1 Dynamic allocation

In this paper, we propose an efficient method for managing the dynamic allocation of UAVs to fire locations that spread with time. The allocation process considers two paramaters, fire density and proximity of the UAVs to fire locations. Regions with higher fire densities correspond to areas with a higher number of fires concentrated in a region. To partition the map into regions according to their fire densities, the first step in our allocation process is to carry out the K-means *clustering algorithm* via the Matlab built-in function, *kmeans*. We set this function to use the Manhattan distance as the evaluation index of similarity in order to group fires with similar locations into the same cluster. The *kmeans* function requires a desired number of clusters, k, as input.

To find the optimal k_{opt}, the elbow method is used. Similar work on utilizing the elbow method and other clustering techniques is shown in [X]. In the elbow method, the process iterates through the possible number of clusters, k, from 2 to some maximum number of clusters, K_{max}, and the sum of square error (SEE) for each k is computed. Ideally, we are looking for a value of k that results in clusters that have a low SEE. This would lead to fires with high location similarity being assigned to the same cluster. The optimal number of clusters is the value of k at which the second derivative of the average of $SEE(k)$ is maximized. The idea is that the marginal drop in the average SEE as k increases will decrease dramatically for some value of k before it reaches a plateau, hence the “elbow criterion”. The partition algorithm is summarized in Algorithm I.

Algorithm 1 Partition algorithm

```
1: procedure CLUSTERING($K_{max}, fireLocs$)
2:     Set $k = 2$
3:     while $k < K_{max}$ do
4:         Compute clusters set $C = kmeans(fireLocs, k)$
5:         Compute mean sum of square error with k clusters, $SEE_{avg}(k)$
6:     Set optimal $k$, $k_{opt} = k$ that maximizes the second derivative of $SEE_{avg}(k)$
7:     Generate initial cluster centroid positions $C_{init}$
8:     Compute clusters set $C = kmeans(fireLocs, k_{opt}, C_{init})$
9:     Assign $fireLocs$ to their corresponding cluster $c \in C$,
10:    return $C$, fire location assignments
```

Once the map is divided into clusters, the number of UAVs that will be allocated to each cluster is defined by (),

$$N_{alloc}(c) = \lceil \rho_c/\rho_{tot} \times (N_{tot} - N_{free}) \rceil, \tag{1}$$

where ρ_c is the cluster priority calculated as the sum of intensity values of fires that belong to cluster c, ρ_{tot} is the sum of intensity values of all fires. N_{tot} is the total number of UAVs and N_{free} is the number of UAVs that are not assigned to a cluster. For each cluster c, we assign a subset of UAVs chosen from all available UAVs. This subset of size N_{alloc} is comprised of those UAVs with locations closest to c. Once each cluster c has an assigned subset of UAVs, the next step is to decide the fire location belonging to c for each UAV to set as its goal. We have two modes in which UAVs can be assigned to fires, *Sync* mode and *NonSync* mode.

In *Sync* mode, for each cluster c, we loop through each UAV from the assigned subset to c and determine the fire with the minimum cost, proportional to its distance from the UAV, and highest priority, proportional to the fire intensity, according to function g as shown in (). The rest of the UAVs in the loop are assigned to the same fire as the first UAV in the loop.

$$\min_{f \in F} g(f,x) = d_f * \|x - f\| + s_f * \|g - f\| - n_f * intensity_f, \tag{2}$$

where d_f is a coefficient between 0.0 and 1.0 and is used as an importance weight for the distance between UAV location x and fire $f \in F$. F represents the set of all fire locations in cluster c. The coefficient n_f represents an importance weight for the fire intensity level, $intensity_f$. This coefficient is set as 1.0 - d_f. Finally, coefficient s_f corresponds to the switching penalty of changing an already assigned fire to a different fire, if the assigned fire has not been reached yet. If the UAV reaches its assigned fire, then the minimization is done over all other fires.

In *NonSync* mode, for each cluster c, we also loop through each UAV from the assigned subset, but only one UAV is assigned to a single fire. The fire is chosen from the set of unassigned fires for each UAV according to (). Each time a fire is assigned, it is removed from the set F.