The Case for Admission Control of Mobile Cameras Into the Live Video Analytics Pipeline

Francescomaria Faticanti
Fondazione Bruno Kessler

Francesco Bronzino
Université Savoie Mont Blanc

Francesco De Pellegrini
Avignon University

ABSTRACT

In this paper we consider the problem of orchestrating video analytics applications over an edge computing infrastructure. Video analytics applications have been traditionally associated to the processing of video streams generated by fixed video cameras. Nowadays, however, the availability of mobile video cameras has become pervasive. We argue that to take advantage of the presence of mobile video cameras—and their informative content—it may be necessary to refactor the edge orchestration logic. We propose a new solution that splits the problem into two connected actions: 1) Placement of processing functions in the infrastructure and 2) Admission of most informative cameras based on their field of view. We hence describe a possible scheme for joint video stream admission and orchestration. Finally, preliminary numerical results are presented, demonstrating that separating the two logic components can improve coverage while reducing the cost of deployment.

CCS CONCEPTS

• **Networks** → *Network control algorithms*; • **Information systems** → *Multimedia streaming*.

KEYWORDS

Video Analytics, Mobile Cameras, Resource Allocation, Admission Control, Edge Computing

ACM Reference Format:
Francescomaria Faticanti, Francesco Bronzino, and Francesco De Pellegrini. 2022. The Case for Admission Control of Mobile Cameras Into the Live Video Analytics Pipeline. In *3rd ACM Workshop on Hot Topics in Video Analytics and Intelligent Edges (HotEdgeVideo '21), January 31–February 4, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3477083.3480151

1 INTRODUCTION

Live video analytics are a key application that combines AI techniques and modern microservice architectures to gather information from video camera feeds. Video analytics applications process several video inputs in a tagged geographical area to perform sophisticated functionalities such as motion detection (e.g., count moving vehicles, identify an accident or a traffic jam) or features extraction (e.g., recognize a target vehicle's plate number). Performing these operations conventionally requires executing multiple tasks in an ordered sequence (e.g., decoding, background extraction, object detection, plate extraction, vehicle counting) [4]. However, the sheer size of data generated by camera flows makes it infeasible to centralize the processing, so that the decentralized edge processing approach is key for their success. As for most AI-based applications, video analytics performance—i.e., their accuracy and time efficiency—depends loosely on throughput and computing resources. It is rather conditioned by the employed functions (e.g., lightweight vs heavyweight shape classifiers) as well as the informative content of video frames (e.g., crowded scenes require better models to extract available information). Each application is composed of a sequence, or chain, of functions that manipulate the incoming images using AI functions. Performance of such models can be impacted by a number of quantifiable factors (e.g., resolution, frame rate, etc.) as well as by less deterministic ones (e.g., how many objects are present in a given frame) [5].

In the standard scenario considered in the literature, video streams are generated by fixed cameras, and the resulting problem becomes a, somewhat, standard problem of orchestration of microservice chains [6]. Unfortunately, solely focusing on static video flows can limit the applications chance of maximizing their accuracy: if subjects exit the cameras' field of view, no model is able to recover the missing information. In this paper, we consider instead how to integrate mobile video streams, e.g., generated by cameras mounted on vehicles or cameras of pedestrians smartphones crossing a tagged area. Depending on the application, a mobile video stream may complement fixed cameras video streams with additional informative content. Hence, considering mobile cameras improves the basic coverage brought by fixed cameras and decreases the probability of having blind spots. In vehicles' surveillance for safety critical events, for example, improving the coverage can play a crucial role in the prevention of car accidents.

However, due to their less predictable behavior, new challenges lie ahead when mobile cameras are integrated in the architecture. To be capable of accounting for the effects of mobility, the orchestration logic deployed must be adapted. In fact, a static placement of application modules which ignores the presence of mobile video cameras may greedily saturate local edge resources and may thus prevent the association of new incoming mobile video streams later on. In order to match effectively the dynamic changes of available sources in the infrastructure, the straightforward solution is to reallocate periodically the application chains over available resources. Unfortunately, frequently re-allocating modules on the infrastructure generates delay problems due to time overhead incurred in the migration of modules across different locations.

The proposed approach aims to acquire mobile video streams by means of pre-allocated application pipelines, leaving re-configuring

HotEdgeVideo '21, January 31–February 4, 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-8700-2/22/01…$15.00
https://doi.org/10.1145/3477083.3480151

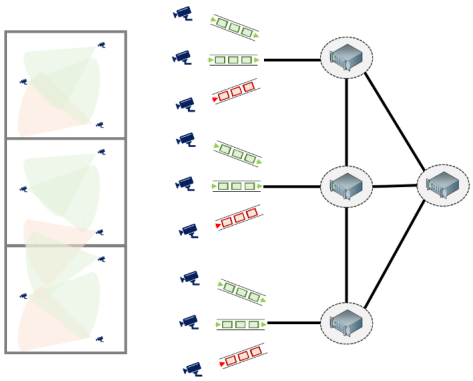

Figure 1: Typical scenario where cameras in each area are associated to an entry point.

the chain placement as a last resort. To do so, we observe that not all video streams should be admitted, but rather only the ones that bring the larger amount of information. To achieve this, we can integrate the standard placement problem with an admission control phase. In this work we describe the resulting combined admission and orchestration problem, where a batch of video analytics applications are instantiated in a heterogeneous edge/cloud environment and a set of fixed and mobile cameras cover the ensemble of target areas.

In the rest of the paper we present our solution for support the dynamic admission of mobile cameras in the processing pipeline. We first present in Section 2 the orchestration problem of live video analytics and their applications. Next, Section 3 presents the two steps solution of placement and admission control that aims to more easily support the integration of mobile cameras into the analytics pipeline. We present in Section 4 our preliminary simulation results that show how even simple admission control strategies can improve the covered observed area for applications over a baseline, while reducing the deployment cost that would be incurred if continuously performing modules placement in the infrastructure. Finally, Section 5 concludes the paper.

2 PROBLEM DESCRIPTION

Here we describe our reference orchestration problem for video analytics. We consider a batch of applications deployed in a heterogeneous system comprising of a certain number of areas, a set of available edge nodes and a set of fixed cameras. Furthermore, we assume that a set of mobile cameras can migrate across the areas as shown in Figure 1. The cameras in each area collect video flows sourcing from fixed cameras installed locally as well as from mobile cameras traversing the tagged area. We assume that each flow feeds a specific video analytics application hosted on the infrastructure.

Applications. The structure of each video application forms a pipeline graph where each node represents an application specific function. An edge between two nodes of the pipeline exists if their functions are related, e.g., the output data of a function upstream feeds the next node downstream. Figure 2a) shows sample representation for such video analytics application model, comprising a chain of four different functions. In practical realisations, it is possible that each application function is provided with different implementations, namely *knobs*. In fact, each implementation guarantees a different degree of accuracy for the application and it engages different resource levels, namely CPU, memory, etc, on the hosting nodes. For example, as shown in [4], a video analytics application, e.g., an application for object tracking, can consist of functions like an object detector and an associator component. Each function presents different implementations, e.g., as shown in [6], an object detector can be implemented with different detection models (Yolo, VGG or AlexNet).

Performance Metrics. In this work we consider two performance metrics for video analytics applications: i) the *coverage* brought by a camera, i.e., the amount of space, for example a specific subarea or a specific field of view [9], that the camera can cover in its direction; ii) the *accuracy* of each application in processing new information brought by the camera, e.g., the amount of objects the application can count from the images coming from the camera. In this paper we focus on the maximization of the coverage metric. The choice of the coverage as the objective function is motivated by the improvement in accuracy reached by an application as the information brought by a set of cameras increases [9]. In future works we are going to study a more precise relation between coverage and accuracy.

Infrastructure. The infrastructure consists of a fixed number of cameras per area connected to a set of heterogeneous edge nodes where applications' functions are deployed. Nodes provide different levels of computational resources. In each area a set of cameras sends several video flows to the applications deployed on the infrastructure providing a certain amount of coverage of the area. We assume that fixed cameras always provide the same coverage over time. On the other hand, mobile cameras can change their position and, consequently, the coverage offered to the applications.

Orchestration Problem. The problem we address in this paper is the determination of an optimal allocation for applications' functions over the infrastructure as well an optimal admission policy for video flows in order to maximize the coverage brought to applications guaranteeing a minimum desired level of accuracy per application. Such a solution should be able to support different types of cameras, i.e., both mobile and fixed ones. The support for mobile cameras poses additional complexity in the decision process. In fact, mobile cameras can change continuously their access point thus modifying the distance from the first computing node available, i.e., the node where the first application's function has been deployed.

3 PROPOSED SOLUTION

Several solutions regarding video analytics applications orchestration have been proposed in the literature [4–6]. However, most of these solutions only deal with video streams coming from fixed cameras. Nowadays, with the significant spreading of mobile cameras, there is a need of including this kind of cameras in the video analytics operations. The introduction of mobile cameras brings some benefits in terms of accuracy for the applications. Indeed, they can cover some parts of a specific area, namely *blind spots*, that would not be possible to reach just using fixed cameras.

In our context, each access point of each area receives video streams coming from either fixed or mobile cameras. These streams

are processed by the applications already deployed on the infrastructure with limited resources on edge nodes. Hence, a selection of flows to be admitted should be performed in order to prevent the saturation of resources at local nodes. In the admission control process we must factor in the contribution of a tagged video stream to the overall coverage of its target application. The next sections describe in more detail the aforementioned procedure.

In order to solve the problem described above, we propose a two-step solution: first an application placement step, followed by an admission control step.

1) *Applications Placement Step.* This solution step entails the resource allocation in order to accommodate the applications pipelines onto the infrastructure. The resource allocation methods should take into account the computational requirements of each application's function and the capacity of nodes where such functions are instantiated. Furthermore, such placement methods should consider networking constraints in terms of available bandwidth.

2) *Admission Control.* This step receives as input the placement performed in the previous step. The admission control policy is designed in order to decide which video stream should be admitted during the processing phase. In a dynamic scenario, the need for such an admission step comes from the presence of mobile cameras that can improve over time the coverage provided to the deployed applications. Furthermore, processing of several video streams can improve the accuracy but it can quickly lead to resource saturation. Admission control should prevent this problem performing a selection on the video streams to be processed based on the coverage they are bringing to applications. Performing such an admission control only slightly penalises accuracy but prevents resource saturation. Finally, an admission control policy brings out the study of the trade-off between the amount of admitted cameras, and hence the resource occupation, and the final accuracy reached by applications.

3.1 Applications Placement

The placement step consists of choosing for each application's function a certain implementation, i.e., a knob, and mapping it to a target node of the network. This kind of mapping also requires as well to identify the edges between two consecutive functions of an application; i.e., the physical path in the infrastructure network. The applications placement involves the resource allocation for the application pipelines on the infrastructure. Each application module may have different implementations (knobs).

This placement problem can be polynomially reduced to the well known Virtual Network Embedding (VNE) problem [3] (it is sufficient to consider the variant where a unique knob exists per application module). Thus, since the VNE problem is known to be NP-hard, the placement step involves the solution of an NP-hard problem; furthermore, since negative results exists in the literature for the approximation of VNE problem [1]; indeed, such negative results discourage seeking for approximation algorithms. Given the computational complexity of the VNE problem, we adopt a lightweight heuristics consisting of splitting further the problem in two sub-problems: i) *node mapping* and ii) *edge mapping*.

For the first sub-problem it is worth noting that different applications pipelines can share the same set of modules deployed on

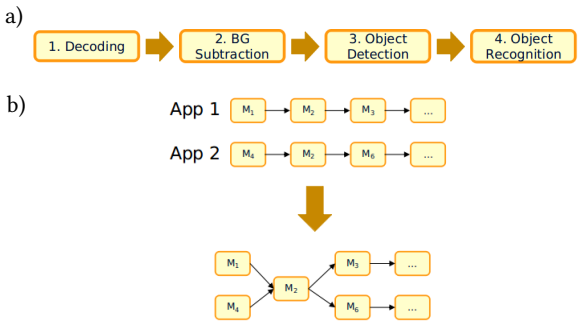

Figure 2: a) Example of video analytics application represented as a chain of dependent modules; b) Example of module sharing among application pipelines.

the infrastructure as shown in Figure 2b). This corresponds to the consolidated technique of merging application functions, which is employed routinely to save computational and networking resources [4]. We adopt this technique to create a unique directed graph by merging all the common functions each pair of applications. The implementation chosen for each function is the one that maximise the accuracy for each application. I.e., for each function, we choose the implementation that leads to the maximum accuracy for all the applications sharing that function. This is the *golden trace* concept used in [6].

Given the aggregated graph obtained by merging the common functions, namely the *aggregated graph*, we perform the functions mapping minimizing the number of edge servers used. The objective is to avoid the distribution of the application among the infrastructure minimizing the bandwidth consumption and to maximally exploit the edge resources without exceeding nodes' resources. Minimizing the number of servers used for the functions mapping can be formulated as a standard Bin Packing problem [7]. In this case, though, despite such problem is again one of the NP-hard family, there exist different heuristic methods which offer approximation bounds in order to solve the problem [2].

The second sub-problem, i.e., the edge mapping, can be readily solved after the node mapping step. To this aim it is sufficient to compute the less congested path between each couple of edge nodes where two adjacent functions are placed, i.e., in the aggregated graph.

In summary, the applications placement involves the following consecutive steps:

(1) Creation of an aggregated graph by using the merging techniques between applications pipelines and adopting the golden trace in order to choose the implementation (knob) for each function. (2) Mapping of functions and edges of the total graph onto the network's servers and paths, respectively. The functions mapping is performed by minimizing the number of servers by the First-Fit algorithm [2].

At the end of the placement step we obtain a graph of application modules mapped to the infrastructure graph, as shown in Figure 3.

3.2 Admission Control

The admission control phase aims to select, for each area, the set of video streams to be admitted at the "entry point" of that area. Video

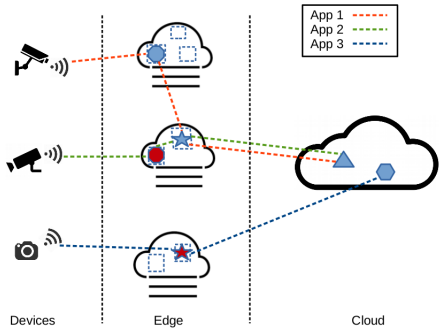

Figure 3: Example of applications deployment among an infrastructure consisting of an edge and a cloud side

streams can come either from fixed or mobile cameras. However, the reward function driving the admission is composed by the coverage brought by each flow and the match between the information contained in the flow and the applications receiving the information by the flow. Once a given flow has been admitted for a specific application, the subgraph of the total graph, computed in the previous step, related to the application is activated for processing the information brought by the flow. Note that the cost function for this operation is determined by the placement and the configuration selected in the previous step for the application. Hence, the cost related to the residual bandwidth on the paths followed by the flow among the pipeline's locations and the computational load is known and determined by the placement computed in the first step.

The main objective of the admission control step is to select flows that maximize coverage matching the information desired by the application, and allow to stay below the total amount of computational and networking resources usage.

3.3 Coverage and Application Matching

As shown in Figure 1, each camera covers a given portion of a specific area. The relationship between the coverage percentage of a camera and the accuracy of the application depends on the information carried by the flow sourcing from the tagged camera. For this reason we can say that there should be a match between the information carried by the flow and what the application needs. This can be translated into a difference between the coverage of the camera and the coverage desired by the application. The coverage can be modeled, e.g., as a portion of the target area or as an angle covered by the camera. Furthermore, we assume that the coverage of the fixed cameras never changes in time, whilst mobile cameras can add new information, i.e., by covering blind spots of fixed cameras.

In summary, the admission control selects, for each area, the set of video streams that maximises the total coverage desired by the applications receiving the stream. The admission control must take into account the cost function given by the resource allocation, i.e., bandwidth, memory and CPU consumption. Future works will investigate more precise methods to model the admission control description and resolution.

4 NUMERICAL EVALUATION

In this section we present some preliminary evaluations of our solution. Our goal is twofold: i) confirm that admission control is a promising tool to handle the mobility of cameras and can significantly improve the coverage provided to the applications; ii) provide some insight into the trade-off which rules *instantiation costs* and the churn-rate of application functions placed onto the infrastructure. The instantiation costs are related to the number of functions should be moved when a new applications placement is computed.

4.1 Simulation settings

The main setting of the numerical simulations is described in the following.

Network Infrastructure. The network infrastructure we consider is a playground composed by four areas with one edge server per area. Two additional edge servers are connected to the others associated to the remaining areas. For the sake of simplicity, the connectivity among edge servers is represented by a complete network graph where each edge of the graph has an available bandwidth of 60 Mbps.

Cameras and Mobility Model. The number of fixed cameras is set to 8, with two fixed cameras per region, and a total of 10 mobile cameras. At the beginning of each simulation run, each mobile camera is associated to a certain area at random. At each iteration of the simulation mobile cameras can change the area according to a random walk over the set of areas [8]. Each camera has a fixed frame-rate and a coverage for the associated area. The entire area corresponds to a grid of which the app coverage represents a subset.

Applications. Each application is built by picking certain number of functions from a set of 15 different ones. Each function has a maximum of 5 possible implementations with different weights for the computation of the total accuracy of applications [4]. Furthermore, each application has, for each area of the network, a desired subarea from which it should receive images; such subarea is mapped to a specific subset of the area grid.

4.2 Numerical Results

The numerical results have been obtained using a Python-based simulator. Each data point in the graphs represents the average value over 30 instances where the infrastructure is fixed and instances are randomized both in the desired coverage of applications and in the initial distribution of mobile cameras.

The proposed solution, namely `AC`, is compared with two baseline approaches in terms of coverage:
`No-AC`: unlike `AC`, all the video streams – from both fixed and mobile cameras – are admitted as long as the network resources are not saturated and without considering the coverage that each camera brings to its reference application.
`fixedCameras`: it performs the same operations of `No-AC` but it does so only on fixed cameras, i.e., it neglects the potential contribution of mobile cameras.

Furthermore, in order to understand how the placement of applications functions impacts the gained coverage, we compared two different approaches. The first one, called *reactive placement*, performs a new placement of functions only when there is not

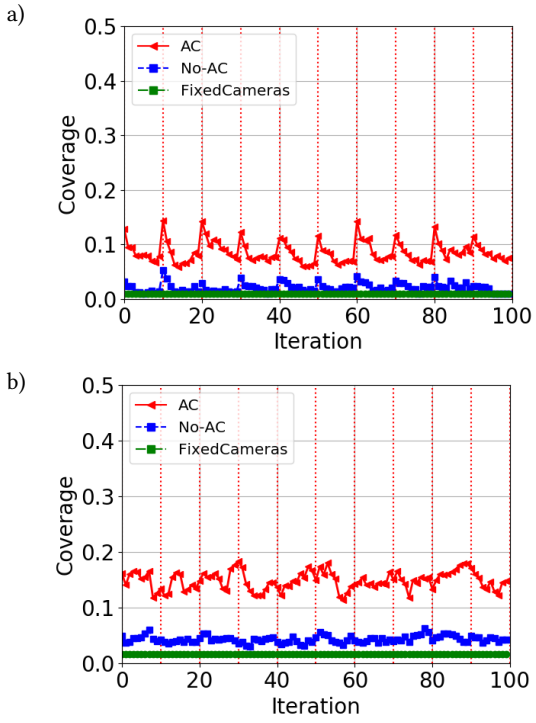

Figure 4: Average coverage: a) Reactive b) Proactive replacement

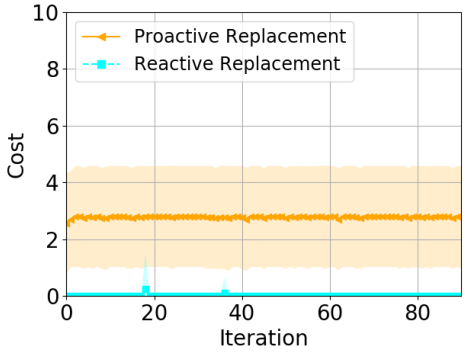

Figure 5: Average number of migrated functions.

enough bandwidth to admit at least one stream per application (i.e., not even the fixed camera from the baseline scenario can be admitted). The second approach, called *proactive placement*, performs a rearrangement of the applications functions at each iteration. The heuristic replacement procedure we implemented in this case migrates applications functions from the most loaded servers to the ones that are attached to the area with highest concentration of cameras. In both the approaches the replacement of functions is performed according the placement step described in Section 3 prioritising less loaded servers for the placement of applications' functions.

In Figure 4a) we evaluate the average coverage offered by the three procedures. The coverage is computed as the average over all the applications of *applications' ratios*. For each application, the *application's ratio* is computed as the fraction of the total area desired by the application actually covered by the admitted cameras. Admission control events occur periodically every ten iterations – as highlighted by red vertical lines: at those events video flows admitted per area are selected. As expected, the `fixedCameras` have constant and lower coverage given the sole consideration of fixed cameras. `No-AC` does not take into account the coverage in the admission of cameras leading to a low coverage in average. Our proposed solution shows significant performance gains in terms of coverage, confirming that a scheme based on the cascade of placement and admission control of mobile cameras attains a significant improvement of the coverage with respect to the baselines.

Similarly, Figure 4b) shows the average coverage of the three approaches when a replacement of applications functions is proactive and performed at each iteration. The difference between the three approaches is similar to the one noticed in Figure 4 but it is worth noting here that that there is an increment in the average coverage with respect the case where the replacement is performed in reactive fashion.

A continuous reconfiguration of the applications functions across the infrastructure can be beneficial from the coverage's point of view but there exists an associated cost. Figure 5 reports on the average cost of proactive and reactive replacement. The cost is measured as the number of functions moved from two consecutive configurations of applications functions placement. The number of migrated functions greatly impacts the system's availability since function migration involves several additional delays such as, e.g., the image download time and other delays related to data transfer. Overall, all these operations can cause a significant interruption of the applications operation. Consequently, a frequent function replacement improves indeed coverage but, on the other hand, may lead to a considerable cost for the applications placement reconfiguration.

5 CONCLUSIONS

We have considered in this paper the problem of admitting video streams for video analytics applications when fixed and mobile cameras coexist. We argue that an admission control policy for video streams coupled to the application chains placement – under capacity constraints on computational and networking resources – can significantly improve the performance figures of applications' coverage. Furthermore, we have investigated the trade-off between the reallocation churn-rate of applications functions and the cost in terms of the number of functions which need to be migrated across the infrastructure at each reallocation event. The take-away message is that while mobile cameras bring more information to video applications, admitting all available video streams can quickly saturate available resources. Thus, admission control helps applications to select video streams that significantly improve the coverage without exceeding the capacity of available resources. In future works we plan to precise the relation between the coverage and the accuracy in processing the information brought by selected cameras. Furthermore, we are going to study how to tune efficiently proactive replacement rates to mitigate the undesirable additional delays it may induce in the applications' operations.

REFERENCES

[1] Edoardo Amaldi, Stefano Coniglio, Arie MCA Koster, and Martin Tieves. 2016. On the computational complexity of the virtual network embedding problem. *Electronic Notes in Discrete Mathematics* 52 (2016), 213–220.

[2] György Dósa. 2007. The tight bound of first fit decreasing bin-packing algorithm is FFD (I)âĽď 11/9OPT (I)+ 6/9. In *International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Springer, 1–11.

[3] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann De Meer, and Xavier Hesselbach. 2013. Virtual network embedding: A survey. *IEEE Communications Surveys & Tutorials* 15, 4 (2013), 1888–1906.

[4] Chien-Chun Hung, Ganesh Ananthanarayanan, Peter Bodik, Leana Golubchik, Minlan Yu, Paramvir Bahl, and Matthai Philipose. 2018. Videoedge: Processing camera streams using hierarchical clusters. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 115–131.

[5] Samvit Jain, Xun Zhang, Yuhao Zhou, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Paramvir Bahl, and Joseph Gonzalez. 2020. Spatula: Efficient cross-camera video analytics on large camera networks. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 110–124.

[6] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. 253–266.

[7] Silvano Martello and Paolo Toth. 1990. Bin-packing problem. *Knapsack problems: Algorithms and computer implementations* (1990), 221–245.

[8] Andrè Panisson. 2012. pymobility. https://github.com/panisson/pymobility

[9] Enes Yildiz, Kemal Akkaya, Esra Sisikoglu, and Mustafa Y Sir. 2013. Optimal camera placement for providing angular coverage in wireless video sensor networks. *IEEE transactions on computers* 63, 7 (2013), 1812–1825.