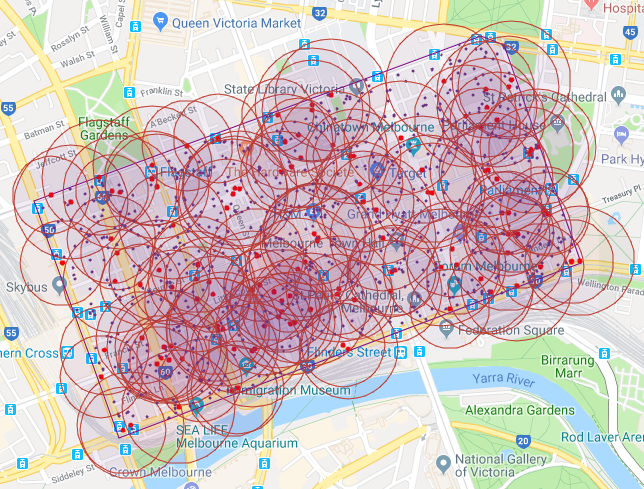
EVALUATION & RESULTS

We evaluated the performance of our proposed algorithm to achieve energy minimal task offloading with users’ mobility and delay constraints into consideration. we used data of base stations within the Melbourne central business district area in Australia Figure 3 Melbourne CBD Base stations, which has a total area of 6.2 km2 provided by the Australian Communications and Media Authority [6]. There are a total of 126 base stations situated inside the Melbourne CBD area.



*Figure 3 Melbourne CBD Base stations*

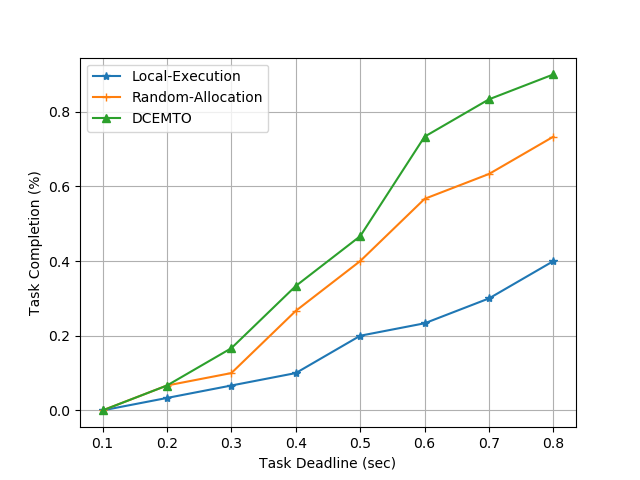
We follow the Random waypoint mobility model for the users moving in the area covered by the base stations provided in the dataset. The random waypoint model is a random model for the movement of mobile users, and how their location, velocity and acceleration change over time.

Without loss of generality, we refer to the parameter settings in the existing works [2], [7]. The coverage radius of BS is randomly taken varying from 70 to 100 meters and the associated MEC’s CPU capacity is in [7, 20] GHz, computation power is in [3,5] watts and transmission power ranges from 0.1 to 1 watt. The bandwidth of the communication channel is set up as 1MHz. Users are moving with the speed of 1~3 MPH and generate tasks with input data size varies from 1 MB to 3 MB having CPU requirement is in [1,10] GHz and deadline constraint within [0.1,1] seconds. Users’ mobiles are having CPU capacity ranges from 1 to 10 GHz, transmission power is randomly taken from [7,15] watts and computation power from [7,10] watts. We consider mobile-execution and random-allocation as our baseline algorithms:

* Local-Execution: Feasible Tasks satisfying delay constraint are executed within the mobiles without offloading.
* Random-Allocation: Each user’s task is assigned to a server randomly chosen from the servers present along the user’s trajectory.

*Table 2 Simulation Parameters*

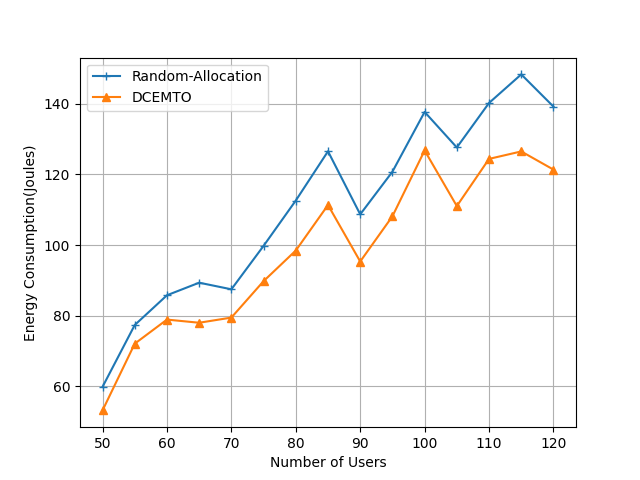
|  |  |
| --- | --- |
| Parameter | Value/Range |
| Coverage radius of BS | 70 to 100 meters |
| MEC’s CPU capacity | [7, 20] GHz |
| MEC’s computation power | [3,5] watt |
| MEC’s transmission power | [0.1,1] watt |
| Channel bandwidth | 1MHz |
| User’s mobility speed | 1~3 MPH |
| Input data size | [1,3] MB |
| Task CPU requirement | [1,10] GHz |
| Task deadline constraint | [0.1,1] sec |



*Figure 4 Task Completion% vs Task Deadline*

We fixed the number of users to 30 and plotted Task Completion percentage by varying Task deadline constraint in Figure 4. It depicts the higher Task completion percentage in DCEMTO compared to the other two approaches. DCEMTO’s task completion is 9.16% higher than Random-Allocation and 27.08% higher than Local-Execution on average. Also, Task completion percentage increases by increasing Deadline constraint.

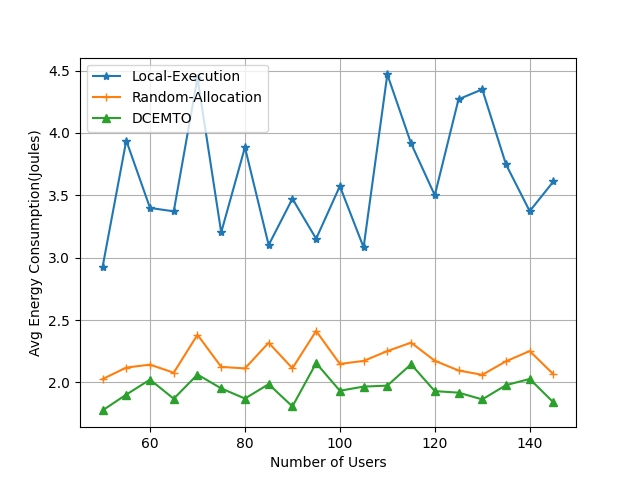
Figure 5 shows how energy consumption varies with the change in the number of users. In general, total energy consumption increases with an increase in the number of users. But the plot is not continuously increasing. It is due to less task completion percentage in some cases. Hence, in the next study, Average Energy Consumption is compared to a varying number of users. DCEMTO’s energy consumption is 11.22% efficient than Random-Allocation method.



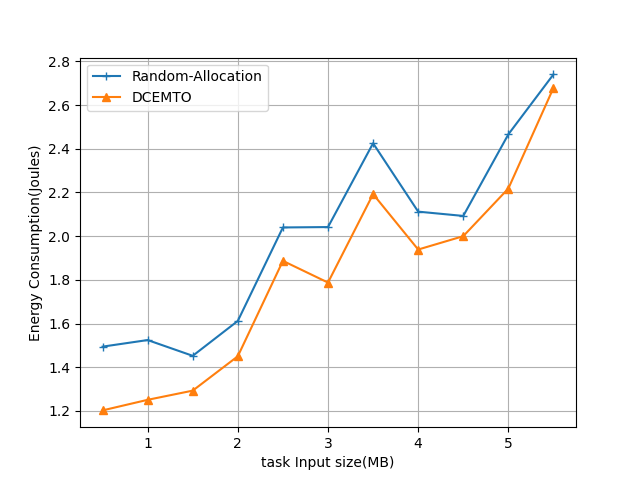
*Figure 5 Energy Consumption vs Number of Users*

Figure 6 indicates the variation of average energy consumption with a change in the number of users. DCEMTO outperforms other approaches. Average energy consumption in DCEMTO is 10.5% energy efficient than Random-Allocation and 46.5% efficient than Local-Execution.

Figure 7 shows how energy consumption varies with varying task input size. It’s clear that higher the task input size, energy consumption increases. It is due to higher transmission energy needed in task offloading. Energy consumption in DCEMTO is 9.55% efficient than Random-Allocation approach.

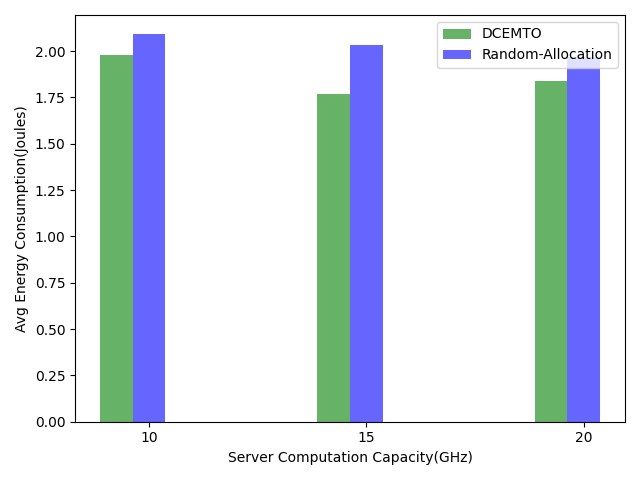


*Figure 6 Avg Energy Consumption vs Number of Users*



*Figure 7 Task Input Size vs Energy Consumption*

Figure 8 depicts the difference in energy consumption with the change in MEC server’s computation capacity. Independent of server’s computation capacity, the energy consumption of DCEMTO is lesser compared to Random-Allocation approach. Energy consumption in DCEMTO is 8.22% efficient than Random-Allocation approach.



*Figure 8 Energy Consumption vs Server Computation Capacity*