We consider that each user u ∈U has one computation task at a time, denoted as Tu , that is atomic and cannot be divided into subtasks. Each computation task Tu is characterized by

a tuple of two parameters, du ,cu , in which du [bits] specifies the amount of input data necessary to transfer the program execution (including system settings, program codes, and in-put parameters) from the local device to the MEC server, and cu [cycles] specifies the workload, i.e., the amount of computation to accomplish the task. The values of du and cu can be obtained through carefully proﬁling of the task execution [7],

The number of CPU cycles for computing 1-bit of input data at vehicle $i$ is denoted as $c\_0$ \cite{Zhang2017}, which is indivisible and cannot be broken down into smaller components \cite{Saleem2021}. $c\_0$ can be obtained through carefully profiling of the task execution \cite{Yang2015}. We consider that each vehicle $v\in \mathcal{V}$ has a different computation task at a time, denoted as $$, is defined by a tuple consisting of two parameters, $$, in which $c\_{i,e}$ [cycles] specifies the workload, i.e. the amount of computation cost to accomplish the task, $c\_{i,e}$ can be obtained through $c\_{0}\*d\_{i,up}$ \cite{Shuang2021}. Each task should be offloaded to the MEC server and then transmission to the cloud server. By offloading the computation task to the MEC server, the vehicles would get more computing resources, however, it would consume additional time for sending the task input in the uplink.

Each computational task, denoted as $T\_i$, is defined by a tuple consisting of two parameters