IEICE TRANS. INF. & SYST., VOL.E100–D, NO.12 DECEMBER 2017

LETTER

Cost Aware Offloading Selection and Resource Allocation for Cloud Based Multi-Robot Systems

Yuan SUN[†a)], ***Student Member***, **Xing-she ZHOU**[†], ***and*** **Gang YANG**[†], ***Nonmembers***

SUMMARY In this letter, we investigate the computation offloading problem in cloud based multi-robot systems, in which user weights, communication interference and cloud resource limitation are jointly considered. To minimize the system cost, two offloading selection and resource allocation algorithms are proposed. Numerical results show that the proposed algorithms both can greatly reduce the overall system cost, and the greedy selection based algorithm even achieves near-optimal performance.
key words: *computation offloading, cloud computing, multi-robot systems, user weights*

1. Introduction

The integration of cloud technology allows for the design of high-performance and low-cost multi-robot systems [1], [2]. With the aid of cloud offloading, the computation capability of robots can be greatly enhanced so that they can be deployed in extreme environments, such as disaster relief, grasping unknown objects. The energy consumption of robots also can be reduced and hence they can work longer than before.

However, offloading also incurs additional cost, and obtaining cost-efficient offloading schemes is not very easy. First, if multiple users offload their tasks simultaneously, the communication interference among them will cause a reduction of wireless communication quality, and then delay the task execution time [3]. Second, unlike public clouds, private clouds or edge clouds do not have unlimited computation capability [4]. To minimize the system cost while satisfying task deadlines, transmission powers and cloud resources need to be reasonably allocated.

In existing works on computation offloading, communication interference and cloud resource limitation have been considered with offloading selection. The differences among them are that some works only jointly optimize offloading selection and communication interference management [3], [5], and some works only aim to make optimal resource allocation and offloading selection decisions [6]–[8], while the others [9] jointly consider the three issues.

One limitation among most existing works is that they assume that system managers always have the same preferences on different users. In fact, this assumption is not always reasonable. Assume that a robot (denoted by A) with high battery level and a robot (denoted by B) with low battery level both have an offloading request. From the perspective of system managers, accepting the offloading request from B can obtain more benefits, because it can extend the working time of B, and hence increase the overall system efficiency.

In this letter, the preferences of system managers, communication interference and cloud resource limitation are simultaneously considered in the offloading problem for cloud based multi-robot systems. This letter has several contributions. First, we model the preferences of system managers on robots as user weights, and formulate the offloading problem into a system cost minimization problem. Second, because the problem is NP-Hard, two low-complexity offloading selection and resource allocation algorithms are presented to efficiently solve the problem. Third, through numerical simulations, we show that the proposed algorithms can give quite good or even near-optimal results.

2. System Model and Problem Formulation

System Model: We assume that there are N mobile robots working in a disaster site, denoted by a set of $\mathcal{N} = \{1, 2, \ldots, N\}$, and each robot has a computation-intensive task to offload. A central station with a private cloud and a wireless access point (e.g., WiFi access point) is set up in the disaster site to make computation offloading possible. The robots communicate with the central station through the wireless channel. The central station has a *cloud controller*, which collects offloading requests and makes offloading selection and resource allocation decisions.

Communication Model: Similar to previous studies [3], [5], the wireless channel from the robots to the access point follows quasi-static block fading. Let $\mathcal{L}$ denote the set of offloaded tasks. Given $\mathcal{L}$, the uplink data rate for robot n to offload its task over the wireless channel can be computed as $R_n(\mathcal{L}) = W \log_2(1+g_n q_n/(\sigma_n^2+\sum_{i\in\mathcal{L},i\neq n} g_i q_i))$ where q_n is the transmission power of robot n, and g_n denotes the channel gain from robot n to the access point, and σ_n^2 is the thermal noise power with the link between robot n and the access point, and W denotes the channel bandwidth.

Computation Model: We let D_n denote the input data size of task n. L_n denotes the total number of CPU cycles required to accomplish task n. T_n^d denotes the deadline of task n.

Manuscript received June 9, 2017.
Manuscript revised July 26, 2017.
Manuscript publicized August 28, 2017.
[†]The authors are with the School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an, 710072, P.R. China.
a) E-mail: sunyuan@mail.nwpu.edu.cn
DOI: 10.1587/transinf.2017EDL8128

1) Local computing: We let f_n denote the computation capability (i.e., CPU frequency) that robot n allocates to task n. Then, the execution time of task n by local computing can be given by $T_n^l = L_n f_n^{-1}$. According to [4], [5], the energy consumption of robot n is given by $E_n^l = \kappa L_n f_n^2$, where κ is the effective switched capacitance depending on the chip architecture.

2) Cloud computing: The execution of offloaded task n includes three phases in sequence: (i) transmitting the input data of task n to the cloud; (ii) executing the task on the cloud; (iii) returning the output data to robot n. Similar to some of existing studies on computation offloading [3], [5], we choose to ignore the output data receiving phase, because the output data size is often considerably smaller than the input data size. Taking an object pose estimation task [10] as an example, the input data includes a set of images and the total size of it could be hundreds of kilobytes at least, while the output data is just the object location and pose, and takes a few dozens of bytes at most.

Let f_c denote the clock frequency of the processing unit on the private cloud; and let h_n denotes the fraction of the overall processing power assigned to task n. Then, the execution time of task n by cloud computing can be calculated as $T_n^c(\mathcal{L}) = D_n/R_n(\mathcal{L}) + L_n h_n^{-1} f_c^{-1}$, and the energy consumption of robot n can be calculated as $E_n^c(\mathcal{L}) = q_n D_n/R_n(\mathcal{L})$.

Problem Definition: The preference of the system manager on robot n is modeled as user weight $\rho_n \in (0, 1)$. We formulate the offloading problem into the weighted overall system cost minimization problem as follows

$$\mathbf{P1}: \min_{\mathcal{L},\mathcal{H},\mathcal{Q}} \sum\nolimits_{n\in\mathcal{L}} (1-\rho_n)E_n^c(\mathcal{L}) + \sum\nolimits_{n\in(\mathcal{N}\setminus\mathcal{L})} (1-\rho_n)E_n^l \tag{1a}$$

$$\text{s.t. } T_n^c(\mathcal{L}) <= T_n^d, \forall n \in \mathcal{L}, \tag{1b}$$

$$q_n <= Q_n, \forall n \in \mathcal{L}, \tag{1c}$$

$$\sum\nolimits_{n\in\mathcal{L}} h_n <= 1, \tag{1d}$$

$$\mathcal{L} \subseteq \mathcal{N}, \tag{1e}$$

where $\mathcal{H} = \{h_n | n \in \mathcal{N}\}$, $\mathcal{Q} = \{q_n | n \in \mathcal{N}\}$. According to (1a), we find that for minimizing the overall system cost, the offloading requests from the robots with larger user weights is obviously more preferred. Constraint (1b) ensures that all the tasks are completed before their deadlines, and constraint (1c) gives the upper bounds for the transmission powers, denoted by $\{Q_n | n \in \mathcal{N}\}$, and constraint (1d) guarantees that the total processing power assigned does not exceed the maximum computation capability of the cloud.

3. The Proposed Algorithms

The decision variable $\mathcal{L}$ makes **P1** nonconvex and NP-Hard. To efficiently obtain a sub-optimal solution, we propose to solve offloading selection and resource allocation separately.

Resource Allocation: Given the offloaded task set $\mathcal{L}$, **P1** is reduced to the resource allocation problem as follows

$$\mathbf{P2}(\mathcal{L}): \min_{\mathcal{H},\mathcal{Q}} \sum\nolimits_{n\in\mathcal{L}} (1-\rho_n)E_n^c(\mathcal{L}) + \sum\nolimits_{n\in(\mathcal{N}\setminus\mathcal{L})} (1-\rho_n)E_n^l \tag{2a}$$

$$\text{s.t. } (1b), (1c), (1d),$$

where $\mathcal{H} = \{h_n | n \in \mathcal{L}\}$, $\mathcal{Q} = \{q_n | n \in \mathcal{L}\}$. $\mathbf{P2}(\mathcal{L})$ is non-convex because it has a complicated and non-convex objective function. To efficiently find its optimal solution, we will transform it to an equivalent convex problem.

We first introduce three auxiliary variables x_n, y_n and z_n, and replace q_n with e^{z_n}. Then, we import a constraint $e^{z_n}/R_n(\mathcal{L}) <= e^{-x_n}$ and use $D_n e^{-x_n}$ to replace $E_n^c(\mathcal{L})$ to handle the nonconvexity of $E_n^c(\mathcal{L})$. Finally, we add another constraint $R_n(\mathcal{L}) >= e^{y_n}$ to handle the nonconvexity of $T_n^c(\mathcal{L})$. These lead to

$$\mathbf{P3}(\mathcal{L}): \min_{\mathcal{H},\mathcal{X},\mathcal{Y},\mathcal{Z}} \sum_{n\in\mathcal{L}} (1-\rho_n)D_n e^{-x_n} + \sum_{n\in(\mathcal{N}\setminus\mathcal{L})} (1-\rho_n)E_n^l \tag{3a}$$

$$\text{s.t. } D_n/e^{y_n} + L_n h_n^{-1} f_c^{-1} <= T_n^d, \forall n \in \mathcal{L}, \tag{3b}$$

$$R_n(\mathcal{L}) >= e^{y_n}, \forall n \in \mathcal{L}, \tag{3c}$$

$$e^{z_n} <= Q_n, \forall n \in \mathcal{L}, \tag{3d}$$

$$\sum\nolimits_{n\in\mathcal{L}} h_n <= 1, \tag{3e}$$

$$e^{z_n}/R_n(\mathcal{L}) <= e^{-x_n}, \forall n \in \mathcal{L}, \tag{3f}$$

where $\mathcal{X} = \{x_n | n \in \mathcal{L}\}$, $\mathcal{Y} = \{y_n | n \in \mathcal{L}\}$, $\mathcal{Z} = \{z_n | n \in \mathcal{L}\}$.

The objective function (3a) and constraints (3b, 3d, 3e) are convex, and the convexity of constraints (3c, 3f) is also proved in [11]. Therefore, $\mathbf{P3}(\mathcal{L})$ is a convex problem, and can be globally solved by interior-point methods [12].

Algorithm 1: Greedy Selection based System Cost Minimization Algorithm

```
Initialize L^[0] = ∅, A^[0] = N, and the iteration number i = 0.
While A^[i] is not empty do
  Foreach m ∈ A^[i] do
    Solve P3(L^[i] ⋃ {m}) to obtain the system cost. If it is
    infeasible, set the system cost as +∞.
  End Foreach
  If ∃m ∈ A^[i] makes P3(L^[i] ⋃ {m}) feasible then
    Select the task m̃ that can minimize the system cost.
    A^[i+1] = A^[i] \ {m̃}, L^[i+1] = L^[i] ⋃ {m̃}.
    i = i + 1.
  Else Goto 11
End While
Set the offloaded task set L̃ = L^[i] and minimize the system
cost by solving P3(L̃).
```

Greedy Offloading Selection: We first develop a greedy offloading selection mechanism. In this mechanism (see Algorithm 1), we iteratively select one task to offload in each step until all the tasks are selected or the offloaded task number reaches its limit. In each iteration, the task that can yield the smallest system cost is selected to offload. To this end, $\mathbf{P3}(\mathcal{L})$ needs to be solved at most N times in each iteration. Therefore, the total number of $\mathbf{P3}(\mathcal{L})$ problems required to be solved grows *quadratically* with N. To further reduce the computation complexity, we will propose another offloading selection mechanism.

Individual Sparse Offloading Selection: This offloading selection mechanism works as follows (see Algorithm 2).

Algorithm 2: Individual Sparsity based System Cost Minimization Algorithm

1 Randomly initialize $\mathcal{W}^{[0]}$. Set the maximum iterations as 15, $\epsilon = 10^{-3}$, $p = 1$ and $m = 0$.
2 Repeat
3 Obtain $\mathcal{U}^{[m+1]}$ and $\mathcal{V}^{[m+1]}$ by minimizing the upper bounds $J(\mathcal{U}, \mathcal{V}; \mathcal{W}^{[m]}) = \sum_{n=1}^{N}(w_n^{[m]}u_n^2 + w_{N+n}^{[m]}v_n^2)$
4 Update the weights $\mathcal{W}^{[m+1]}$ according to (6a, 6b).
5 $m = m + 1$.
6 Until convergence or attain the maximum iterations.
7 Calculate the priority coefficients $\Theta = \{\theta_n | n \in \mathcal{N}\}$.
8 Sort the tasks in the ascending order of the value of Θ as $\theta_{\pi_1} <= \theta_{\pi_2} <= \ldots <= \theta_{\pi_N}$, where π is a permutation of $\mathcal{N}$.
9 Set $k_{low} = 0$, $k_{up} = N$.
10 Repeat
11 Set $s = \lfloor (k_{low} + k_{up})/2 \rfloor$.
12 Perform the feasibility test of **P3**$(\{\pi_1, \pi_2, \ldots, \pi_s\})$. If feasible, set $k_{low} = s$; otherwise, set $k_{up} = s$.
13 Until $k_{up} - k_{low} = 1$.
14 Set $s^* = k_{low}$. The offloaded task set $\widetilde{\mathcal{L}}$ is obtained as $\widetilde{\mathcal{L}} = \{\pi_1, \pi_2, \ldots, \pi_{s^*}\}$.
15 Solve **P3**$(\widetilde{\mathcal{L}})$ using interior-point methods to minimize the overall system cost.

First, we solve an individual sparsity inducing norm minimization problem (Lines 1–6), the solutions of which measure the violation of the latency and energy consumption constraints for all tasks. Next, based on the solution, the priority coefficients $\Theta = \{\theta_n | n \in \mathcal{N}\}$, where $\theta_n = (1-\rho_n)u_n / \sum_{i\in\mathcal{N}}(1-\rho_n)u_i + (1-\rho_n)v_n / \sum_{i\in\mathcal{N}}(1-\rho_n)v_i$, are calculated. The user weights are considered in the coefficients. The normalizations ensure that the two constraints make equal contributions to the coefficients. Then, we sort the tasks in the ascending order according to their priority coefficients. Finally, we adopt binary search to progressively remove the tasks that have the largest coefficients (Lines 9–13). The remaining tasks form the final offloaded task set $\widetilde{\mathcal{L}}$. In this mechanism, the overall number of **P3**$(\mathcal{L})$ problems to solve grows *logarithmically* with N.

The individual sparsity inducing norm minimization problem is defined as follows

$$\mathbf{P4}: \min_{\mathcal{H},\mathcal{Q},\mathcal{U},\mathcal{V}} \|\mathcal{U}\|_0 + \|\mathcal{V}\|_0 \tag{4a}$$
$$\mathbf{s.t.}\ T_n^c(\mathcal{N}) - T_n^d <= u_n, \forall n \in \mathcal{N}, \tag{4b}$$
$$E_n^c(\mathcal{N}) - E_n^l <= v_n, \forall n \in \mathcal{N}, \tag{4c}$$
$$q_n <= Q_n, \forall n \in \mathcal{N}, \tag{4d}$$
$$\sum\nolimits_{n\in\mathcal{N}} h_n <= 1, \tag{4e}$$
$$u_n >= 0, v_n >= 0, \forall n \in \mathcal{N}, \tag{4f}$$

where $\mathcal{U} = \{u_n | n \in \mathcal{N}\}$, $\mathcal{V} = \{v_n | n \in \mathcal{N}\}$, and u_n, v_n are two auxiliary variables used to measure the violation of the latency and energy consumption constraints for task n, respectively. The objective of **P4** is to minimize the constraint violation for all tasks.

P4 is NP-Hard because of the l_0 norms. To solve **P4**, we first transform it to the more tractable problem as follows

$$\mathbf{P5}: \min_{\mathcal{H},\mathcal{X},\mathcal{Y},\mathcal{Z},\mathcal{U},\mathcal{V}} \sum_{n\in\mathcal{N}}(u_n^2 + \epsilon^2)^{p/2} + \sum_{n\in\mathcal{N}}(v_n^2 + \epsilon^2)^{p/2} \tag{5a}$$
$$\mathbf{s.t.}\ D_n / e^{y_n} + L_n h_n^{-1} f_c^{-1} - T_n^d <= u_n, \forall n \in \mathcal{N}, \tag{5b}$$
$$D_n e^{-x_n} - E_n^l <= v_n, \forall n \in \mathcal{N}, \tag{5c}$$
$$e^{z_n} <= Q_n, \forall n \in \mathcal{N}, \tag{5d}$$
$$R_n(\mathcal{N}) >= e^{y_n}, \forall n \in \mathcal{N}, \tag{5e}$$
$$e^{z_n} / R_n(\mathcal{N}) <= e^{-x_n}, \forall n \in \mathcal{N}, \tag{5f}$$
$$(4e), (4f),$$

where $\mathcal{X} = \{x_n | n \in \mathcal{N}\}$, $\mathcal{Y} = \{y_n | n \in \mathcal{N}\}$, $\mathcal{Z} = \{z_n | n \in \mathcal{N}\}$. The basic transformation idea is to use the smoothed l_p norms to approximate the objective function of **P4** [13] and to use the similar variable substitution tricks as we do for **P2**$(\mathcal{L})$ to make the feasible set of **P4** convex.

Then, we can develop an iterative reweighted-l_2 algorithm to solve **P5** (see Lines 1–6 in Algorithm 2). By successively minimizing the upper bounds $J(\mathcal{U}, \mathcal{V}; \mathcal{W}^{[m]})$ of the objective function (5a), the iterates $\{(\mathcal{U}^{[m]}, \mathcal{V}^{[m]})\}_{m=1}^{\infty}$ can be obtained. To guarantee the quality of the iterates, we use the elaborately selected upper bounds [13], given by $J(\mathcal{U}, \mathcal{V}; \mathcal{W}^{[m]}) = \sum_{n=1}^{N}(w_n^{[m]}u_n^2 + w_{N+n}^{[m]}v_n^2)$, where

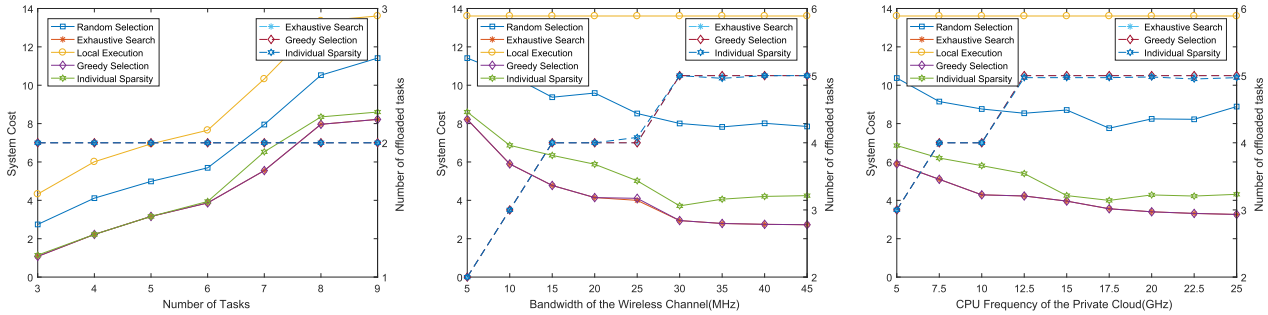
$$w_n^{[m]} = (p/2)((u_n^{[m]})^2 + \epsilon^2)^{\frac{p}{2}-1}, \forall n = 1, \ldots, N, \tag{6a}$$
$$w_{N+n}^{[m]} = (p/2)((v_n^{[m]})^2 + \epsilon^2)^{\frac{p}{2}-1}, \forall n = 1, \ldots, N. \tag{6b}$$

4. Numerical Results

System Setting: The robots are randomly located around the central station, and the coverage radius of the central station is 300 m. For the shared wireless channel, the background noise σ_n^2 is -100 dBm, and the maximum transmission power Q_n is 100 mW. According to the physical interference model [14], we set the channel gain $g_n = d_n^{-\alpha}$, where d_n is the distance between robot n and the central station and $\alpha = 4$ is the path loss factor. Each robot has one task to offload. For each task, the number of CPU cycles required is between $1.0 - 1.5 \times 10^9$, and the input data size ranges from $100-800$ KB, and the deadline is between $1-3$ seconds. For robot n, the allocated computation capacity f_n ranges from $0.5 - 1.2$ GHz, and κ is set as 10^{-11} [5]. The user weight ρ_n is a random value between 0 and 1, and it follows a uniform distribution $U(0, 1)$.

Simulations: There are two baseline algorithms: the *random selection* based algorithm, which randomly selects tasks to offload, and the *exhaustive search* based algorithm, which can give optimal solutions. In each simulation, the individual sparsity based algorithm and the random selection based algorithm both run 100 times to get the average value.

1) Number of tasks: In Fig. 1 (a), we compare the system cost achieved by the proposed algorithms and baselines as the number of tasks increases. From the curves, we find that the proposed greedy selection based algorithm achieves near-optimal performance, and the results given by the proposed individual sparsity based algorithm have a certain gap



(a) $W = 5$ MHz, $f_c = 5$ GHz (b) $N = 9$, $f_c = 5$ GHz (c) $N = 9$, $W = 10$ MHz

Fig. 1 System cost under different settings. The solid curves correspond to the system cost, while the dashed curves correspond to the number of offloaded tasks.

with the optimal results. We also find that with the increasing number of tasks, the system cost also goes up. The reason for this is that the communication and computation resources are limited, and only a few offloading requests can be satisfied.

2) Bandwidth of the wireless channel: We further investigate the impacts of the bandwidth of the wireless channel in Fig. 1 (b). We find that the greedy selection based algorithm also gives near-optimal results with different bandwidth of the wireless channel, and the individual sparsity based algorithm with lower-complexity gives quite good results.

As the bandwidth increases from 5 MHz to 30 MHz, the number of offloaded tasks goes up from 2 to 5. At the same time, the system cost is greatly decreased. It indicates that with the increasing bandwidth, the communication interference among the robots can be mitigated. However, after the bandwidth grows to a certain critical value (e.g., $W = 30$ MHz in Fig. 1 (b)), if we continue to enlarge the bandwidth, the system cost does not have a significant reduction. It implies that at this point another system parameter (e.g., CPU frequency of the private cloud) might become the bottleneck, and merely enlarging the bandwidth is not a wise action.

3) CPU frequency of the private cloud: In Fig. 1 (c), we evaluate the performance of the proposed algorithms and baselines with varying computation capacity of the private cloud. We find that the results given by the greedy selection based algorithm are very close to the optimal results, and the individual sparsity based algorithm with lower-complexity still achieves quite good performance.

From the curves, as the CPU frequency of the private cloud grows from 5 GHz to 12.5 GHz, the offloaded task number increases from 3 to 5, and the system cost also drops. Another observation is that if the CPU frequency exceeds a certain critical value (e.g., $f_c = 17.5$ Ghz in Fig. 1 (c)), continuing to raise the CPU frequency does not bring in obvious benefits. It is the similar to the case that we find in Fig. 1 (a).

4) Summary: With cloud offloading, quite a proportion of the system cost can be reduced by the proposed algorithms as compared to local execution. Moreover, the system cost reduction is closely related to the ratio of offloaded tasks. Given the fixed total task number, with more tasks offloaded, more system cost can be reduced (see Fig. 1 (b) and 1 (c)).

The proposed greedy selection based algorithm can achieve near-optimal performance. Although the individual sparsity based algorithm with lower-complexity has a certain performance gap (22% on average) with the exhaustive search algorithm, it is obviously superior to the random selection based algorithm.

5. Conclusions

In this letter, we investigate the computation offloading problem in cloud based multi-robot systems. Two offloading selection and resource allocation algorithms are proposed to minimize the system cost. Numerical results show that the proposed algorithms can give quite good or even near-optimal results. For future investigation, we will explore multiple wireless channels or clouds to further improve current cost aware offloading schemes.

References

[1] J. Wan, S. Tang, H. Yan, D. Li, S. Wang, and A.V. Vasilakos, "Cloud robotics: Current status and open issues," IEEE Access, vol.4, pp.2797–2807, 2016.

[2] S. Dey and A. Mukherjee, "Robotic slam: A review from fog computing and mobile edge computing perspective," Proc. 13th Int. Conf. on Mob. & Ubiquit. Syst.: Comput. Netw. & Serv., New York, NY, USA, pp.153–158, ACM, 2016.

[3] K. Liu, J. Peng, H. Li, X. Zhang, and W. Liu, "Multi-device task offloading with time-constraints for energy efficiency in mobile cloud computing," Future Gener. Comp. Sy., vol.64, pp.1–14, 2016.

[4] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," IEEE Trans. Veh. Technol., vol.66, no.4, pp.3435–3447, April 2017.

[5] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," Proc. 35th IEEE Int. Conf. on Comp. Commun., pp.1–9, April 2016.

[6] W. Sun, J. Xu, Q. Liang, and Y. Lv, "A framework for energy-optimal cloud robotic system under mimo wireless channel," Proc. 2015 Int. Conf. on Comput. Sci. & Mech. Autom., pp.113–118, Oct. 2015.

[7] P. Pandey, D. Pompili, and J. Yi, "Dynamic collaboration between networked robots and clouds in resource-constrained environments," IEEE Trans. Autom. Sci. Eng., vol.12, no.2, pp.471–480, April

2015.

[8] A. Rahman, J. Jin, A. Cricenti, A. Rahman, and D. Yuan, "A cloud robotics framework of optimal task offloading for smart city applications," Proc. 2016 IEEE Glob. Commun. Conf., pp.1–7, Dec. 2016.

[9] A. AL-Shuwaili, O. Simeone, A. Bagheri, and G. Scutari, "Joint uplink/downlink optimization for backhaul-limited mobile cloud computing with user scheduling," IEEE Trans. Signal. Inform. Process. Netw., in print, 2017.

[10] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge," Proc. 2017 IEEE Int. Conf. on Robot. & Autom., pp.1386–1383, May 2017.

[11] C. Guo, B. Liao, L. Huang, X. Lin, and J. Zhang, "On convexity of fairness-aware energy-efficient power allocation in spectrum-sharing networks," IEEE Commun. Lett., vol.20, no.3, pp.534–537, March 2016.

[12] S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.

[13] Y. Shi, J. Cheng, J. Zhang, B. Bai, W. Chen, and K.B. Letaief, "Smoothed l_p-minimization for green cloud-ran with user admission control," IEEE J. Sel. Areas Commun., vol.34, no.4, pp.1022–1036, April 2016.

[14] T.S. Rappaport et al., Wireless Communications: Principles and Practice, Prentice Hall PTR New Jersey, 1996.