the offloading decision between the MD i and the MEC server j, where $\lambda_{i,j} = 1$ means that the MD i decides to offload the computing task to the MEC server j, otherwise $\lambda_{i,j} = 0$.

Suppose that the MD i task offloading proportion is x_i, and the locally calculating proportion is (1-x_i). According to (3), we know that the local calculation time is

$$T_i^{\text{local}} = (1 - x_i)\frac{s_i}{f_i^{\text{local}}}, \tag{4}$$

where f_i^{local} represents the local computing capability of the MD i.

The computation time t_i^{cloud} of the MEC server is mainly divided into two parts, one is the transmission time of the computation task, and the other is the calculation execution time of the task processed by the MEC server. Combined with (2) (3), the task execution time on the MEC server side can be expressed as

$$T_i^{\text{cloud}} = \frac{x_i b_i}{r_i} + \frac{d_{i,j}}{v} + \alpha + \frac{x_i s_i}{f_{i,j}}, \tag{5}$$

where $f_{i,j}$ represents the computing capability allocated by the MEC server j to the MD i, and $f_{i,j} = f_j \Big/ \sum_{i=1}^{N} \lambda_{i,j}$, where $\sum_{i=1}^{N} \lambda_{i,j}$ indicates the number of MDs that all choose to offload the computing task to the MEC server j.

Therefore, the task computation time of the MD i is

$$T_i = \max\{T_i^{\text{local}}, T_i^{\text{cloud}}\}. \tag{6}$$

Based on this, the problem of minimizing overall computation time can be described as follows:

$$\begin{aligned} \min_{\{\lambda_{i,j}, x_i\}} \quad & \sum_{i=1}^{N} T_i, \\ s.t. \quad & \sum_{i=1}^{N} f_{i,j} \le f_j^{\max}, \\ & T_i \le T_i^{\max}, \\ & 0 < p_i \le p_i^{\max}, \\ & f_i^{\text{local}} \ge 0. \end{aligned} \tag{7}$$

In (7), the first constraint indicates that the sum of the computing resources allocated by the MEC server j for the MDs does not exceed its maximum computing capability;

the second constraint indicates that the total execution time of the computing task is lower than the sum of the maximum tolerable time $T_i^{\max}$ of each MD. In this paper, we set $T_i^{\max}$ as the time which all tasks are executed locally, because the selection task offloading execution time cannot be performed higher than the task locally; the third constraint represents the range of variation of the uplink transmission power of the MD i; the fourth constraint indicates that the MD i has a certain computing capability f_i^{local}.

4 Distributed Task Splitting and Offloading Algorithm

After one MEC server is selected, the proportion x_i of offloading task determines T_i^{local} and T_i^{cloud}. According to (6), we can know that the computation time of the MD i depends on the maximum value of the local computing time T_i^{local} and the MEC server side time T_i^{cloud}. The computation time of the MD i has three possible situations, i.e., $T_i^{\text{local}} < T_i^{\text{cloud}}$, $T_i^{\text{local}} > T_i^{\text{cloud}}$, $T_i^{\text{local}} = T_i^{\text{cloud}}$. Based on (4) and (5), we can find that the computation time of the MD i when $T_i^{\text{local}} > T_i^{\text{cloud}}$ or $T_i^{\text{local}} < T_i^{\text{cloud}}$ is more than when $T_i^{\text{local}} = T_i^{\text{cloud}}$. Therefore, when $T_i^{\text{local}} = T_i^{\text{cloud}}$, the optimal T_i^* exists and the task computation time reaches a minimum value, then

$$x_i^* = \frac{\frac{s_i}{f_i^{\text{local}}} - \frac{d_{i,j}}{v} - \alpha}{\frac{b_i}{r_i} + \frac{s_i}{f_i^{\text{local}}} + \frac{s_i}{f_{i,j}}}. \tag{8}$$

Based on (8), we can know that the time caused by computing locally of MD i is as fast as the MEC time for the transmission and calculate caused by offloading to the MEC server. Therefore, the problem (7) can be converted into

$$\begin{aligned} &\min_{\{\lambda_{i,j}\}} \sum_{i=1}^{N} (1 - x_i^*) \frac{s_i}{f_i^{\text{local}}}, \\ &s.t. \ \sum_{i}^{n} f_{i,j} \le f_j^{\max}, \\ &T_i \le T_i^{\max}, \\ &f_i^{\text{local}} > 0. \end{aligned} \tag{9}$$

In order to solve the optimization problem of (9), we need to obtain the specific offloading ratio x_i according to the distance $d_{i,j}$ and the computing capability $f_{i,j}$ allocated by the MEC server j to the MD i. Therefore, the problem (9) can be transformed into an offloading decision problem. In order to solve the offloading decision problem, we adopt the game approach to find the optimal decision and achieve minimum global computation time.

Let a_i indicates the offloading decision of MD i, and let a_{-i} indicates the offloading decision of all other MDs except MD i. The goal of the game is to minimize the computational time of each MD, i.e.,

$$\min_{\{\lambda_{i,j}\}} T(a_i, a_{-i}), \forall a_i, \tag{10}$$

where $T(a_i, a_{-i})$ is the overall computing time for all MDs based on the current decision, i.e.,

$$T(a_i, a_{-i}) = \sum_{i}^{N} T_i(\lambda_{i,j}). \tag{11}$$

The offloading decision problem can be described as a distributed offloading decision game for multi-MD and multi-MEC servers. The game can be denoted by

$$\Gamma = (K, \{\mathbb{A}_{i,j}\}_{i\in K}, \{T(a_i, a_{-i})\}_{i\in K}), \tag{12}$$

where K represents all game participants, $\{\mathbb{A}_{i,j}\}_{i\in K}$ is the decision set of participant i, $\{T(a_i, a_{-i})\}_{i\in K}$ is the revenue function of MD i in the game, and this paper refers to the time function.

According to the Nash equalization (NE) existence theorem, there is at least one pure strategic NE in a finite number of repeated games [12]. Our distributed offloading decision game have finite players and offloading strategy space, and each player can choose a pure strategy from a limited set of offloading strategies, so have a NE. Thus, our game will get NE over a limited number of iterations. In this state of NE, none MD can further reduce computing time by changing its strategy. If the strategy of the equilibrium point is denoted as

$$a^* = (a_1^*, \ldots, a_{i-1}^*, a_i^*, a_{i+1}^*, \ldots, a_K^*), \tag{13}$$

where $a*$ is the optimal decision. According to the equilibrium point decision, the optimal computation time T_i can be expressed as

$$T(a_i^*, a_{-i}) \leq T(a_i, a_{-i}), \forall a_i. \tag{14}$$

Next, in order to get the equilibrium point decision a^*, we propose an effective DSOA. In this algorithm, it is assumed that the MDs can get all the information, including the distance, the computing capability of the MEC server, and the size of tasks that other MDs need to offload. First, each MD first selects the MEC server closest to itself as the offload destination, and the current decision is called the initial decision. Within each iteration, the MD can select a better decision based on the decisions of other MDs in the previous iteration, and calculate whether the decision is an overall better decision. If $T(a_i', a_{-i}) < T(a_i, a_{-i})$, store the updated $\mathbb{D}$, $\mathbb{T}$ and broadcast them to other MDs. Then, each iteration updates the decision that is the global optimum in the current $\mathbb{D}$. After a finite number of iterations, when the $\mathbb{D}$ is empty, i.e., all MDs have no better choice than current decisions, it indicates that the entire system has reached the NE, and the algorithm has converged to the global optimal solution.

Algorithm distributed task splitting and offloading algorithm (DSOA)

Initialize: $N, M, B, d, p_i, \sigma^2, b_i, s_i, f_j, f_i^{\text{local}}, \alpha$; an offloading decision space of the MDs $a_i = \{1, \cdots, M\}$; the initial offloading decision of the MD i is the MEC server j closest to it, i.e., (a_i, a_{-i}); the initial computation time $T(a_i, a_{-i})$ based on the initial decision; the update set $\mathbb{D} = \varnothing$ of the MD i; the time update set $\mathbb{T} = \varnothing$;
for each iteration τ
  for all MD i do
    compute x_i, T_i, and compute whether $T_i(\lambda_{i,k}) < T_i(\lambda_{i,j})$ exists;
    if $T(a_i', a_{-i}) < T(a_i, a_{-i})$ then
    save $\lambda_{i,k}$ to $\mathbb{D}$ and save $T(a_i', a_{-i})$ to $\mathbb{T}$
    end if
  end for
  while $\mathbb{T} = \varnothing$ do
    find the minimum time $T(a_i^*, a_{-i})$ from the set of $\mathbb{T}$;
    update MD's decisions, let $a_i = a_i^*$;
  end while
end for

5 Simulation Results

In this section, the transmission distance between MDs to MEC server is in the [100, 1000] m following uniform distribution. The bandwidth of the wireless channel is $B = 1$ MHz. The transmit power of the MD i is 100 mW. The Gaussian thermal noise σ^2 is -100 dBm. The path fading factor γ is 3. The downlink backhaul time α of the calculated result is 0.5 μs. Each MD's computing task data size b_i is between 200 KB and 500 KB which follows uniform distribution, and the required CPU cycles range from 5,000 megacycles to 10,000 megacycles. The computing capability of the MEC server is 4 GHz, and the computing capability of the MD is 0.1 GHz.

In order to verify the DSOA, we use the exhaustive scheme as a benchmark for comparison, as shown in Fig. 2. Considering that the exhaustive algorithm has a large amount of computation and is not suitable for adopting more MDs and MEC servers, therefore the number of MEC servers is set to 4, and the number of MDs $N = 1, 2, \ldots, 8$. It can be found that the overall minimum computation time of our proposed DSOA scheme is very consistent with exhaustive scheme results. The numerical results confirm that the solution can provide a near-optimal solution to our problem.

In addition, we present the convergence behavior of the algorithm under different MD numbers, which are $N = 30, 50$ respectively. The number of MEC servers is 8. It can be easily observed from Fig. 3 that the overall computation time of the system gradually decreases as the number of iterations increases. After a finite number of iterations, the DSOA converges to a certain value. In addition, it can be found from the two curves that when $N = 30$, the number of iterations to reach equilibrium is 7, and when $N = 50$, the number of iterations is 13, indicating that the more devices, the more iterations are needed.

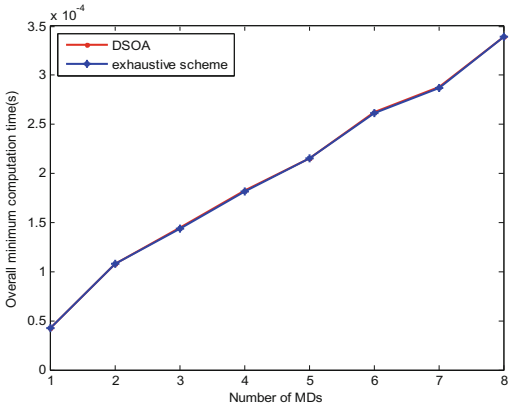

Fig. 2. Comparison between DSOA and exhaustive scheme

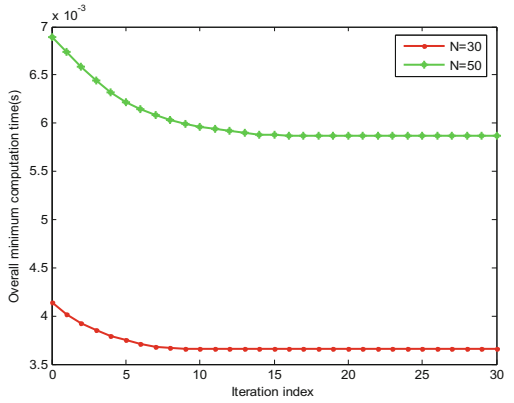

Fig. 3. Iteration comparison of DSOA with different number of MDs

In order to verify the performance of our proposed DSOA, we simulated the local computing strategy, all offloading computing strategies and our DSOA respectively, and the other two strategies also adopt game theory to get optimal decision. At this time, the number of MEC servers is set to 10. The comparison results are shown in Fig. 4. It can be seen from the simulation results that our solution can increase the calculation time by 43% and 63% respectively compared to all computing locally and all offloading to the MEC server. In the scheme of computing locally, all MDs perform calculations on the device itself, and the CPU consumption per unit is large, so the time of local execution of the computing task is significantly increased. The scheme of offloading all tasks to the MEC server, as the number of MDs increases, the latency will increase significantly. This is because the more MDs there are, the more tasks that need to be offloaded, the communication resources and MEC server resources are limited, so it will cause more time.

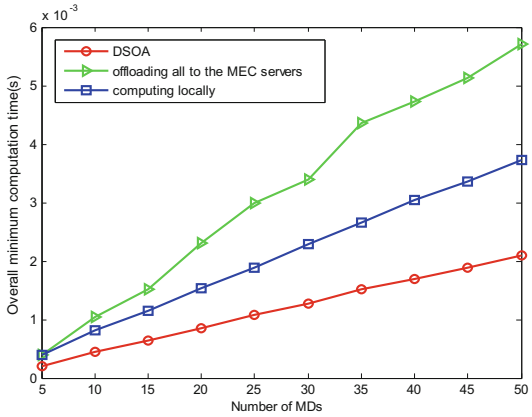

Fig. 4. Comparison results by adopting different offload strategies

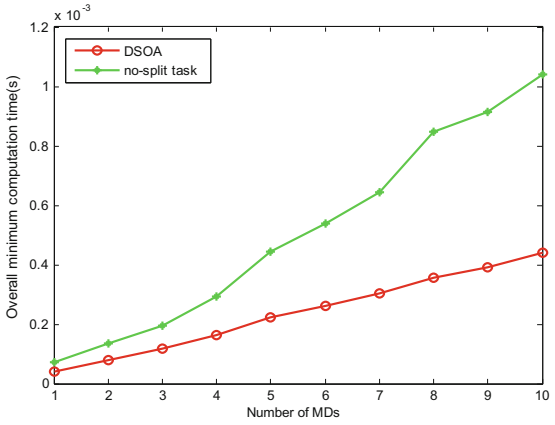

Fig. 5. Comparison results between task splitting and non-splitting

Finally, we compare DSOA with the scheme of no splitting the task. Under this scheme, the MD can choose to calculate locally or offload it to the MEC server and it also adopt game theory to get optimal decision. As can be seen from Fig. 5, as the number of MDs increases, DSOA will reduce the computation time by nearly 57% compared to the scenario without task splitting. Therefore, our DSOA has better flexibility and adaptability. Numerical results not only demonstrates the need to use multiple MEC servers for computational offloading in UDN, but also demonstrates that our DSOA can adopt multiple server selections and partially offload to reduce overall computation time.

6 Conclusions

Due to most of MDs have limited resources and mobile applications are sensitive to task computation time, we consider a minimizing the computational offloading time scheme

for multi-device and multi-MEC servers networks. Different from previous works, the tasks can be splitted to make full use of the computing resources of MDs and MEC servers. Next, we transform the minimize computation offloading time problem to an offloading decision problem. In order to get the optimal offloading decision, we propose an effective DSOA based on game theory. Experiments show that the DSOA can effectively achieve the minimum overall computation time and improve network performance. In the future work, we will pay attention to more practical MEC scenarios and solve the weighting problem of joint time and energy consumption.

References

1. Sun, W., Liu, J., Zhang, H.: When smart wearables meet intelligent vehicles: challenges and future directions. IEEE Wirel. Commun. **24**(3), 58–65 (2017)
2. Zhang, S., Zhang, N., Zhou, S., Gong, J., Niu, Z., Shen, X.: Energy sustainable traffic steering for 5G mobile networks. IEEE Commun. Mag. **55**(11), 54–60 (2017)
3. Yang, T., Zhang, H., Ji, H., Li, X.: Computation collaboration in ultra dense network integrated with mobile edge computing. In: PIMRC, Montreal, Canada, pp. 1–5. IEEE (2017)
4. Zhang, W., Wen, Y., Wu, D.: Collaborative task execution in mobile cloud computing under a stochastic wireless channel. IEEE Trans. Wirel. Commun. **14**(3), 81–93 (2015)
5. Wang, J., Peng, J., Wei, Y., Liu, D., Lu, J.: Adaptive application offloading decision and transmission scheduling for mobile cloud computing. IEEE China Commun. **14**(3), 169–181 (2017)
6. Kiani, A., Ansari, N.: Toward hierarchical mobile edge computing: an auction-based profit maximization approach. IEEE Internet Things J. **4**(6), 2082–2091 (2017)
7. Zhang, J., et al.: Joint offloading and resource allocation optimization for mobile edge computing. In: GLOBECOM IEEE Global Communications Conference, Singapore, pp. 1–5. IEEE (2017)
8. Ren, J., Yu, G., Cai, Y., He, Y.: Latency optimization for resource allocation in mobile-edge computation offloading. IEEE Trans. Wirel. Commun. **17**(8), 5506–5519 (2018)
9. Guo, J., Zhang, H., Yang, L., Ji, H., Li, X.: Decentralized computation offloading in mobile edge computing empowered small-cell networks. In: Computer Communications. IEEE GlobeCom, Singapore, pp. 1–6. IEEE (2017)
10. Zhang, J., Xia, W., Yan, F., Shen, L.: Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing. IEEE Access **6**, 19324–19337 (2018)
11. Alameddine, H., Sharafeddine, S., Sebbah, S., Ayoubi, S., Assi, C.: Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing. IEEE J. Sel. Areas Commun. **37**(3), 668–682 (2019)
12. Nash, J.F.: Equilibrium points in N-person games. Proc. Nat. Acad. Sci. U.S.A. **36**(1), 48–49 (1950)