Abstract—We address reducing carbon footprint (CF) in the context of edge computing. The carbon intensity of electricity supply largely varies spatially as well as temporally. We consider optimal task scheduling and offloading, as well as battery charging to minimize the total CF. We formulate this optimization problem as a mixed integer linear programming model. However, we demonstrate that, via a graph-based reformulation, the problem can be cast as a minimum-cost flow problem, and global optimum can be admitted in polynomial time. Numerical results using real-world data show that optimization can reduce up to 83.3% of the total CF.

Index Terms—Carbon footprint, scheduling, edge computing.

I. INTRODUCTION

THE CARBON footprint (CF) generated from edge computing networks can be estimated by considering the amount of consumed electricity and its carbon intensity (CI). CI differs greatly by time and space. For example, for the day of 22nd Jan. 2023, the CI in Germany is more than ten times higher than that in Sweden. Locally in Germany for the same day, the CI at 24:00 is 21.7% less than that at 12:00 [1]. The variation is due to the difference in the availability of various power sources over time and space.

The studies in [2], [3], [4], [5], [6], [7], [8], [9], [10] have addressed energy efficiency or CF in edge computing or fog computing. In [2], an energy management framework with distributed renewable energy resources is proposed for enabling a sustainable edge computing paradigm. Considering the mobility of users, the authors of [3] optimize the placement of edge computing networks in urban environments to minimize outages while guaranteeing energy efficiency. The authors of [4] propose a novel edge intelligent energy modeling approach based on Elman Neural Network and feature selection to optimize the energy efficiency of edge servers. The works in [2], [3], [4] do not consider CF. The authors of [5] provide a model that can estimate the CF in distributed data centers. The authors of [6] provide a Lyapunov-based algorithm for a distributed data center to minimize electricity cost subject to CF limit. In [7], the authors consider minimizing the CF for video streaming in fog computing networks. The studies in [8] and [9] provide application placement methods to minimize the CF in fog computing. These works in [5], [6], [7], [8], [9] consider optimizing either the spatial or temporal dimension, without joint spatio-temporal optimization. The authors of [10] make use of the spatial and temporal difference of CI to minimize the CF. However, it does not consider energy sharing.

In this letter, we consider task scheduling and offloading, as well as energy sharing with battery charging to minimize the CF, utilizing temporal and spatial information of CI. Due to the discrete nature of task offloading, the CF minimization problem leads to a mixed integer linear programming (MILP) model. However, we reveal that the problem structure admits a reformulation using minimum-cost flow, implying that its global optimum can be computed in polynomial time. Thus our optimization approach is scalable. Numerical results show that, by CF-aware task offloading and scheduling along with energy sharing, we can reduce up to 83.3% of the total CF.

II. SYSTEM MODEL AND PROBLEM FORMULATION

A. System Model

We consider an edge computing network consisting of $S$ sites (Fig. 1 shows an example where 3 sites are illustrated...
by dashed rectangles). Let $S = \{1, 2, \ldots, S\}$. Each site consists of an edge computing server, a battery, a local renewable energy source, and the local power grid. The servers have the same specification and computing power. We use the index of the site to refer to its components, e.g., server $s$, battery $s$, etc. More specifics of the system model are as follows:

- A server can be powered by its local grid, its renewable energy source, and all the batteries.
- A local battery can be charged by the local grid and local renewable energy source. The battery can provide energy to any other server via a battery management system (BMS), though this comes with an energy transfer loss.

**B. Time Horizon**

We consider a scheduling horizon of $T$ time slots, denoted by $T = \{1, 2, \ldots, T\}$. The set of tasks over the entire time horizon is represented by $N = \{1, 2, \ldots, N\}$. In this letter, the tasks are assumed to be of the same type, i.e., the amount of energy to complete them is uniform, denoted by $E$. We remark that we consider the assumption for simplifying the problem, however, we still believe that it is a reasonable assumption, and our study can provide valuable insights for specific edge computing networks, e.g., a multimedia edge computing IoT system that deals with same-size video chunks [11]. W.l.o.g, we use $E$ as our energy unit, and the relevant parameters (e.g., parameters $R$, $I$, $\alpha$, $\beta$, $L$, and $H$ to be introduced next) are all normalized by $E$, i.e., they are specified in multiples of $E$.

Task $n$ is represented by a tuple $(o_n, d_n, s_n, S_n)$, where $o_n, d_n \in T$, $s_n \in S, S_n \subseteq S$, such that

- $o_n$ is the time slot when the task is generated;
- $d_n$ is the deadline for completing the task;
- $s_n$ is the server that the task is initially associated with;
- $S_n$ represents the set of candidate servers that can perform task $n$. In other words, some servers that cannot take over for some reason will not be included in $S_n$.

Task $n$ is to be completed in time interval $[o_n, d_n]$.

Let us denote by $\pi_{nst}$ a binary variable that is one if and only if task $n$ is completed by server $s$ in time slot $t$. Any task needs to be completed only once, thus we have

$$\sum_{s \in S} \sum_{t = o_n} d_n = 1, \forall n \in N. \quad (1)$$

Due to normalization, a task requires one unit of energy. Thus in time slot $t$, the amount of energy consumed for task completion by server $s$ is $\sum_{n \in N} \pi_{nst}$. Of the energy consumed, we use variables $x_{nst}, y_{s'nsst},$ and $z_{st}$ to represent the amount of energy (again normalized by $E$) by server $s$ using the local grid, battery $s'$ ($s \in S$), and the local renewable source, respectively, in time slot $t$. Then we have

$$x_{nst} + \sum_{s' \in S} y_{s'nsst} + z_{st} = \sum_{n \in N} \pi_{nst}, \forall s \in S, t \in T. \quad (2)$$

Note that each server has a computing capacity, $H$, in terms of how many tasks it can process in a time slot, given its computational resource. Additionally, from an energy perspective, the tasks’ energy demands are normalized, such that the number of tasks server $s$ performs in a time slot numerically equals the amount of energy consumed. The number of tasks performed by server $s$ in time slot $t$ is bounded by the server’s computing capacity, resulting in the following constraint:

$$x_{nst} + \sum_{s' \in S} y_{s'nsst} + z_{st} \leq H, \forall s \in S, t \in T. \quad (3)$$

Variables $u_{st}$ and $v_{st}$ represent the amount of energy from the local grid and the renewable source for charging, respectively. Denote by variable $w_{st}$ the amount of remaining energy of battery $s$ at the end of time slot $t$. Denote by $L$ the battery capacity in terms of how many tasks the battery can support to perform with respect to energy consumption. In time slot $t$, we have the battery capacity constraint:

$$u_{st} + v_{st} + w_{st} \leq L, \forall s \in S, t \in T \setminus \{1\}. \quad (4)$$

By the end of time slot $t$, $w_{st}$ is given by

$$w_{st} = u_{st} + v_{st} + w_{st}(t-1) - \sum_{s' \in S} y_{s'nsst}, \forall s \in S, t \in T \setminus \{1\}. \quad (5)$$

Denote by $R_{st}$ the amount of renewable energy available from source $s$ in time slot $t$. This energy is used for charging the local battery and supplying the local server, thus we have

$$z_{st} + v_{st} \leq R_{st}, \forall s \in S, t \in T. \quad (6)$$

**C. Carbon Footprint and MILP Formulation**

In the system model, CF occurs in three processes related to the grids. Denote by $I_{st}$ the CF of local grid $s$ in time slot $t$. The three types of CF can be calculated as follows.

1) **Task completion:** CF will occur when a server consumes the energy from the local grid to complete some task. The total amount of CF of this can be expressed as

$$CF^G(x) = \sum_{t \in T} \sum_{s \in S} I_{st} x_{nst}. \quad (7)$$

2) **Battery charging:** The total amount of CF due to battery charging via the local grid can be calculated by

$$CF^B(u) = \sum_{t \in T} \sum_{s \in S} I_{st} u_{st}. \quad (8)$$

3) **Task offloading:** We consider the worst-case CF in task offloading, i.e., we assume that energy used for task offloading is entirely from the grid. We use $\alpha_{sns}$ to represent the required amount of the energy for transferring task $n$ from its initial server $s_n$ to $s$ ($\alpha_{sns} = 0$ if $s_n = s$). The total CF of task offloading in the network is given by

$$CF^O(\pi) = \sum_{n \in N} \sum_{s \in S} \sum_{t = o_n} d_n \alpha_{sns} I_{st} \pi_{nst}. \quad (9)$$

In addition, there will be some loss in energy transfer between a battery and the servers of other sites. We convert this loss into an equivalent amount of CF. Denote the loss per unit of energy from battery $s'$ to server $s$ by $\beta_{ss'}$ ($\beta_{ss'} = 0$ if $s' = s$). We assume that local grid $s$ provides energy to make up for the loss, and the total amount of the equivalent CF is thus

$$CF^L(y) = \sum_{s \in S} \sum_{t \in T} \sum_{s' \in S} \beta_{ss'} I_{st} y_{s'sst}. \quad (10)$$
Fig. 2. The graph for which finding the minimum-cost flow gives the global optimum.

We consider minimizing the total CF in the edge computing network. The problem can be formulated by MILP as follows,
\[
\min_{x,y,z,w,u,v,\pi,\theta,\xi,\lambda,\delta,\kappa,\rho,\beta} \text{CF}_G(x) + \text{CF}_B(u) + \text{CF}_O(\pi) + \text{CF}_L(y)
\]
subject to (1)–(6),
\[
\text{(11)}
\]
where the objective function (11) is the overall system CF.

III. PROBLEM SOLVING

Although the CF minimization problem is formulated as MILP in (11), we will demonstrate that the global optimum can be computed in polynomial time. The idea is to construct a graph, along with entities for nodes and arcs, such that the problem maps to finding a minimum-cost flow in the graph. The graph construction, however, is non-trivial. We illustrate the concept in Fig. 2 and detail the construction below.

A. Overview

A minimum-cost flow problem is the optimization problem of finding the cheapest possible way to route flow from supply node(s) to demand node(s) in a directed graph [12]. In the graph, every arc has two attributes: 1) the per-unit flow cost, and 2) flow capacity. A feasible flow solution has to satisfy

1) Flow balance constraint: The total incoming flow to a node plus the node’s supply, if any, equals the total outgoing flow of the node plus the node’s demand, if any.

2) Capacity constraint: The flow on every arc does not exceed the arc’s capacity.

In our graph shown in Fig 2, the green and orange nodes are nodes with supply and demand, respectively; the other nodes are all transshipment nodes with no supply nor demand. In the graph, green θ-nodes and node δ, red (κ, ρ) node pairs, blue (λ, ε) node pairs, and orange τ-nodes represent the renewable sources, the merge of all the local power grids, batteries, servers, and tasks, respectively. The flows in the graph represent the amount of energy, and the per-unit cost of an arc accounts for the CF. Note that the per-unit costs and the arc capacities are set by default to zero and infinite, respectively, unless specified below. We use (a → b) to represent the arc from node a to node b.

B. The Entities in a Time Slot

There are T sections indexed by time slot t (t ∈ T) in the graph. For each of them, there are three types of entities, namely the θ-nodes, (κ, ρ) node pairs, and (λ, ε) node pairs.

1) Green node θ_s^t represents renewable source s in time slot t. Its supply in the graph is set to be R_st, i.e., the amount of energy available for s and t. The meaning of the flows on arcs (θ_s^t → λ_s^t) and (θ_s^t → κ_s^t) are the same as those of the variables z_st and v_st, respectively. By graph construction, the two flows adhere to constraint (6). Node θ_s^t also connects to the orange demand node μ used for receiving the surplus flows (i.e., surplus energy) to guarantee the feasibility of the minimum-cost flow problem.

2) A red node pair (κ_s^t, ρ_s^t) represents battery s in time slot t. The capacity of arc (κ_s^t → ρ_s^t) equals the capacity of battery s, i.e., L_s. The flows on arcs (δ → κ_s^t), (θ_s^t → κ_s^t), and (ρ_s^{t-1} → κ_s^t) correspond to variables u_st, v_st, and w_st(t−1), respectively, and they satisfy constraint (4) due to flow balance.

3) A blue node pair (λ_s^t, ε_s^t) represents server s in time slot t. The flows on arcs (δ → λ_s^t), (ρ_s^t → λ_s^t), and (θ_s^t → λ_s^t) represent variables x_st, y_st, and z_st, respectively. The flows submit to constraint (2) by flow balance. In addition, the capacity of arc (λ_s^t → ε_s^t) equals H, to model constraint (3). The flow through arc (ρ_s^t → λ_s^t) represents that battery s’ provides energy to server s in time slot t, so the per-unit cost of the arc equals β_s'.I_st,
and it accounts for the energy loss in task offloading. Clearly, the total flow cost of these arcs equals (10).

C. Battery Level Evolution

Between any two neighboring time slots $t$ and $t+1$ in the graph, there are $S$ arcs, i.e., $(\rho^t_s \rightarrow \kappa^{t+1}_s), \forall s \in S$. The arcs model the evolution of the battery energy between time slots $t$ and $t+1$. The flow on arc $(\rho^t_s \rightarrow \kappa^{t+1}_s)$ is the remaining energy in battery $s$ at the end of $t$. By flow balance, the flow on arc $(\rho^t_s \rightarrow \kappa^{t+1}_s)$ equals the flow entering node $\rho^t_s$ minus the flows representing energy sharing on arcs $(\rho^t_s \rightarrow \lambda^t_{s'}) \forall s' \in S$, and this flow balance implies constraint (5).

D. The Grid

The local power grids are merged into a green node $\delta$, and we identify the individual local grids via the arcs to the nodes corresponding to batteries and servers:

1) Flow can be sent through arc $(\delta \rightarrow \kappa^t_s)$, and this represents that local grid $s$ charges battery $s$ in time slot $t$. Clearly, its per-unit cost is $I_{st}$ to account for CF in battery charging, and the total flow cost of these arcs equals (8).

2) We use the flow through arc $(\delta \rightarrow \lambda^t_s)$ to represent that local grid $s$ provides energy to server $s$ in time slot $t$, and the unit flow cost of arc $(\delta \rightarrow \lambda^t_s)$ is set to be $I_{st}$.

The sum of flow costs of these arcs is equivalent to (7). Arc $(\delta \rightarrow \mu)$ is used for routing the surplus flows from the grid. The flow on the arc represents the amount of grid energy not used by the edge computing network. The amount of supply of node $\delta$ is set to be a sufficiently large value, e.g., $N$.

E. The Tasks

In the task section of the graph, an orange demand node $\tau_n$ represents task $n$. Its demand is one (energy unit). The arcs between the $\varepsilon$-nodes and $\tau$-nodes are introduced based on the information of the tasks. Specifically, for any task $\tau$-node $n$, it only connects to those $\varepsilon$-nodes that represent candidate servers in $S_n$. The presence of arc $(\varepsilon^t_n \rightarrow \tau_n)$ represents that server $s$ can complete task $n$ in time slot $t$ ($o_n \leq t \leq d_n$). Thus, the flow on this arc represents variable $\pi_{nst}$, and the flow balance constraint for the task node is equivalent to (1). The per-unit cost of arc $(\varepsilon^t_n \rightarrow \tau_n)$ is set to be $\alpha_{s\tau_n}I_{s\tau_n}$, and the sum of flow costs on these arcs is equal to (9).

F. Integrality and Complexity

With the graph constructed, we can obtain the optimal solution to (11) by tracing the corresponding optimal flow. Note that the potential difficulty of (11) is that $\pi$ is binary. Although the flows are not required to be integer, the integrality theorem of minimum-cost flow problems guarantees that there always exists an integer optimal solution [12, Th. 11.5]. Given an integer optimal solution, for each task node, only one unit of flow will be routed from some $\varepsilon$-node as the demand of a task node is one, i.e., a task will not be split between multiple servers at optimum.

In addition, The graph has in total $O(ST + N)$ nodes and $O(S^2 + NST)$ arcs, hence the graph construction can be completed in polynomial time. The complexity of the network simplex algorithm using dynamic trees [13] is $O(\text{ST} + \text{NS}\text{T}^2 + \text{N}\text{ST}) \log(\text{ST} + N) \log(\text{STC} + NC)$, where $C$ is the maximum cost of any arcs. Thus, the CF minimization problem (11) can be solved in polynomial time.

IV. PERFORMANCE EVALUATION

In this section, we use a 24-hour CI data set of Sweden, Germany, and Poland from [1] for performance evaluation. In our simulation, the length of a time slot is an hour. The amount of renewable energy follows a binomial distribution $B(5, 0.5)$ during daytime (7 am to 7 pm), and is zero otherwise. For any task $n$, $o_n$, $d_n$ ($o_n < d_n$), and $s_n$ all follow a discrete uniform distribution. In addition, we set $S_n = S \forall n \in N$, i.e., a task can be offloaded to any server. Parameters $T, S, N, \alpha, \beta$, and $E$ are set to be 24, 3, 100, 0.1, 0.2, and 10 respectively.

Fig. 3 shows the performance results of our proposed scheme with respect to battery capacity $L$ and server capacity $H$. There is a dramatic reduction of CF when the battery capacity goes from zero (i.e., no battery) to 5 units. Thus the importance of battery (and BMS) to creating an energy buffer for pursuing low CF is apparent. When $L > 5$, the total CF is hardly improved as such a capacity level can store all the energy required by the tasks. In addition, when the server capacity increases, the total CF decreases as more tasks can be offloaded to low-CI regions. We compare the performance of four schemes:

- **S1**: This is the proposed scheme considering both task offloading and energy sharing via BMS in the network.
- **S2**: This scheme allows task offloading but the BMS is disabled, i.e., a battery can only provide energy locally.
- **S3**: This is the opposite to S2, namely energy sharing is enabled, but the tasks cannot be offloaded.
- **S4**: In the last scheme, task offloading and energy sharing are both disabled; this corresponds to the most basic benchmark for comparison.

In the simulation, we obtain the performance of the four schemes by solving the corresponding minimum-cost flow problems. Figs. 4 and 5 show the performance results with respect to battery capacity $L$ and server capacity $H$, where we use w/ and w/o to represent with and without, respectively. Overall, compared with the conventional network (scheme S4), task offloading and energy sharing can help significantly reduce the total CF, up to 83.3% by Fig. 5 for $L = 5$ and $H = 10$. 

Fig. 3. Total CF with respect to battery capacity $L$ and server capacity $H$. 

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
V. Conclusion

We have investigated a CF minimization problem for edge computing through joint task offloading and energy sharing. By exploiting the problem’s structure, we proposed a graph-based approach that enables finding the global optimum in polynomial time. The approach can be implemented in practical systems due to its consideration of real-world parameters and adaptability to various edge computing architectures. Numerical results demonstrate the significant potential of using spatial and temporal information of CI for reducing the CF, as well as the benefits of joint task offloading and energy sharing. Future work encompasses extending the model to accommodate non-uniform energy consumption of tasks and exploring additional aspects such as scalability, security, and robustness.

REFERENCES

[1] “Electricity maps.” 2023. [Online]. Available: https://app.electricitymaps.com
[2] W. Li et al., “On enabling sustainable edge computing with renewable energy resources,” IEEE Commun. Mag., vol. 56, no. 5, pp. 94–101, May 2018.
[3] P. Vitello, A. Capponi, C. Fiandrino, G. Cantelmo, and D. Kliazovich, “Mobility-driven and energy-efficient deployment of edge data Centers in urban environments,” IEEE Trans. Sustain. Comput., vol. 7, no. 4, pp. 736–748, Oct.–Dec. 2022.
[4] Z. Zhou, M. Shojafat, J. Abawajy, H. Yin, and H. Lu, “ECMS: An edge intelligent energy efficient model in mobile edge computing,” IEEE Trans. Green Commun. Netw., vol. 6, no. 1, pp. 238–247, Mar. 2022.
[5] W. Van Heddeghem, W. Vereecken, D. Colle, M. Pickavet, and P. Demeester, “Distributed computing for carbon footprint reduction by exploiting low-footprint energy availability,” Future Gener. Comput. Syst., vol. 28, no. 2, pp. 405–414, 2012.
[6] A. Radovanovic et al., “Carbon-aware computing for datacenters,” IEEE Trans. Power Syst., vol. 38, no. 2, pp. 1270–1280, Mar. 2023.
[7] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong, “A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing,” in Proc. Int. Conf. Inf. Netw. (ICOIN), 2015, pp. 324–329.
[8] M. Aldossary and H. A. Alharbi, “Towards a green approach for minimizing carbon emissions in fog-cloud architecture,” IEEE Access, vol. 9, pp. 131720–131732, 2021.
[9] E. Ahvar, S. Ahvar, Z. Á. Mann, N. Crespi, R. Githo, and J. Garcia-Alfaro, “DECA: A dynamic energy cost and carbon emission-efficient application placement method for edge clouds,” IEEE Access, vol. 9, pp. 70192–70213, 2021.
[10] C.-S. Yang, C.-C. Huang-Fu, and L.-K. Fu, “Carbon-neutralized task scheduling for green computing networks,” 2022, arXiv:2209.02198.
[11] C. Long, Y. Cao, T. Jiang, and Q. Zhang, “Edge computing framework for cooperative video processing in multimedia IoT systems,” IEEE Trans. Multimedia, vol. 20, no. 5, pp. 1126–1139, May 2018.
[12] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, Network Flows: Theory, Algorithms, and Applications. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.
[13] R. E. Tarjan, “Dynamic trees as search trees via euler tours, applied to the network simplex algorithm,” Math. Program., vol. 78, no. 2, pp. 169–177, 1997.