MHCDMC Rouding

Qinghui Zhang, *Student Member, IEEE,* Weidong Li, *Member, IEEE,* Qian Su, *Member, IEEE* and Xuejie Zhang, *Member, IEEE*

***Abstract*—The abstract goes here.**

***Index Terms*—high performance computing; resource allocation; scheduling; approximation algorithm; bag-of-tasks.**

I. Introduction

With their maneuverability and increasing affordability, unmanned aerial vehicles (UAVs) have many potential applications in wireless communication systems [1]. In particular, UAV-mounted mobile base stations (MBSs) can be deployed to provide wireless connectivity in areas without infrastructure coverage such as battlefields or disaster scenes. Unlike terrestrial base stations (BSs), even those mounted on ground vehicles, UAV-mounted MBSs can be deployed in any location and move along any trajectory constrained only by their aeronautical characteristics, in order to cover the ground terminals (GTs) in a given area based on their known locations.

After a major natural disaster, the ground-based communication facilities are usually destroyed and communication is interrupted, and important communication information is blocked, which endangers the lives of the affected people and aggravates the difficulty of post-disaster rescue. UAVs have wide application prospects in the field of emergency communication because of their advantages such as rapid deployment and the ability to provide effective air-ground line-of-sight links to cover the affected areas by equipping emergency base stations [1].

In order to protect people's life and property and speed up the post-disaster reconstruction and recovery work, we need to provide communication security for users as soon as possible. According to the actual communication demand, some potential optional UAV deployment locations are selected by relying on information such as population distribution and disaster situation. Selecting the minimum number of UAVs to restore the communication network in that area under the constraints of communication needs is a critical issue. Since larger UAVs have larger energy reserves compared to smaller UAVs, larger UAVs can transmit signals with higher power and have greater bandwidth capacity to obtain better signal coverage performance. Therefore, in this paper we assume that the UAV base station with higher signal transmitting power has a larger bandwidth capacity.

Qinghui Zhang, Qian Su and Xuejie Zhang are with the School of Information Science and Engineering, Yunnan University, Kunming, Yunnan 650091, P.R. China.

E-mail: zhangqinghui@mail.ynu.edu.cn, suqian@ynu.edu.cn, xjzhang@ynu.edu.cn

Weidong Li is with the Department of Mathematics, Yunnan University, Kunming, Yunnan 650091, P.R. China.

E-mail: weidong@ynu.edu.cn.

(Corresponding author: Xuejie Zhang)

contributions:

- We improved the algorithm proposed by Bandyapadhyay in [1] when the require of user is real.We also analyze the relationship between SINR and power variation on ε in the wireless communication model is analyzed.
- We propose a center-of-gravity-based hierarchical position presetting algorithm to preset the UAV position.
- Since the number and location of deployed UAVs are unknown, the SINR between each user and UAV is preset in this paper. After multiple solving, the gap between the preset value and the real value can be reduced significantly. We evaluated the gap between the pre-set SINR and the real SINR as an indication of the accuracy of the UAV base station deployment sites.

II. Related work

Related work

III. System model

A. Network model

In a disaster area that can be viewed as a plane, we need to restore communication services for n affected users (referred to as users) whose locations are known, and use $U = \{u_1, ..., u_n\}$ denotes the set of users. The location coordinates of user $u_j \in U$ are $loc_j^u = (X_j^u, Y_j^u, 0)$, and the communication bandwidth requirement is BR_j. Based on the distribution of affected users, we intend m candidate UAVs deployment. We use $A = \{a_1, ..., a_m\}$ to denote the set of these m candidate UAVs. The position coordinates of $a_i \in A$ are $loc_i^a = (X_i^a, Y_i^a, h)$, where h is the optimal altitude such that the coverage range of UAV is maximized [2]. In addition, all UAVs have the same power p and bandwidth capacity BW. In order to be able to quickly restore communication for users, under the constraint of bandwidth capacity per UAV, we need to deploy as few UAVs as possible and guarantee the communication quality for each user, i.e., the user's signal-to-noise ratio (SINR) is greater than a minimum SINR threshold $SINR_{min}$. We define the problem as a minimum metric capacitated signal coverage (MMCSC) problem.

The communication between the UAV-enabled MBS and the user uses an air-to-ground communication link in the sub-6 GHz band, where Line of Sight (LoS) dominates. A The path loss between u_j and a_i can be expressed as:

$$L_{ij}(\mathrm{dB}) = 20\lg(d_{ij}) + 20\lg(\frac{4\pi f}{c}) + \eta_{LoS}, \tag{1}$$

where d_{ij} denotes the distance between a_i and u_j, f denotes the carrier frequency, c represents the speed of light, and η_{LoS}

represents the shadow fading loss of LoS, which is a constant. the signal-to-noise ratio between a_i and u_j is:

$$SINR_{ij} = \frac{G_{ij}p}{N_I + N_0}, \tag{2}$$

G_{ij} denotes the channel gain between a_i and u_j, N_I denotes the interference noise power in this environment, and N_0 denotes the white noise power. The channel gain G_{ij} is affected by the path loss and satisfies the following relationship:

$$G_{ij}p(\text{dB}) = p(\text{dB}) - L_{ij}(\text{dB}). \tag{3}$$

According to Eq.(1),(2),(3), we can get the coverage radius of $a_i \in A$ when its power is p, the minimum SINR is $SINR_{min}$:

$$r_i = 10^{[p(\text{dB}) - SINR_{\min}(\text{dB}) - \eta(\text{dB}) - N(\text{dB})]/20}, \tag{4}$$

where $\eta(\text{dB}) = 20\log\frac{4\pi f}{c}(\text{dB}) + \eta_{LoS}(\text{dB})$, $N(\text{dB}) = (N_I + N_0)(\text{dB})$. Since the final selected servers are unknown, the interference noise N_I is also unknown. We hereby set $N_I = 0$, and will further study in section 4 N_I for analysis.

Based on the above definition, we can get the integer programming form of the problem.

$$\begin{aligned}
\min \quad & \sum\nolimits_{a_i \in A} y_i && (5)\\
s.t. \quad & x_{ij} \le y_i, \forall u_j \in U,\ \forall a_i \in A. && (5a)\\
& \sum\nolimits_{u_j \in U} (x_{ij} \cdot BR_j) \le y_i \cdot BW, \forall a_i \in A. && (5b)\\
& \sum\nolimits_{a_i \in S} x_{ij} = 1, \forall u_j \in U. && (5c)\\
& x_{ij} = 0, \forall u_j \in U, \forall a_i \in A : SINR_{ij} < SINR_{\min}. && (5d)\\
& x_{ij} \in \{0,1\}, \forall u_j \in U, \forall a_i \in A && (5e)\\
& y_i \in \{0,1\}, \forall a_i \in A. && (5f)
\end{aligned}$$

Constraint (5a) means that u_j can be served by a_i only after the MBS a_i is selected. Constraint (5b) is the bandwidth resource capacity resource constraint of each MBS, the sum of bandwidth demand of users it serves cannot exceed its own capacity. Constraint (5c) of indicates that every user must be served. Constraint (5d) means that if u_j is served by a_i, then the SINR of u_j has to be greater than $SINR_{min}$. Constraints (5e) and (5f) are two integer decision variable constraints. We relax the integer programming(LP) (5) to be able to obtain its linear programming (6):

$$\begin{aligned}
\min \quad & \sum\nolimits_{a_i \in A} y_i && (6)\\
s.t. \quad & x_{ij} \le y_i, \forall u_j \in U, \forall a_i \in A. && (6a)\\
& \sum\nolimits_{u_j \in U} (x_{ij} \cdot BR_j) \le y_i \cdot BW, \forall a_i \in A. && (6b)\\
& \sum\nolimits_{a_i \in S} x_{ij} = 1, \forall u_j \in U. && (6c)\\
& x_{ij} = 0, \forall u_j \in U, \forall a_i \in A : SINR_{ij} < SINR_{\min}, && (6d)\\
& x_{ij} \ge 0, \forall u_j \in U, \forall a_i \in A && (6e)\\
& 0 \le y_i \le 1, \forall a_i \in A. && (6f)
\end{aligned}$$

We can obtain the optimal solution $\sigma = (x, y)$ of LP 6 in polynomial time, which is a fractional solution. The cost of the LP solution σ (feasible or otherwise), denoted by cost(σ), is defined as $\sum_{a_i \in A} y_i$.

In this solution, for any $x_{ij} > 0$, we can ensure that $SINR_{ij} \ge SINR_{\min}$. In a real scenario, MBS can enhance its own signal coverage through spectrum multiplexing and other methods. We define $SINR'_{\min}(< SINR_{\min})$ as a new minimum SINR after using the multiplexing technique. When $SINR'_{min} \le SINR_{ij} \le SINR_{\min}$, a_i can also provide services with acceptable signal quality for u_j by adopting multiplexing techniques. In addition, MBSs can also increase the signal coverage by enhancing the signal power. Let p' be the power after enhancement.

For the two variables $SINR'_{\min}$ and p', they have a limited variation. In this paper, we do not focus on how small $SINR'_{\min}$ is or how large p' is; we only give the definition of the magnitude of their variable as follows.

$$\varepsilon^p = \frac{p' - p}{p}, \tag{7}$$

$$\varepsilon^{SINR} = \frac{SINR_{\min} - SINR'_{\min}}{SINR_{\min}}. \tag{8}$$

Under the influence of ε^p and ε^{SINR}, r_i follows. When the power of a_i is p' and the minimum SINR is $SINR'_{\min}$, its radius is r'_i. Then we can get the variation ε of the radius:

$$\varepsilon = \frac{r'_i - r_i}{r_i} = 10^{\left[p\varepsilon^p(\text{dB}) + SINR_{\min}\varepsilon^{SINR}(\text{dB})\right]/20} - 1. \tag{9}$$

In the algorithm proposed in this paper, considering ε is able to reduce the number of servers that may eventually be selected, and the performance of our algorithm is also related to it.

We can regard MMCSC problem as a capacitated bipartite graph cover (CBGC) problem. For a bipartite graph $G = (A, U, E)$, where A and U are the two sets of vertex, $E = \{e_{ij} | SINR_{ij} \ge SINR_{\min}, a_i \in A, u_j \in U\}$.The capacity of each vertex $a_i \in A$ is BW, the request of each vertex $u_j \in U$ is BR_j. The definition of CBGC problem is that choosing a subset $A' \subseteq A$ which is capacitated to cover all vertex in U. In this problem, each vertex $a_i \in A$ is like the source of *flow*, and each vertex $u_j \in U$ has flow demand. The flow of a_i can only flow to the connected u_j in G.

IV. Grid-based three-step rounding approximation algorithm

In this section, we present an efficient rounding algorithm for the solution $\sigma = (x, y)$ of LP (6). The rounding algorithm consists of three parts, namely "determining superior and inferior servers", "clustering of servers" and "selecting the final servers". With Alg.1, we can finally get the set of the selected servers A^*. Finally, we demonstrate through theoretical analysis that the approximation ratio of the algorithm is () when the SINR and the power are varied by ε^p and ε^{SINR}, respectively.

Before presenting the above algorithm, we give here two important definitions.

Algorithm 1 Hierarchical Rounding

Input: $(U, A,)$.
Output: A set of selected MBS A^*.
1: $\sigma \leftarrow$ the solution of LP(6).
2: $\overline{\sigma} \leftarrow DSIS(\sigma)$
3: $\widehat{\sigma}, \mathcal{C} \leftarrow CoS(\overline{\sigma})$.
4: $A^* \leftarrow SFS(\widehat{\sigma}, \mathcal{C})$.
5: **return** A^*.

Definition 1. *For a fraction solution $\sigma = (x, y)$ of LP.(6), given a parameter $0 < \alpha \leq 1/2$, and each server with non-zero y_i can be classified according to the relationship with α. If $y_i = 1$, we call a_i a superior server and denote their set by $\mathcal{S}$. If $0 < yi \leq \alpha$, we call a_i an inferior server and denote their set by $\mathcal{I}$.*

Definition 2. *Reroute the flow from a subset of servers $A' \subseteq A$ to a server $a_k \notin A'$, it means that the users $U' \subseteq U$ served by A' will be allocated to a_k to serve. A new solution $(\overline{x}, \overline{y})$ will get from current solution (x, y) as flowing way: (1) $\overline{x}_{kj} = x_{kj} + \sum_{a_i \in A'} x_{ij}, \forall u_j \in U'$; (2) $\overline{x}_{ij} = 0, \forall u_j \in U', \forall a_i \in A'$.*

A. Determining Superior and Inferior Servers

Since the number of inferior servers is directly related to the result of the rounding algorithm, this subsection is to trim the replaceable servers in $\mathcal{I}$. In addition, servers with $\alpha < y_i < 1$ are also added to $\mathcal{S}$ at the end of Alg.2.

The main loop of Alg.2 iterates over each user. For each user $u_j \in U$, if $\alpha < \sum_{a_i \in \mathcal{I}_j} \overline{y}_i \leq 2\alpha$, it means that u_j is served by many inferior servers currently, and we need to eliminate some unnecessary servers among them. We construct a set $\mathcal{I}_j$ of inferior servers currently serving u_j. These servers are contained in a square centered at u_j with side length $4r$. We divide the square into grid cells with side length $(r\varepsilon)/2$. We will get $O(\varepsilon^{-2})$ grid cells in the square. Thus there are $O(\varepsilon^{-2})$ groups in $\mathcal{G}_j$.

In Alg.2, lines 10-14, we choose the server a_m arbitrarily in each group and reroute the flow from the other servers in the group to a_m. a_m can serve every user served by $a_i \in G_l$ because the distance between the servers in G_l does not exceed $r\varepsilon$. The distance to a_m for all users served by a_i is $r\varepsilon + r = (1+\varepsilon)r$. Moreover, the capacity constraint of a_m is not broken after reroute since:

$$\begin{aligned}\sum_{u_j \in U, a_i \in G_l} x_{ij} BR_j &\leq \sum_{a_i \in G_l} \sum_{u_j \in a_i} x_{ij} BR_j \\ &\leq \sum_{a_i \in G_l} y_i BW \\ &= BW \sum_{a_i \in G_l} y_i \\ &\leq BW \cdot 2\alpha \\ &\leq BW,\end{aligned}$$

where $u_j \in a_i$ means u_j served by a_i.

Algorithm 2 Determining Superior and Inferior Servers (DSIS)

Input: A fraction solution of LP(6) σ, a parameter α and the magnitudes of variable of power and SINR ε^p and ε^{SINR}.
Output: A fraction solution $\overline{\sigma} = (\overline{x}, \overline{y})$ containing only superior and inferior servers.
1: Initialize $\overline{x} \leftarrow x$, $\overline{y} \leftarrow y$.
2: $\mathcal{S} \leftarrow \{a_i | \forall a_i \in A \text{ and } \overline{y}_i = 1\}$.
3: $\mathcal{I} \leftarrow \{a_i | \forall a_i \in A \text{ and } 0 < \overline{y}_i \leq \alpha\}$.
4: **for all** $u_j \in U$ **do**
5: **if** $\sum_{a_i \in \mathcal{I}: \overline{x}_{ij} > 0} \overline{y}_i > \alpha$ **then**
6: Construct an set $\mathcal{I}_j$ of inferior servers serving u_j such that $\alpha < \sum_{a_i \in \mathcal{I}_j} \overline{y}_i \leq 2\alpha$.
7: Cut the square centered at u_j with side length $4r$ to grid cells, and each cell has side length $(r\varepsilon)/2$.
8: For each grid cell that contains at least one server, creating a group G_l consisting of all servers contained in the grid cell.
9: $\mathcal{G}_j \leftarrow$ all groups in the square centered at u_j.
10: **for all** $G_l \in \mathcal{G}_j$ **do**
11: $a_m \leftarrow$ arbitrary server contained in G_l.
12: Reroute the flow from $\forall a_i \in G_l \backslash \{a_m\}$ to a_m.
13: $\overline{y}_m \leftarrow 1$; $\overline{y}_i \leftarrow 0, \forall a_i \in G_l \backslash \{a_m\}$.
14: **end for**
15: **end if**
16: **end for**
17: $\overline{y}_i \leftarrow 1, \forall a_i \in A, \alpha < y_i < 1$.
18: **return** $\overline{\sigma}$.

At the end of Alg.2, all servers with $\alpha < y_i < 1$ are turned into superior servers. Since then, after the processing of Alg.2, only inferior and superior servers are left in $\overline{\sigma}$ (without considering the servers with $y_i = 0$).

Lemma 1. *For each user $u_j \in U$, we have that*

$$\sum_{a_i \in \mathcal{I}: \overline{x}_{ij} > 0} \overline{y}_i \leq \alpha. \tag{10}$$

B. Clustering of Servers

After determining superior and inferior servers, note that there can be many inferior servers in $\overline{\sigma}$. Including all these servers in the final solution may incur a huge cost. Clustering of Servers in this subsection is a significant stage for our rounding algorithm. We use a reasonable strategy based on rerouting flow from some servers to others to select a small number of servers. For a selected inferior server, we will reroute as much flow as possible from other intersecting servers, such that it frees up some capacity at those servers. For a superior server, if we can reroute all flow from an inferior server that intersects it, we add this inferior server to the cluster led by it.

We now describe the stage in detail. Recall that any servers in $\overline{\sigma}$ is either superior or inferior. Also $\mathcal{S}'$ denotes the set of superior server and $\mathcal{I}'$ the set of inferior servers in $\overline{\sigma}$. Note

Algorithm 3 Clusting of Servers(CoS)

Input: A fraction solution from Alg.2 $\overline{\sigma}$
Output: A fraction solution $\widehat{\sigma} = (\widehat{x}, \widehat{y})$, a set of clusters $\mathcal{C}$.

1: Initialize $\widehat{x} \leftarrow \overline{x}$, $\widehat{y} \leftarrow \overline{y}$, $\mathcal{O} \leftarrow \emptyset$.
2: $\mathcal{S}' \leftarrow \{a_i | \forall a_i \in A \text{ and } \widehat{y}_i = 1\}$.
3: $\mathcal{I}' \leftarrow \{a_i | \forall a_i \in A \text{ and } 0 < \widehat{y}_i \leq \alpha\}$.
4: $C_i \leftarrow \{a_i\}, \forall a_i \in \mathcal{S}'$; $\mathcal{C} \leftarrow \{C_i | \forall a_i \in \mathcal{S}'\}$
5: **while** $\mathcal{I}' \neq \emptyset$ **do**
6: **for all** $a_i \in \mathcal{S}'$, $a_t \in \mathcal{I}'$ **do**
7: **if** a_t intersects a_i and $RB_i \geq \sum_{u_j \in U} \widehat{x}_{tj} BR_j$ **then**
8: Reroute the flow from a_t to a_i.
9: $C_i \leftarrow C_i \cup \{a_t\}$; $\mathcal{I}' \leftarrow \mathcal{I}' \backslash \{a_t\}$.
10: **end if**
11: **end for**
12: **if** $\mathcal{I}' \neq \emptyset$ **then**
13: $A_i \leftarrow \{u_j | \widehat{x}_{ij} > 0, \forall u_j \in U\}$, $\forall a_i \in \mathcal{I}'$.
14: $k_i \leftarrow \min\left\{BW, \sum_{u_j \in A_i} BR_j\right\}$.
15: $a_t \leftarrow \arg\max_{a_i \in \mathcal{I}'} k_i$.
16: $\mathcal{O} \leftarrow \mathcal{O} \cup \{a_t\}$; $\mathcal{I}' \leftarrow \mathcal{I}' \backslash \{a_t\}$.
17: **if** $k_t \equiv \sum_{u_j \in A_t} BR_t \leq BW$ **then**
18: Reroute the flow of all $u_j \in A_i$ from $A \backslash \mathcal{O}$ to a_t.
19: **end if**
20: **if** $k_t \equiv BW < \sum_{u_j \in A_t} BR_t$ and $k_t \geq 2BR_t^{\min}$ **then**
21: **while** $\exists u_j \in A_t : (1 - x_{tj}) BR_j \leq RB_t$ **do**
22: $u_k \leftarrow \arg\min_{u_j \in A_t} [RB_t - (1 - x_{tj}) BR_j]$.
23: Reroute the flow of u_k from $A \backslash \mathcal{O}$ to a_t.
24: **end while**
25: **end if**
26: **if** $k_t \equiv BW < \sum_{u_j \in A_t} BR_t$ and $BR_t^{\min} \leq k_t < 2BR_t^{\min}$ **then**
27: $u_k \leftarrow \arg\max_{u_j \in A_t : BR_j \leq BW_t} BR_j$.
28: Reroute the flow of u_k from $\mathcal{I}'$ to a_t.
29: $f \leftarrow \sum_{a_i \in \mathcal{O}} \widehat{x}_{ik}$.
30: Reroute the $\min\{RB_t / BR_j, (1 - f)\}$ amount of flow from $\mathcal{C}$ to a_t.
31: **end if**
32: **end if**
33: **end while**
34: $\widehat{y}_i \leftarrow 1$, $C_i \leftarrow \{a_i\}$, $\mathcal{C} \leftarrow \mathcal{C} \cup C_i, \forall a_i \in \mathcal{O}$.
35: **return** $\widehat{\sigma}$, $\mathcal{C}$.

that any superior servers a_i may serve a user u_j which is at a distance $(1 + \varepsilon)r$ from a_i. Denote by C_i a cluster initialized to $\{a_i\}$, and $\mathcal{C}$ denote the set of clusters.

In the main **while** loop of Alg.3, each inferior server will be added to either a cluster corresponding to some superior server (lines 6-11), or a set $\mathcal{O}$ which will eventually be part of the final solution (lines 12-32). We first handle the inferior servers that intersect with a superior server separately. If $a_i \in \mathcal{S}'$ has enough remaining bandwidth RB_i $(= BW - \sum_{u_j \in U} (x_{ij} BR_j))$, then reroute the flow from the intersecting receivable inferior server a_t to a_i and add a_t to C_i. After the execution of this for loop, if $\mathcal{I}'$ is not empty, the remaining inferior servers either do not intersect any superior servers, or the intersecting superior servers that do not have enough capacity. In following process, after selecting an inferior server and reroute the flow from other servers to it, the remaining capacity of some superior servers may be released. So in a new main **while** iteration, a cluster may be added a new inferior server.

After the above process, if $\mathcal{I}'$ is not empty, we will choose an inferior server in it add to $\mathcal{O}$. For any server $a_i \in \mathcal{I}'$, let A_i denote the set of users currently being served by a_i. Let k_i denote the minimum of its capacity, and the sum requests of users that it currently serves. We select the server $a_t \in \mathcal{I}'$ with the maximum value of k_i, and add it into the set $\mathcal{O}$. We can distinguish three cases according to the size of k_t. For the three cases of k_t, we will reroute different amounts of flow to a_t.

- **Case(i)**: $k_t \equiv \sum_{u_j \in A_t} BR_t \leq BW$. It means a_t has sufficient capacity to meet the requirements of all users it serves. We reroute the flow of all $u_j \in A_i$ from $A \backslash \mathcal{O}$ to a_t. In particular, for each $a_i \in \mathcal{S}'$, RB_i increases by $\sum_{u_l : a_t \text{ serves } u_l} \widehat{x}_{tl}$.
- **Case(ii)**: $k_t \equiv BW < \sum_{u_j \in A_t} BR_t$ and $k_t \geq 2BR_t^{\min}$, where $BR_t^{\min} = \min_{u_j \in A_t} BR_j$. It means a_t can serve at least one user. Observe that the flow out of server a_t is $\sum_{u_j \in A_t} x_{tj} BR_j \leq \alpha BW$; thus $RB_t \geq (1 - \alpha) BW = (1 - \alpha) k_t$.
  We use the capacity of a_t to reroute as much flow as possible from $A \backslash \mathcal{O}$. Note that we can reroute the flow of at least $\left\lfloor \frac{(1-\alpha)k_t}{BR_t^{\min}} \right\rfloor \geq \lfloor 2(1 - \alpha) \rfloor \geq 1$ users of A_t in this manner, since $BW \geq 2BR_t^{\min}$ and $\alpha \leq \frac{1}{2}$.
- **Case(iii)**: $k_t \equiv BW < \sum_{u_j \in A_t} BR_t$ and $BR_t^{\min} \leq k_t < 2BR_t^{\min}$. It means a_t can serve at most one user. Note that a_t has used $\sum_{u_j \in A_t} x_{tj} BR_j \leq \alpha BW$. We choose one user u_k with max bandwidth requirement not exceed BW, and reroute the flow of u_k from $\mathcal{I}'$ to a_t. In this step, we reroute at most α amount of flow because that

$$\begin{aligned} BR_k \sum_{a_i \in \mathcal{I}' : \widehat{x}_{ik} > 0} \widehat{x}_{ik} &\leq BR_k \sum_{a_i \in \mathcal{I}' : \widehat{x}_{ik} > 0} \widehat{y}_i \\ &\leq \alpha BR_k, \end{aligned}$$

  this follow the two inequalities (6a) and (10). Therefore, we have

$$\begin{aligned} RB_t &\geq BW - \alpha BW - \alpha BR_k \\ &\geq BW(1 - 2\alpha) \\ &\geq 0. \end{aligned}$$

  At last reroute the $\min\{RB_t / BR_k, (1 - f)\}$ amount of flow from $\mathcal{C}$ to a_t.

When the main **while** loop terminates, we have each inferior serer is either in $\mathcal{O}$ or clustered. We set $\widehat{y}_i \leftarrow 1$ for each server $a_i \in \mathcal{O}$, thus making it superior. For convenience, we also set cluster $C_i = \{a_i\}$ for each $a_i \in \mathcal{O}$. Note that, through out the algorithm we ensure that, if a user $u_j \in U$ is currently served by a server $a_i \in \mathcal{I}'$, then the amount of flow u_j receives from any serer $a_{i'} \in \mathcal{O} \cup \mathcal{I}'$ is the same as that in the preprocessed solution, i.e., the flow assignment of u_j w.r.t. the servers in $\mathcal{O} \cup \mathcal{I}'$ remains unchanged.

Algorithm 4 Selecting the Final Servers(SFS)

Input: A fraction solution from Alg.3 $\widehat{\sigma}$, a set of clusters $\mathcal{C}$.
Output: Final solution A^*.

1: Initialize $\widetilde{x} \leftarrow \widehat{x}$, $\widetilde{y} \leftarrow \widehat{y}$.
2: $\mathcal{S}'' \leftarrow \{a_i | \forall a_i \in A \text{ and } \widetilde{y}_i = 1\}$.
3: $\mathcal{I}'' \leftarrow \{a_i | \forall a_i \in A \text{ and } 0 < \widetilde{y}_i \leq \alpha\}$.
4: **for all** $C_h \in \mathcal{C}$ **do**
5: **if** $C_h \equiv \{a_h\}$ **then**
6: $\widetilde{y}_h \leftarrow 1$.
7: **else**
8: Cut the square containing C_h with side length $6r$ to grid cells, and each cell has side length $(r\varepsilon)/4$.
9: For each grid cell that contains at least one server, creating a group G consisting of all servers contained in the grid cell.
10: $\mathcal{G} \leftarrow$ all groups from C_h.
11: **for all** $G \in \mathcal{G}_2$ **do**
12: $a_m \leftarrow$ arbitrary server contained in G.
13: Reroute the flow from $G \backslash \{a_m\}$ to a_m.
14: $\widetilde{y}_m \leftarrow 1$, $\widetilde{y}_i \leftarrow 0, \forall a_i \in G \backslash a_m$.
15: **end for**
16: **end if**
17: **end for**
18: $A^* \leftarrow \{a_i | a_i \in A \text{ and } \widetilde{y}_i = 1\}$.
19: **return** A^*.

C. Selecting the Final Servers

In this subsection, we will select multiple servers per cluster in a way so that the selected servers when considered $SINR'_{\min}$ and p' can cover all the users in the cluster. This idea is similar to the ones in in Alg.2.

After Clustering of Servers, we can get the set of clusters $\mathcal{C}$ and a fraction solution $\widehat{\sigma}$. Let $\mathcal{S}''$ and $\mathcal{I}''$ denote, respectively, the set of superior and inferior servers after Clustering of Servers. We will consider each cluster $C_i \in \mathcal{C}$. If C_h contains only one server a_h, we choose a_h in the final solution $\widetilde{\sigma}$.

Otherwise, we will cut the square containing C_h with side length $6r$ to grid cells, and each cell has side length $\frac{r\varepsilon}{4}$. For each group $G \in \mathcal{G}$, we select a server a_m arbitrary in G as the leader server and reroute the flow from $G \backslash \{a_m\}$ to a_m. The selected server a_m can serve all users served by G with $SINR'$ and p'. In Alg.3, we reroute all flow from $a_i \in C_h \backslash \{a_h\}$ to a_h. The flow out of a_h is $\sum_{u_j \in a_h} \widehat{x}_{hj} BR_j \leq BW$. So the capacity of a_m can meet the requirements of users in G. Note that we will get $O(\varepsilon^{-2})$ cells, the number of groups is $O(\varepsilon^{-2})$. The number of servers we select from C_h is also $O(\varepsilon^{-2})$. Summing over all cluster, the number of servers selected is $O(\varepsilon^{-2} |\mathcal{S}''|) = O(\varepsilon^{-2})$.

D. Analysis of the Approximation Ratio

Lemma 2. *cost($\overline{\sigma}$) $= O(\varepsilon^{-2})\cdot$ cost(σ).*

Proof. Consider an iteration of Alg.2. At the beginning of the iteration, all servers in $\mathcal{I}_j$ were inferior, and $y(\mathcal{I}_j) > \alpha$.After the iteration,none of the servers in $\mathcal{I}_j$ are inferior. Furthermore, we select $O(\varepsilon^{-2})$ servers from $\mathcal{I}_j$ in the iteration. It follows that the entire DSIS algorithm increases the cost by a factor of at most $O(\varepsilon^{-2})$. □

Lemma 3. *Alg.3 CoS increases the cost of the solution $\overline{\sigma}$ only by a constant factor.*

Proof. Consider an iteration of Alg.2. At the beginning of the iteration, all servers in $\mathcal{I}_j$ were inferior, and $y(\mathcal{I}_j) > \alpha$.After the iteration,none of the servers in $\mathcal{I}_j$ are inferior. Furthermore, we select $O(\varepsilon^{-2})$ servers from $\mathcal{I}_j$ in the iteration. It follows that the entire DSIS algorithm increases the cost by a factor of at most $O(\varepsilon^{-2})$. □

Lemma 4. *cost($\widetilde{\sigma}$) $= O(\varepsilon^{-2})\cdot$ cost($\widehat{\sigma}$).*

Proof. Consider an iteration of Alg.4. We select $O(\varepsilon^{-2})$ servers from C_h in the iteration. It follows that the entire SFS algorithm increases the cost by a factor of at most $O(\varepsilon^{-2}|\mathcal{S}''|) = O(\varepsilon^{-2})$. □

Theorem 1. *There is an approximation algorithm, Alg.1, for MMCSC problem, which delivers a bi-criteria $(O(\varepsilon^{-4}), 1+\varepsilon)$-approximate solution, where ε defined in Eq.9. In addition, the time complexity of the algorithm is $O()$.*

proof (Approximation ratio): With the analysis of Lemma 2, Lemma 3 and Lemma 4, we have that

$$\begin{aligned} \text{cost}(\widetilde{\sigma}) &= O(\varepsilon^{-2}) \cdot \text{cost}(\widehat{\sigma}) \\ &\leq cO(\varepsilon^{-2}) \cdot \text{cost}(\overline{\sigma}) \\ &= O(\varepsilon^{-4}) \cdot \text{cost}(\sigma) \\ &\leq O(\varepsilon^{-4}) \cdot OPT. \end{aligned}$$

(Polynomial Running Time): We will analyze the complexity of Alg.2, Alg.3 and Alg.4 respectively.

In Alg.2, we have to determine whether n users have too many inferior servers to serve it (i.e. $\sum_{a_i \in \mathcal{I}: \overline{x}_{ij} > 0} \overline{y}_i > \alpha$). And we will take $O(m\varepsilon^{-2})$ time to construct $\mathcal{G}_j$. There are at most ε^{-2} groups in $\mathcal{G}_j$, and the complexity of reroute the flow from $\forall a_i \in G_l \backslash \{a_i\}$ to a_i is mn. So the complexity of lines 10-14 is $mn\varepsilon^{-2}$. Thus, the complexity of Alg.2 is $O(mn^2\varepsilon^{-2})$.

We can conclude that Alg.3 would terminate within m iterations theoretically. In each iteration, we first try to add the inferior servers into a cluster in $O(m^2 n)$. And the follow step is selection of inferior server. For the three cases, case ii spends the longest time, is mn^2. Then, the time complexity of Alg.3 is $O(m^3 n + m^2 n^2)$.

The outer loop of Alg.4 is performed at most m iterations, and the time complexity of each iteration is similar to that of each iteration of Alg.2, which is $O(mn\varepsilon^{-2})$. So the time complexity of Alg.4 is $O(m^2 n\varepsilon^{-2})$.

In summary, the time complexity of our proposed Grid-based three-step rounding approximation algorithm is $O(mn^2\varepsilon^{-2} + m^3 n + m^2 n^2 + m^2 n\varepsilon^{-2})$. ■

V. Pre-set location generation algorithm

The position of each UAV in A will have a direct impact on the result of Alg.1. Therefore how to get A is equally crucial. Due to the flexible mobility of UAVs, they can be deployed to any location in the region. But it is impractical to traverse every possible location. This subsection proposes a hierarchical pre-set location generation algorithm based on the graph center of gravity.

A. Barycenter-based hierarchical pre-set location generation algorithm

In this subsection, we propose a Barycenter-based hierarchical pre-set location generation algorithm that divides the user distribution area hierarchically by using a hierarchical idea similar to that in Section 3. For each square region of a hierarchy, the user located in the square region is constructed as a "mass" plate, and a preset point is generated at the center of gravity of the plate.

B. Neighborhoods pre-set location generation algorithm

In this subsection, we propose a Neighborhoods pre-set location generation algorithm. if the signal transmitting power of the UAV is fixed, then for each user, there is at least one UAV that is no more than Euclidean distance away from it d. We can accordingly draw for each user a plane of height h in the circle, and the intersection of all circles in that plane is the preset point of the UAV.

VI. Experimental Results

VII. Conclusion and future work

We feel that the local assignment algorithm in Section will find application in related areas.

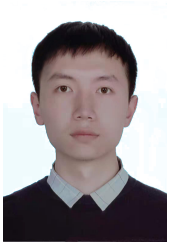
Acknowledgment

This work was supported in part by the National Natural Science Foundation of China [Nos. 12071417, 61762091, and 62062065] and the 12th Postgraduate Innovation Project of Yunnan University [No. 2020294].

References

[1] S. Bandyapadhyay, S. Bhowmick, T. Inamdar, and K. Varadarajan, "Capacitated covering problems in geometric spaces," *Discrete & Computational Geometry*, vol. 63, no. 4, pp. 768–798.

[2] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP Altitude for Maximum Coverage," *IEEE Wireless Communications Letters*, vol. 3, pp. 569–572, Dec. 2014.



Qinghui Zhang received a B.E. degree in software engineering from YangZhou University in 2020. He is currently working towards an M.S. degree at Yunnan University. His research interests include cloud computing and edge computing.

Weidong Li received a Ph.D. degree from the Department of Mathematics of Yunnan University in 2010. He is currently a professor at Yunnan University. His main research interests are combinatorial optimization, approximation algorithms, randomized algorithms and cloud computing.

Qian Su received a master's degree in computer science from Yunnan University in 2009. She is a lecturer and a Ph.D. candidate in the School of Information Science and Engineering, Yunnan University. Her research interests include edge computing and cloud-edge collaboration.

Xuejie Zhang received a master's degree in computer science and engineering from Harbin Institute of Technology in 1990. He received a Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong in 1998. He is currently a professor in the Department of Computer Science and Engineering of Yunnan University. His main research interests are high-performance computing, distributed systems, computer networks and cloud computing.