PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Workload Optimization for Swarm-Powered Ad-hoc Clouds
Auteur: Author:	Leandro Rochink Costa
Date:	2021
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Rochink Costa, L. (2021). Workload Optimization for Swarm-Powered Ad-hoc Clouds [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/6650/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/6650/
Directeurs de recherche: Advisors:	Andrea Lodi, Daniel Aloise, & Luca Giovanni Gianoli
Programme: Program:	Doctorat en mathématiques

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Workload optimization for swarm-powered ad-hoc clouds

LEANDRO ROCHINK COSTA
Département de mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Mathématiques

Juin 2021

© Leandro Rochink Costa, 2021.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Workload optimization for swarm-powered ad-hoc clouds

présentée par **Leandro ROCHINK COSTA**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Nadia LAHRICHI, présidente
Andrea LODI, membre et directeur de recherche
Daniel ALOISE, membre et codirecteur de recherche
Luca Giovanni GIANOLI, membre et codirecteur de recherche
Alejandro QUINTERO, membre
Brunilde SANSO, membre
Ilario FILIPPINI, membre externe

DEDICATION

To my family...

ACKNOWLEDGEMENTS

The accomplishment of this thesis wouldn't be possible without the support of numerous people whose help had directly or indirectly contributed during my Ph.D. pursuit. For them, I must give the most sincere gratitude.

My research advisers, professors Andrea Lodi, Daniel Aloise, and Luca G. Gianoli, I would like to give my deepest thanks for giving me the opportunity to pursue the Ph.D. at Polytechnique Montréal and for the many hours of guidance, teaching, and support. Your knowledge and mentoring were crucial, and you inspired me to be a better researcher. I feel extremely fortunate to have worked with you during my Ph.D.

This thesis wouldn't be possible without the support and hard work of all staff from GERAD and the Canada Excellence Research Chair in Data Science for Real-Time Decision-Making, especially, Khalid Laaziri, Koladé Nourou, Mariia Kopach, and Mehdi Taobane. The staff from Humanitas Solutions equally played an important part in this thesis, in particular, Abdo Shabah, Abdel R. Barghout, Luca Bianchi, Maxime Colin, and Vincent Boyavalle.

The completion of this thesis would also be impossible without the professors Daniel Aloise and Caroline Thennecy de Medeiros Rocha. Thanks for introducing me to the operations research field, for mentoring me, and for offering my first opportunity as an undergraduate research intern in 2010. You two have been a source of inspiration since then.

I would also like to thank the professors Nadia Lahrichi, Alejandro Quintero, Brunilde Sanso, and Ilario Filippini for accepting being part of the jury of my thesis.

I'm also very thankful for all the great friends that I have made in Montreal. My officemates, Federico and Jiaqi, thanks for all daily conversations and laughs. For sure, I cannot forget all the Brazilians (and the Brazilians in the heart) friends whose presence had made this journey joyful and closer to Brazil. To my friends in Brazil, I also extend my gratitude for all these many years of friendship.

Lastly, it's impossible to express the gratitude that I have for my parents, Eloise and Luis, as for my siblings, Wagner, Kari, and Larissa, for being essential in my journey as a human being.

RÉSUMÉ

Les applications de détection modernes peuvent être décomposées en phases de détection et de calcul, où les données sont généralement collectées par un système local dédié de l'internet des objets dans la phase de détection pour être traitées dans la phase de calcul. Compte tenu de la quantité massive de données présentes dans ces applications, l'étape de calcul peut nécessiter des ressources de stockage et de calcul robustes dans le nuage. Cependant, les infrastructures communes de l'informatique en nuage présentent des problèmes (par exemple, une latence élevée) qui sont inadmissibles pour les applications limitées dans le temps comme les interventions d'urgence et les opérations de secours en cas de catastrophe. Les paradigmes de l'informatique en périphérie de réseau et du nuage ad-hoc traitent ces problèmes en fournissant des ressources de stockage et de calcul à proximité de leurs utilisateurs, permettant ainsi aux applications de conserver leurs phases de calcul locales. Par conséquent, ces paradigmes ont été largement utilisés dans des applications avec des restrictions de temps et de réseau. Par exemple, un essaim de drones collectant des photos et des vidéos d'une zone post-catastrophe peut créer une infrastructure de nuage ad-hoc sans fil pour traiter les données collectées indépendamment de la connexion Internet. Par conséquent, l'optimisation de la charge de travail supportée par un nuage ad-hoc local est cruciale pour stimuler le succès des applications limitées dans le temps. Cette thèse étudie l'utilisation optimale des nuages ad-hoc créé par un essaim de drones tout en respectant les contraintes de groupement et de mise en réseau. Afin d'illustrer l'applicabilité des contributions de cette thèse, nous adoptons une mission de cartographie 3D propulsée par des essaims pour des opérations d'intervention d'urgence réelles comme cas d'utilisation.

La première contribution propose un nouveau problème d'optimisation, nommé *Covering-Assignment Problem for swarm-powered ad-hoc clouds* (CAPsac), pour la génération et l'exécution efficaces de charges de travail de calcul multi-nœuds soumises à des contraintes de géolocalisation et de groupement. Le CAPsac vise à minimiser le temps d'achèvement global des tâches déchargées sur le nuage ad-hoc tout en tenant compte des délais de transmission entre les drones. En plus de prouver que le problème est NP-difficile, nous présentons deux formulations combinatoires pour le résoudre. Les résultats sur la solution des formulations montrent que l'une d'entre elles peut être utilisée pour résoudre, dans la limite de temps configurée, plus de 50% des instances réalistes considérées impliquant jusqu'à 200 images et 6 drones.

L'obtention rapide de charges de travail quasi-optimales est cruciale pour le cas d'utilisation

adopté de la mission de cartographie 3D propulsée par essaim. Par conséquent, la deuxième contribution présente une heuristique de programmation mathématique basée sur la décomposition et une heuristique de recherche en voisinage variable pour résoudre le CAP-sac. L'analyse expérimentale montre que les méthodes quasi-optimales proposées atteignent rapidement l'optimalité ou améliorent les meilleures solutions connues pour presque toutes les instances réalistes testées comprenant jusqu'à 1000 images et 15 drones.

Enfin, concernant notre troisième contribution, nous décrivons et évaluons le *swarm-powered Optimized 3D Mapping Pipeline* (OptiMaP) pour les missions de cartographie 3D d'intervention d'urgence. L'OptiMaP est construit comme une application ROS embarquée qui est connectée via un middleware de télécommunication ad-hoc (HEAVEN) fourni par notre partenaire Humanitas Solutions. Les simulations comprenant jusqu'à 5 drones et 363 images ont révélé que les deux versions de l'OptiMaP peuvent réduire de manière significative le temps de réalisation des missions de cartographie 3D propulsée par essaim.

ABSTRACT

Modern sensing applications can be decomposed into sensing and computing phases, where data is typically collected by a dedicated local internet of things system in the sensing phase to be processed in the computing phase. Given the massive amount of data present in such applications, the computing step may require robust storage and computing resources in the cloud. However, common cloud computing infrastructures present issues (e.g., high latency) that are inadmissible for timely manner applications as emergency response and disaster relief operations. The edge computing and ad-hoc cloud paradigms address those issues by providing storage and computing resources close to their users, thus allowing applications to keep their computing phases local. Consequently, those paradigms have been extensively employed in applications with time and network restrictions. For instance, a swarm of drones collecting photos and video over a post-disaster area can create a wireless ad-hoc cloud infrastructure to process the collected data regardless of network connectivity. Therefore, optimizing the workload carried by a local ad-hoc cloud is crucial for boosting the success of timely manner applications. This thesis investigates the optimal use of swarm-powered ad-hoc clouds while fulfilling clustering and networking constraints. In order to illustrate the applicability of the contributions of this thesis, we adopt a swarm-powered 3D mapping mission for real-life emergency response operations as our use case.

The first contribution proposes a new optimization problem, namely the *Covering-Assignment Problem for swarm-powered ad-hoc clouds* (CAPsac), for the efficient generation and execution of multi-node computing workloads subject to data geolocation and clustering constraints. The CAPsac aims to minimize the overall completion time of the tasks offloaded to the ad-hoc cloud while accounting for inter-drone transmission delays. Besides proving that the problem is NP-hard, we present two combinatorial formulations to solve it. Computational results on the solution of the formulations show that one of them can be used to solve, within the configured time limit, more than 50% of the considered real-world instances involving up to 200 images and 6 drones.

Obtaining near-optimal workloads quickly is crucial for the adopted swarm-powered 3D mapping mission use case. Therefore, the second contribution presents a mathematical programming heuristic based on decomposition and a variable neighborhood search heuristic for solving the CAPsac. The experimental analysis shows that the proposed near-optimum methods rapidly reach optimality or improve the best known solutions for almost all tested realistic instances comprising up to 1000 images and 15 drones.

Finally, in our third contribution, we describe and assess the *swarm-powered Optimized 3D Mapping Pipeline* (OptiMaP) for emergency response 3D mapping missions. The OptiMaP is built as an embedded ROS application that is connected through an ad-hoc telecommunication middleware (HEAVEN) provided by our partner Humanitas Solutions. The simulations comprising up to 5 drones and 363 images revealed that the two versions of the OptiMaP can significantly decrease the completion times of swarm-powered 3D mapping missions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

AoA	Angle-of-Arrival
BIS	Block-Information-Sharing
CAPsac	Covering-Assignment Problem for swarm-powered ad-hoc clouds
CAPT	Concurrent Assignment and Planning of Trajectories
CPP	Chinese Postman Problem
DoC	Degree of Completeness
DSC	Drone Small Cell
DTRP	Dynamic Traveling Repairman Problem
EC	Edge Computing
FOV	Field-of-View
GPS	Global Positioning System
GTSP	Generalized Traveling Salesman Problem
HEAVEN	Heterogeneous Embedded Ad-hoc Virtual Emergency Network
HXS	HyperXSpace
IoFT	Internet of Flying Things
IoT	Internet of Things
IP	Internet Protocol
LNS	Large Neighborhood Search
LP	Liner Programming
MILP	Mixed-Integer Linear Programming
MDLP	Mobile Drone Location Problems
MMF	Min-Max Fairness
MMMT	Multi-MSA-Multi-Target
MPC	Model Predictive Control
MVS	Multi-View Stereo
NSA	Nash Seeking Algorithm
OptiMaP	swarm-powered Optimized 3D Mapping Pipeline
OR	Operations Research
pCAPsac	Photo-based CAPsac
PPP	Persistent Patrolling Problem
QoS	Quality-of-Service
rCAPsac	Region-based CAPsac
ROS	Robotic Operating System

RTT	Round-Trip-Time
RWP	Random Way-Point model
SDLP	Static Drone Location Problems
SfM	Structure from Motion
SITL	Software-In-The-Loop
SR	Swarm Robotics
TCP	Transmission Control Protocol
TEF	Task Error Function
UAVs	Unmanned Aerial Vehicles
UV	Unmanned Vehicles
VND	Variable Neighborhood Descent
VNS	Variable Neighborhood Search

CHAPTER 1 INTRODUCTION

The progress on the capabilities of mobile devices formed the perfect environment for the conception of the Internet of Things (IoT) [1] and smart cities [2]. Such paradigms yield huge amount of data and processing demands [3]. Services providing dynamically scalable and virtualized storage and computing resources as cloud computing [4] have been the main force to deliver data storage and processing offload for such scenarios, but they may present high latency and mobility support issues [5].

Similarly to cloud computing, the Edge Computing (EC) paradigm provides equivalent pool of resources and allows computation and storage at the edge of the network, by exploiting computer and storage resources laying along the communication path between the data sources and remote data centers [6]. Naturally, EC emerged as a game-changing alternative to traditional cloud computing systems since it addresses issues that are disruptive to timely manner applications, e.g., high latency and lack of connectivity.

EC benefits from the deployment of Unmanned Aerial Vehicles (UAVs), also known as drones, given their low-cost, mobility, and autonomous capabilities [7–11]. The EC capabilities are enhanced when drones act as an autonomous swarm and cooperate to perform complex tasks. In swarm robotics, simple embodied robots are coordinated in a decentralized manner to perform complex collective tasks in such a way that their performance and resilience are maximized [12]. Consequently, UAV swarming is employed in a wide range of applications as target search and tracking, surveillance, and mapping [13]. In the context of EC applications, a swarm of drones can establish a dedicated wireless network [14] and act as an ad-hoc cloud from which processing and storage resources can be exploited [15–18].

Swarm robotics (SR) has drawn the attention of the Operations Research (OR) community as well [19,20]. In fact, in terms of decentralized methods, OR researchers have investigated how to improve UAV applications in search problems [21–26], target assignment problems [27–37], covering problems [38, 39], scheduling problems [40], and other [41–43]. In the context of this thesis, we investigate the exploitation of a swarm-powered ad-hoc cloud through the proposition of a new NP-hard optimization problem, namely the Covering-Assignment Problem for swarm-powered ad-hoc clouds (CAPsac).

Given a collection of tasks that need to be carried by an ad-hoc cloud, let us define workload as the division of those tasks into distinct groups, i.e., batches of processing tasks. The CAPsac aims to minimize the completion time of processing tasks offloaded to a swarm-powered ad-hoc cloud by jointly optimizing workload generation and workload assignment. This is

fundamentally different from the *decompose-then-allocate* and the *allocate-then-decompose* paradigms [44], which are commonly found in the multi-robot task allocation literature. In particular, we focus on applications with (i) clustering and (ii) network constraints, which means that batch of tasks must be constructed based on their geographical position, and that the data exchanges cannot exceed a given amount of time.

Digital photogrammetry [45, 46] for applications as 3D mapping are commonly employed in the UAV swarming context in which aerial images of a region of interest are captured in order to construct its respective 3D model. That group of aerial images are typically collected by a UAV swarm in cooperative fashion, but the 3D reconstruction process is performed in a dedicated computer [47–53]. Consequently, such centralized approach is vulnerable to latency and connectivity issues and does not exploit the computing and networking resources within the swarm of drones.

The possibility of organizing the UAV swarm to establish an ad-hoc cloud infrastructure [15–18] can highly boost the 3D reconstruction process for swarm-powered 3D mapping missions. Naturally, the CAPsac can be used to optimally create and allocate the multi-node computing workload that constitutes the 3D reconstruction process. Thus, an optimal solution for the CAPsac describes how to minimize the total 3D reconstruction process by optimally i) splitting the area of interest (aerial images) into various sub-regions, and then ii) allocating each sub-region to a certain UAV.

In order to foster the use of the CAPsac by practitioners, we illustrate its potential and applicability through a real-life emergency response application. We employ the CAPsac to optimize the completion time of the 3D reconstruction phase of a swarm-powered 3D mapping mission. Constructing 3D maps of the mission area boosts emergence response since they improve the decision-making and situation awareness during the mission. For instance, those 3D twin models allow first responders to highlight hazards such as damage in roads and buildings, or risk zones [54–57]. Time is precious for those missions and providing 3D maps as quickly as possible despite internet connection is indeed critical.

1.1 Research objectives

The swarm robotics and the edge computing technology rise as relevant research topics given their potential applicability, and the well-established techniques from operations research are used to optimize a long list of drones applications, e.g., [19, 20]. Unfortunately, the literature contains few works combining OR with SR and EC, especially for distributed 3D mapping missions. We focus on combining SR and EC in the context of optimizing the use of

swarm-powered ad-hoc clouds. Optimized workloads for swarm-powered ad-hoc clouds can result in the reduction of their overall completion time and promote the success of real-life applications. The objectives of this thesis are:

- To propose a new optimization problem to improve the workloads created for exploiting swarm-powered ad-hoc clouds while respecting transmission delays, reliability, and spatial convexity constraints.
- To develop solution methods, exact and heuristics, for the proposed problem. Exact methods are crucial to gain insights from the problem which are in turn used to develop fast near-optimal methods. Since time is precious in emergence response operations, devising heuristic methods is crucial to foster effective operations.
- To deploy a UAV-based 3D mapping mission employing the proposed problem to optimize the workload carried by real swarm-powered ad-hoc clouds. The deployment and anticipated analysis of emergency response missions by simulations are key for coordinating real life operations, since they permit evaluating different response protocols without facing an actual crisis.

1.2 Thesis outline

The remainder of this document is organized as follows. Chapter 2 provides basic concepts and notions to the development of this work. The organization of this thesis is presented in Chapter 3. Chapter 4 describes our first contribution in which the CAPsac is formally proposed. Furthermore, besides presenting the NP-hardness proof of the CAPsac, we also present two Mixed-Integer Linear Programming (MILP) formulations to solve the CAPsac in Chapter 4. Chapter 5 presents the proposed decomposition-based and the Variable Neighborhood Search (VNS) heuristics for the CAPsac that were presented in our second contribution. Chapter 6 consists of the deployment of a 3D mapping mission exploiting a swarm-powered ad-hoc cloud via a distributed 3D mapping pipeline creates near-optimal workloads. Finally, we provide a general discussion in Chapter 7 followed by our general remarks concerning the contributions of this thesis in Chapter 8.

CHAPTER 2 LITERATURE REVIEW

In this chapter, we summarize relevant concepts and literature for exploiting swarm-powered ad-hoc clouds as addressed by this thesis.

2.1 Swarm robotics

The terms Swarm Robotics and Swarm Intelligence have been broadly discussed in the specialized literature. The swarm concept has inspiration in the natural swarm behavior among certain animal species, e.g., ants, birds, and bees [58]. Swarm robotics is defined as the research field addressing how to coordinate, in a distributed and decentralized manner, a large group of simple embodied robots to perform collective tasks and maximize the swarm performance [12].

Moreover, a system of multi-robots can be denoted a swarm only if it respects some criteria [12]:

1. The robots must be autonomous and able to interact with the world;
2. the number of robots within the swarm should be large enough to permit scalability. Unfortunately, a minimal quantity isn't well defined in the literature, and 10-20 robots should be sufficient.
3. The swarm must comprise just a few homogeneous groups of robots. The objective is to maximize the system homogeneity and to have simple multi-purpose robots.
4. The robots should be the most simple as possible. They should be designed in such a way that an individual robot would hardly perform the demanded tasks by itself.

Finally, the robots should be restricted to limited sensing and local communication.

The main advantages of adopting swarm robotics — robustness, flexibility, and scalability — lie on their simplicity and number of robots:

- **Robustness**, which is granted by the redundancy and decentralized control of the swarm, allows the swarm to operate in a wide range of scenarios;
- **Flexibility** permits different robots to handle distinct tasks at the same time, since the large number of robots creates a system that has massive parallel power for performing both processing and concrete tasks;

- **Scalability** allows varying the size of the swarm without changing the whole system;

In addition, the simple design of the robots results on cheaper systems, concerning acquisition and maintenance costs, since it has fewer components and there is no need to develop sophisticated operational algorithms. Finally, simple robots have smaller energy requirements. Therefore, the swarms are likely to be efficient in terms of energy consumption.

Those advantages, especially robustness, parallel power, and flexibility, are significant factors that make swarm robotics to overcome other multi-individual systems as multi-robots, sensor networks, and multi-agent systems [58]. As a consequence, the swarming technology has been extensively applied for different sorts of tasks and application domains: self-organized aggregation; autonomous formation pattern; cooperative transport; clustering and storing objects; navigation; target location; self-deployment; collaborative manipulation; and collective decision-making and task allocation [13, 59].

2.1.1 Optimization for swarm robotics

Operations research techniques have already been applied to solve problems concerning unmanned aerial vehicles. Most optimization methods handling UAVs are based on the centralized control paradigm. Both Otto *et al.* [19] and Coutinho *et al.* [20] present broad surveys on optimization approaches for UAV applications. This section lists relevant studies that focus on applying optimization techniques for a swarm of UAVs. Also, the listed studies are grouped following our proposed taxonomy, which arranges the applications accordingly to the addressed problem: *Search Problem*; *Target Assignment Problem*; *Node Covering Problem*; *Scheduling Problem*; and other specific problems.

Search Problem

In a traditional search problem, a set of targets, which can be static or mobile, is searched by the UAV(s). The objective may be either to minimize the time to locate all the targets when their number is known in advance; or to minimize the uncertainty of data concerning the targets' location; or to maximize the probability to find the targets; or to minimize the mission cost given a minimum acceptable probability of finding the targets.

Sujit and Ghose [21] aimed to solve a target search problem exploiting multiple cooperative UAVs. The proposed method handles a discrete version of the region of interest and the respective uncertainty map, which is based on the region's previous knowledge. The proposed decentralized algorithm is inspired by the k-nearest shortest path. Moreover, the paths were built to maximize the uncertainty reduction in the uncertainty map. Multiple trips

are allowed, and both fully connected and partially connected drones are considered. The authors go further on the topic in [60]. In this work, a q-step-looking-ahead planning game model is proposed to explore different game strategies. Also, the UAVs are permitted to refuel/recharge in distinct base stations. A two-agent game model is adopted and three types of strategy — non-cooperative, security, and cooperative — are evaluated. Regarding the non-cooperative strategy, the agents do not communicate their next actions, which are decided according to their best payoff expectation, with each other. Also, no inter-agent communications are implemented by security strategies, according to which each agent act to guarantee a minimum payout. The cooperative strategy is characterized by agents that collaborate to jointly select the set of actions that will maximize the collective payoff. The simulation indicates that the non-cooperative and cooperative strategies have similar results when a fully connected and ideal communication network is considered. In contrast, the non-cooperative strategy has the best results when a non-ideal communication network is used. Finally, a sector partitioning approach is proposed to decrease the computational time for large q-looking-head steps cases.

Oh *et al.* [22] and Oh *et al.* [23] addressed their research to the road search problem with multiple UAVs. The problem is modeled as a Chinese Postman Problem (CPP) that is modified to handle various agents. The proposed approximative method is decentralized and based on a combination of nearest insertion strategy and Dubins paths [61]. Road selection conflicts are solved through a proposed auction algorithm.

Lanillos *et al.* [24] developed decentralized decision-making methods that address the single-UAV and multi-UAV target search problem with static targets. The decentralized algorithms rely on the gradient-descent method over a utility function that comprehends both short-term and final expected rewards. The algorithms minimize a measure based on the non-detection joint probability. In order to keep the problem tractable, only a solution, i.e., a vector of actions, for N-ahead steps is created. Since the gradient-descent method is decentralized, all drones should continuously communicate their estimated target location beliefs. Two strategies are proposed for the multi-target scenario: the separated target location belief; and the united target location belief. The former keeps a different location belief map for each target. The latter merges all the target location belief maps into a single location belief map. Finally, different simulation scenarios are used to compare the proposed methods with another target search method [25] and to evaluate the two multi-target proposed strategies.

Ji *et al.* [26] developed a distributed framework for the cooperative search problem. The objective is to find unknown targets in a predefined region the soonest possible. The drones collect the local sensing information and update the probability and uncertainty maps in a

distributed way. Based on the updated maps and the objective function, the drones change their flight direction in real time.

Target Assignment Problem

The target assignment problem aims to allocate targets to the set of drones in order to visit all targets, whose location is known, while the mission cost is minimized or the mission reward is maximized. Also, the objective can be to maximize the overall visit frequency of the whole target set. Moreover, both the assignment and the path planning characterize a solution to the problem. The problem can be denoted as a task assignment problem where tasks are target locations that need to be visited by the UAVs.

Tang and Ozguner [27] proposed an online method for the cooperative multi-MSA-multi-target (MMMT) problem that provides a target assignment and a surveillance motion plan for each UAV to scout the targets. The authors consider mobile targets whose mobility is dictated by a random walk procedure. The quality of a solution is given by the average amount of time between two consecutive observations of the same target, and the objective is to minimize this average. Moreover, the method has two steps: task decomposition and individual path generation. The former is responsible to group the targets by the k-means algorithm, where k is equal to the number of available UAVs. Then, the k-best algorithm is performed to designate each set of targets to a UAV. In the latter step, the surveillance path of each drone is made by means of a suboptimal gradient-based algorithm. Finally, the solution adjustment is triggered when predefined events occur: a UAV finishes its motion plan; a drone makes a negotiation over a target allocation;

Karaman and Inalhan [28] solved the target assignment problem for a multi-UAV scenario. The proposed method is exact and decentralized; the Dantzig-Wolfe decomposition and the Column Generation strategy are explored in the enumeration. Each drone solves its specific subproblem, and the column with the most negative reduced cost is added to the restricted master problem. All data concerning the restricted master problem is kept updated in all drones. Finally, the proposed method can handle the addition of new targets as well.

Niccolini *et al.* [29] presented a bio-inspired approach for the task assignment problem which is performed by a swarm of agents. The Description Function Methodology is used. A set of functions, denoted Agent Description Function (Agent DF), Current Task Description Function (current TDF), Desired Task Description Function (desired TDF), Task Error Function (TEF), and Degree of Completeness (DoC), are used to coordinate the agents' engagement and task completion. Given an agent i and a task k, the engagement of the agent i to the task k is dictated by a threshold, i.e., a probability which is a function of the magnitude of

the stimulus signal of the task k and the capability of the agent i to perform the task k. In this context, each agent has a different interest to perform each task. Moreover, the stimulus of each task is a function of the task's DoC through time. An agent chooses the task with the highest probability of engagement.

In Viguria *et al.* [30], a distributed task-allocation framework is proposed. Different tasks are considered and applied to fire detection and extinguish missions. Also, computational and real simulations were done, and the concept of service is proposed as well. Services are all complementary jobs that are necessary to perform a given task, e.g., to send images of a point of interest while another drone is extinguishing the fire. The proposed tasks manager perform the S+T algorithm which is a market-based auction algorithm. Moreover, robots bid for a given task and the drone with lowest bid cost is assigned to perform it. Concerning the bid cost computation, a robot considers if it needs complementary services to perform the offered task.

Choi *et al.* [31] solved an online task assignment problem. The proposed method has two stages: optimization order; and communication and negotiation; At first, an initial task assignment is given to the drones. In the first stage, each drone performs a genetic algorithm to minimize the costs of executing its tasks. In the second stage, the drones improve task allocation by exchanging task designations among themselves. The task exchange procedure follows a One-to-One strategy.

Barrientos *et al.* [32] proposed a platform to solve both task assignment and coverage path planning problems for a given area of interest in the context of a precision agriculture application. In real experiments, three UAVs must cooperatively take pictures of a given region in order to construct its mosaic. Two main phases are proposed. First, the area is decomposed, and the resulting sub-areas are allocated to the drones. Second, a covering path for each sub-area is created to minimize the number of turns and mission time of each UAV. The area decomposition and sub-area allocation are done following a negotiation protocol, which is an extension of the alternate-offers protocol [62], and by considering their own characteristics. As the task subdivision and the task allocation are done in the continuous domain, the area of interest, the sub-areas, and the borders are converted to a discrete domain, which is represented by a grid, before the coverage path planning step. Finally, the coverage path planning is based on the *wavefront planner* [63].

Moon *et al.* [33] and Moon *et al.* [34] proposed a hierarchical framework for the task assignment and path planning problems. The objective is to minimize the total cost of the mission, i.e., the summation of the costs of the assigned tasks. The task assignment is based on a negotiation-based strategy, which makes each drone to execute its task with minimal cost.

If a conflict in the task assignment is detected, the UAV with the lowest cost takes the task. The path planning problem is solved locally in the drones by an $A*$ based algorithm, which aims to decrease the distance from the drone to its task. Also, a collision avoidance algorithm for non-expected obstacles is described. In [34], additional filtering procedures are applied to improve the computed paths.

Turpin *et al.* [35] approached the Concurrent Assignment and Planning of Trajectories (CAPT) problem, whose goal is to allocate a set of tasks to the drones and construct collision free UAVs-tasks trajectories. Both centralized and decentralized methods are proposed. The centralized algorithm is a two-phase method. First, the assignment problem is solved by the Hungarian algorithm. If there are more tasks than drones, the problem is solved iteratively. At each iteration, the task assignment is kept, and only the non-allocated tasks are considered in the next iteration. Afterward, the collision-free trajectories are created based on the drones' assignment. The decentralized version assumes that each drone already has a task to perform at the beginning. Considering a given communication range, the drones exchange a task when it is advantageous concerning the chosen objective function, which can be either minimize the sum of traveled distances or minimize the square of velocities.

Enright *et al.* [36] summarized the current methods applied to UAV routing concerning dynamic scenarios, in which new tasks are created on-the-fly following stochastic criteria. The Dynamic Traveling Repairman Problem (DTRP) and the Persistent Patrolling Problem (PPP) are approached by Spatial Queueing Theory-based methods for the single and multi-server cases. The algorithms for the multi-UAV scenarios exploit decentralized spatial partition policies in order to apply the single UAV methods in each sub-region of the constructed spatial partition. Both Voronoi and Power Diagrams tessellations are considered in the discussed partitioning policies. Also, motion, location, and sensing constraints are addressed, and lower bounds are established.

Sadeghi and Smith [37] proposed a decentralized auction-based Large Neighborhood Search (LNS-Auction) method for the task allocation problem. The tasks are points in the Cartesian space, and the task allocation problem is converted to a Generalized Traveling Salesman Problem (GTSP). Moreover, the robot's tasks and scheduling are represented as a tour in the GTSP. Given an initial feasible solution, a random robot starts an auction, and a subset of its assigned tasks are deleted (destroy step) and offered to an aleatory subset of robots. All the robots in the auction, i.e., auctioneer and chosen robots, place bids for subsets of the offered tasks (repair step). For a subset of tasks, its bid value is computed by checking the cost of inserting it in a robot's tour. Nearest insertion, Cheapest insertion, and Farthest insertion can be used in the bidding procedure. Finally, the auctioneer solves an auction problem

and communicates the winners which perform the changes in their assigned tasks. The auctions are triggered until the stop criterion is reached. Also, computational experiments are performed, and bounds on the tour cost are proposed.

Node covering problems

This class of problems focuses on persistent node covering where the nodes can be vehicles, base stations, transmission stations, and ground and aerial targets. Given a set of drones, the primary objective is to keep the nodes always covered by the footprint of at least a single drone. It is useful for communication and tracking applications, for example. A solution for a node covering problem can either the drones' three-dimensional position or the drones' path. The objective function varies accordingly to the application. For instance, a communication application could aim to maximize the Quality-of-Service (QoS) of a set of clients, i.e., nodes.

Zorbas *et al.* [38] formalized the Static and Mobile (dynamic) Drone Location Problems, denoted as SDLP and MDLP, and provide centralized methods for both problems. Accordingly to the characteristics of the problems, only the MDLP has a decentralized online method. The problems' objective is to define the three-dimensional position of the drones in such a way that all the targets are covered by a drone. Moreover, the covering must minimize the number of active drones or minimize the total amount of energy necessary for the mission. Besides needing an initial solution, the decentralized approach is divided into rounds. In each round, the drones compute their new position locally based on the new location of the nearby targets. If it is detected that a single drone can cover the targets of two drones in such a way that the objective function is improved, one of the drones changes its state from active to non-active and assigns their covered targets to the other drone. The opposite may occur when it is better to have two drones handling a set of targets.

Ladosz *et al.* [39] tackled the positioning and path planning problem for a fleet of UAVs that act as transmission nodes in a communication network, which has UAVs and mobile ground nodes (vehicles). The objective is to maximize the transmission quality regarding the restrictions and losses in an urban scenario. The optimization is done by a particle swarm and genetic algorithms, and a decentralized approach is adopted. Furthermore, each drone computes its trajectory and position based on the location and path of all other UAVs and mobile ground nodes.

Scheduling Problems

Given a set of tasks, which can have precedence rules, this class of problems concerns to find a task scheduling for a set of drones. The objective is to complete all tasks as soon as possible, i.e., to minimize the makespan of the tasks.

Caraballo *et al.* [40] proposed a heuristic method for the Assembly planning problem [64]. The objective is to minimize the makespan of a task that can be decomposed in smaller sub-tasks. In other to minimize the total completion time, the heuristic assigns sub-tasks to the drones according to precedence rules. The developed method uses centralized local allocation strategy, denoted Block-Information-Sharing (BIS), to define a decentralized algorithm.

Specific problems

Grancharova *et al.* [41] described a routine to optimize the communication network topology while a trajectory planning problem is solved. Given a set of dynamic and static targets, a set of UAVs, and a base station, the proposed Model Predictive Control (MPC) methods provide a finite-horizon trajectory plan and a network topology which is updated continuously. The trajectory plan is computed in a decentralized way while the loss in the transmission paths of the current network topology is minimized. The network topology optimization routine is performed at the base station at every given time interval. This routine solves a series of maximum flow problems in order to define the best routing paths for transmission of data between the base station and the targets. Also, velocity, acceleration, anti-grounding, airspace, and radio loss constraints are considered. Finally, two simulation scenarios are created to validate and to evaluate the proposed methods.

In Koulali *et al.* [42], a distributed game theory-based method is proposed, and a series of simulations are used as validation. In the formulated non-cooperative game, drones search for mobile users, which are spread in a given area, in order to serve them as a Drone Small Cell (DSC). The UAVs need to transmit packets actively (beaconing) and wait to get the user's answers. A drone's search is successful if it receives at least a predefined number of answers from the users. The goal is to maximize the energy efficiency and the UAV-user encounter probability by creating a beaconing schedule for each drone, i.e., by defining how long each drone will actively search for users. Also, drones and mobile users move randomly in the considered region following a Random Way-point model (RWP). Finally, the proposed method, denoted Nash seeking algorithm (NSA), can reach the unique Nash equilibrium in a small number of iterations and with good energy efficiency in comparison to the always active search strategy.

Xu *et al.* [43] addressed the angle-of-arrival (AoA)-based target tracking problem whose goal is to estimate the target's location based on AoA data of the UAVs. The authors propose a decentralized pseudo-linear Kalman filter for the 3D domain. Also, a gradient descent-based and grid search-based optimization methods are described. Finally, collision avoidance routines and a dynamic communication topology are considered.

Fotouhi *et al.* [65] aims to improve the spectral efficiency of a set of DronesCells, i.e., drone-mounted base stations that are responsible for providing networking resources to mobile users in the established geographical region (cell). Considering a given time interval, the proposed methods act choosing the motion direction of all drones for the next time interval. Game theory, exact and heuristic approaches are evaluated by simulations which show a small difference between the results of the developed methods.

2.2 Edge computing

Cloud computing establish the concept of computing as a utility in which centralized computational resources are available as commodities. Formally, cloud computing is a service over the Internet that provides dynamic scalable resources [4]. Its popularity lies in the possibility of outsourcing computing power, data storage, and services to third parties [66]. Cloud computing is the backbone of many data and computational demanding paradigms as the Internet of Things (IoT) [1], smart cities [2], and big data [3]. Nevertheless, issues as high latency and mobility support are present in the cloud computing solutions [5].

Similarly to cloud computing, Edge Computing (EC) provides lightweight resource virtualization on connected resource-constrained devices with moderate capabilities that are located at the edge of the network, that is, along the path from the data source to the data center [67]. EC mainly differs from cloud computing by proving data storage and computing resources closer to the end user [68]. That grants EC with low latency, small jitter, reduced network load, better data security, and location awareness [67]. EC technology is complementary to cloud computing since it allows data to be collected, processed, and analyzed locally before being sent to a central data center for more complex tasks. Furthermore, EC can be divided into three main paradigms: Cloudlets [69], Fog computing [70], and Mobile Edge computing [71]. We refer the interested reader to an extensive comparison between EC paradigms and cloud computing in [67]

Unmanned Aerial Vehicles, also referred to drones, leverage the IoT resulting in the creation of the Internet of Flying Things (IoFT) [72]. Naturally, EC also takes advantage of the low-cost and aerial capabilities found in the drones. In the context of EC, drones can establish

a dedicated wireless network [14] and act as an ad-hoc cloud [15–18] whose processing and storage resources can be exploited for processing task offloading — IoT devices with low capabilities send computation tasks to be carried in the ad-hoc cloud instead of locally in the device.

For instance, Jeong *et al.* [73] and Jeong *et al.* [8] addressed the bit allocation in upstream and downstream transmissions and the path planning joint problem for a drone providing computing resources for mobile users. The objective was to minimize the total mobile energy consumption in a system with one UAV. The authors proposed two non-convex models that deal with orthogonal and non-orthogonal access scheme in the transmissions, respectively. The computational experiments showed that the successive convex approximations algorithms were able to improve the energy consumption in a system with one drone and three mobile users. Similarly, in Hu *et al.* [10], the authors tackled the joint offloading and path design non-convex problem minimizing the sum of the maximum delay among all ground users that are served by a drone. However, they considered that tasks could be partially offloaded, i.e., the model decides the task ratio processed by the users and the task ratio carried by the drone. Besides extending the problem to minimize the average delay over the users, a penalty dual decomposition-based and a simplified ℓ_0-norm algorithms were proposed. A UAV-powered resource allocation framework for partial and binary computation offloading is devised in [74] in such a way that the weighted sum of the computed bits is maximized. Furthermore, the framework jointly optimizes CPU frequencies, offloading times, UAV trajectory and the wireless power transmitted to the users.

Multi-layer edge computing architectures that offload tasks for both EC servers and drones are addressed in [7,9,75]. In Hu *et al.* [75], the weighted sum of energy consumption over a drone and a set of users is minimized through jointly optimizing resource scheduling, bandwidth allocation, and the UAV's trajectory. Yu *et al.* [9] investigated the use of a drone exclusively to provide connectivity to IoT devices offloading tasks to EC servers while minimizing the weighted sum of the service delay of the devices and the drone energy consumption. That was done by optimizing the drone position, communication and computing resource allocation, and tasks splitting. In contrast, Wan *et al.* [7] adopts a three-layer EC system that addresses both data collection and preprocessing by multiple UAVs, which act as edge servers and offload tasks to a data center for further analysis. Besides designing a reinforcement learning solution provides path planning to optimize the UAV coverage, the authors propose a network scheduling algorithm that exploits the tradeoff between UAV processing and network transmission.

Game theory approaches were employed by [76, 77]. A game in which UAVs opt to process

computation tasks either locally, in a base station or in a data center was proposed by [76]. In such game, the drones aim to minimizing a combination of energy consumption and task delay. A two-level hierarchical game involving computation tasks being offloaded to EC servers and UAVs is designed in [77]: (i) EC servers and UAVs form coalitions to share computing resources; (ii) each agent (EC servers or UAVs) take actions individually based on their long-term payoff. That hierarchical approach models different service providers operating in the same EC system.

Finally, Wang *et al.* [78] presents a two-layer optimization solution for multiple UAVs providing computation resources to a large number of mobile users. The first layer optimizes the number of deployed UAVs and their positions. Given the deployment obtained in the previous layer, a task scheduling integer programming problem minimizing system energy consumption is addressed in the second layer. A greedy method is employed to tackle the large scale in the scheduling problem and find near-optimal solutions.

2.3 Max-min fairness flow allocation

In most network scenarios, the amount of data load is much larger than the network capacity even with optimal routing engineering [79]. As a result, when no traffic (flow) control is done, long transmission delays occur which may violate maximum delay requirements from the network users. Thus, a flow control mechanism is required whenever there exist any capacity constraint between two points on the network. Besides limiting traffic within a network, it is equally important to do it in a fair way such that distinct transmission session priorities and service requirements are met [79].

The Max-Min Fairness (MMF) paradigm is typically employed to provide balanced distribution of resources that are disputed by a set of demands [80]. Naturally, the MMF paradigm is relevant to the Internet Protocol (IP) networks since it provides traffic flow allocation according to a fairness utility function [81]. For instance, the MMF is known to approximate the transmission behavior in the transmission control protocol (TCP) well [82]. In brief, given a set of (traffic) transmission demands, a transmission rate (flow) allocation solution respects the MMF paradigm when it is not possible to increase a transmission rate without decreasing another.

Let us define as $G(N, A)$ the network with a set of nodes N connected by the set of links A. Denote by F the set of all active (traffic) transmission demands, i.e., the set comprising all demands with data to be exchanged. Given a $f_{sd} \in F$ (i.e., an active transmission demand from the source $s \in N$ to destination $d \in N$), let V^{sd} be the routing path from s to d in the

network G, and denote by ϕ_{sd} the transmission rate allocated to f_{sd}. For each link $(i,j) \in A$, let c_{ij} be the capacity of the link from $i \in N$ to $j \in N$ and $\overline{F}_{ij}$ the set of demands passing through the link (i,j).

Formally, a set of transmission rates Φ satisfies the MMF paradigm if and only if there is at least one bottleneck link $(i,j) \in A$ on the routing path V^{sd} of each active traffic demand $f^{sd} \in F$ [80]. A link (i,j) is considered as a bottleneck of traffic demand f^{sd} if and only if [80]:

(i) its capacity is saturated, i.e., $\sum_{f^{ab} \in \overline{F}_{ij}} \phi^{ab} = c_{ij}$ and

(ii) the transmission rate ϕ^{sd} of traffic demand f^{sd} is the highest among those of the other traffic demands routed over link (i,j), i.e., $\phi^{sd} \geq \phi^{ab} \quad \forall f^{ab} \in \overline{F}_{ij}$.

Given a demand $f^{sd} \in F$, let w_{ij}^{sd} be the binary variable equal to one if the link (i,j) is a bottleneck of f^{hd}, as well as let u_{ij} be the highest transmission rate among all the traffic demands carried by a link $(i,j) \in A$, i.e., $u_{ij} = \max_{f^{ab} \in \overline{F}_{ij}} \{\phi^{ab}\}$. The following groups of constraints are used to impose the MMF paradigm for all transmissions F in the context of a MILP optimization problem [81,83]. For the sake of simplicity, the multiplicative parameters introduced in [81,83] to cope with MMF deviations were all considered equal to 1.

$$\sum_{(i,j) \in V^{sd}} w_{ij}^{sd} \geq 1 \qquad \forall (s,d) \in F \tag{2.1}$$

$$\sum_{f^{ab} \in \overline{F}_{ij}} \phi^{ab} \leq c_{ij} \qquad \forall (i,j) \in A \tag{2.2}$$

$$\sum_{f^{ab} \in \overline{F}_{ij}} \phi^{ab} \geq c_{ij} w_{ij}^{sd} \qquad \forall (i,j) \in A, \forall f^{sd} \in \overline{F}_{ij} \tag{2.3}$$

$$u_{ij} \geq \phi^{sd} \qquad \forall (i,j) \in A, \forall f^{sd} \in \overline{F}_{ij} \tag{2.4}$$

$$\phi^{sd} \geq u_{ij} - c_{ij}(1 - w_{ij}^{sd}) \qquad \forall (i,j) \in A, \forall f^{sd} \in \overline{F}_{ij}. \tag{2.5}$$

Constraints (2.1) ensure that all the active demands have at least one bottleneck link on their routing path. The capacity of each link $(i,j) \in A$ is respected through inequalities (2.2). The first condition required to consider a link (i,j) to be a bottleneck is jointly handled by constraints (2.2) and (2.3), which ensure that any bottleneck link (for at least one traffic demand) is saturated. The second bottleneck condition is instead fulfilled via constraints (2.4) and (2.5). Constraints (2.4) force u_{ij} to be **greater** or equal to the highest transmission rate among the demands that are flowing through link (i,j). Finally, constraints (2.5) guarantee that ϕ^{sd} will not be exceeded by any other transmission rate of traffic demands routed over link $(i,j) \in A$ when (i,j) is a bottleneck link of traffic demand f^{sd}.

Additionally, both transmission rates Φ and routing paths can be optimized according to a utility function as presented in [81, 83]. However, whenever the routing paths are already defined, a simple polynomial algorithm can be used to obtain a MMF rate allocation: the *Water (or Proportional) filling* algorithm [79]. Denote by A^k the not saturated links at iteration k, and let F^k be the transmission demands not crossing a saturated link in the beginning of the iteration k. Let us define n_{ij}^k the number of transmission demands that pass through $(i,j) \in A$ and do not cross a saturated link in the iteration k, and denote by c_{ij}^k the used capacity of the link $(i,j) \in A$ at iteration k. Given a rate allocation vector Φ^k at iteration k, the rate increase added to all demand rates $\phi_{sd}^k \in \Phi^k$ that do not cross a saturated link in the iteration k is denoted by r^k. Finally, according to [79], the water filling algorithm is described in the Algorithm 1.

Algorithm 1 Water filling method

1: $k = 1$; $c_{ij}^0 = 0 \quad \forall (i,j) \in A$; $\phi_{sd}^0 = 0 \quad \forall f_{sd} \in F$; $F^1 = F$; $A^1 = A$;
2: **repeat**
3: $\quad n_{ij}^k = |\overline{F}_{ij} \cap F^k| \quad \forall (i,j) \in A^k$;
4: $\quad r^k = \min_{(i,j) \in A^k} \{(c_{ij} - c_{ij}^{k-1})/n_{ij}^k\}$;
5: $\quad \Phi^k = \Phi^{k-1}$;
6: $\quad \phi_{sd}^k = \phi_{sd}^{k-1} + r^k \ \forall f_{sd} \in F^k$;
7: $\quad c_{ij}^k = \sum_{f_{sd} \in \overline{F}_{ij}} \phi_{sd}^k$;
8: $\quad A^{k+1} = \{(i,j) \in A | c_{ij} - c_{ij}^k > 0\}$;
9: $\quad F^{k+1} = \{f_{sd} \in F | f_{sd}$ does not cross any saturated link $(i,j) \in A^{k+1}\}$;
10: $\quad k = k + 1$;
11: **until** F^k is empty

In the Algorithm 1, step 1 sets the initial values of k, c_{ij}^0, ϕ_{sd}^0, F^1, and A^1. At each iteration k of the *repeat* loop (steps 2-11), the transmission rate of demands, which not passing thought a saturated link, are equally incremented by r^k until there none of such demands (i.e., F^k is empty). Furthermore, step 3 computes the number of demands in F^k that pass through the links $(i,j) \in A^K$. The value of r^k is defined in step 4, and the steps 5 and 6 increase the transmission rate of all demands in F^k accordingly. Then, the capacity used by the transmissions demands at each link is updated in the step 7. Finally, steps 8, 9, and 10 updates the A^{k+1}, F^{k+1}, and k, respectively.

2.4 3D reconstruction based on images

The goal of the 3D reconstruction based on images is to construct a 3D model of a given scene or object which is described by a set of images, P, from different perspectives. In general,

the 3D reconstruction procedure consists of the combination of two techniques: Structure from Motion (SfM) [84] and Multi-View Stereo (MVS) [85] techniques [86, 87]. In sequence, finishing steps as meshing and texturing are also performed.

Regarding the SfM [88], it computes the camera parameters and the sparse point cloud, i.e., the approximate positions of the key features in $\mathbb{R}^3$ space given a set of images from different viewpoints. For a specific photo, its camera parameters are the position and orientation (i.e., pose), the focal length, and the pixel sensor size at the moment the photo was captured. Let a key feature be an image patch that identifies some important characteristic of the picture (e.g., corners, edges, and blobs). The SfM-based algorithms are decomposed in feature extraction, feature matching, geometric verification, image registration, triangulation, and bundle adjustment [84, 89]:

- **Feature extraction**: given a photo $p \in P$, detects all key features K_p in the image p. The detection can be done by different methods, e.g., SIFT [90], PCA-SIFT [91], and SURF [92].

- **Feature matching**: For a pair of photos $(p_i, p_j) \in P \times P$, the feature matching identifies their mutual features $K_{p_i p_j} = K_{p_i} \cap K_{p_j}$ based only on their appearance.

- **Geometric verification**: This step removes the misleading elements in $K_{p_i p_j} \forall (p_i, p_j) \in P \times P$. Transformation maps between photos p_i and p_j are computed and used for the task.

- **Reconstruction Initialization**: Computes the sparse point cloud and the camera parameters taking into account only two photos which are carefully chosen and marked as registered.

- **Incremental Reconstruction**: Performs image registration and triangulation steps until there is no more $p \in P$ to register.

  - Image registration: Given a registration metric (e.g., the largest number of mutual features with the registered photos), a new non-registered image is registered and camera parameters updated.

  - Triangulation: Updates the sparse point cloud. This step adds new points, which were identified in the newly registered photo to the sparse point cloud, and improves all other feature locations.

- **Bundle adjustment**: Refines the sparse point cloud and the camera parameters by minimizing the re-projection error.

The Multi-View Stereo algorithms can be different in terms of the scene representation, photo-consistency measure, visibility model, shape prior, reconstruction algorithm, and initialization requirements [85]. To construct the 3D representation of a scene, the MVS methods need the scene image set and the SfM output, i.e., camera parameters and sparse point cloud. Given those inputs, the MVS algorithm performs the dense point cloud generation — from the set of photos P and the SfM output, a detailed 3D structure of the scene is reconstructed.

Finally, after performing the SfM and MSV algorithms, meshing and texturing steps still need to be performed to transform the dense point cloud into a complete 3D model:

1. **Meshing**: Given a dense point cloud as input, the 3D model surface is estimated (i.e., triangle meshes). Different methods can be applied to this step, e.g., [93, 94].

2. **Texturing**: Based on the appearance of pixels in the input photos, characteristics like color, illumination, and texture are added to the 3D model, e.g., [95].

The quality of the final 3D model is highly correlated with the quality of the set P concerning resolution, viewpoint variety, and the number of images. Therefore, in the context of 3D reconstruction based on aerial images, generating adequate locations where the photos need to be taken (i.e., waypoints) is crucial. When generating waypoints, parameters as perspective, lens quality, overlap, coverage, and object geometry affect the number of photos required to produce 3D maps with good resolution [87, 96]. The same amount of waypoints may be appropriate to regions of interest with different dimensions when adjusting those parameters [57]. Among those parameters, the overlaps are important for the 3D reconstruction since they define how much the photo footprints (i.e., area captured by a photo) may overlap between each other. Large overlaps improve the resulting 3D model but may require more photos to capture the whole region of interest. Usually, good values of overlaps range between 60% and 80% [96].

2.4.1 3D reconstruction by unmanned vehicles platforms

Considering the use of unmanned vehicles (UV), the 3D reconstruction has been applied to several fields. Nex and Remondino [97] had identified applications including forestry [98–100], archaeology and architecture [101–105], environmental monitoring [106–110], emergency management [111–113] and precision agriculture [114–116]. Unfortunately, the current applications use a centralized approach to perform the 3D reconstruction and do not take advantage of the distributed processing power within the multi-drone platform. For instance, Meyer *et al.* [47] applied 3D reconstruction to survey a heritage site in Mexico but the 3D map is created offline by a computer in the base station.

Moreover, multiple unmanned vehicles are used commonly in the photo acquisition step. Doherty *et al.* [48] proposed a collaborative task-delegation framework for a multiple-robot platform, and a 3D reconstruction task is used as validation. In this context, the area selected for the 3D reconstruction task is split, and each sub-area is assigned to an available UAV for scanning. The delegation procedure does not apply any optimization method, and the 3D reconstruction is done at the base station.

Loianno *et al.* [49] defined a collaborative online procedure which exploits a multi-smartphone-based UAV system to explore and map an unknown environment. The proposed iterative method generates 3D point clouds for indoor spaces. The UAV creates a local sparse point cloud, and the base station is responsible for merging all the sparse point clouds. The base station also generates and sends the next acquisition points to the UAVs at each iteration. To guarantee a collision-free 3D trajectory for the UAVs, a centralized and a semi-decentralized version of 3D path planning methods were proposed.

Schmiemann *et al.* [50] presented a distributed scanning procedure able to combine information among ground and aerial agents. Their algorithm is centralized and needs a base station to control the unmanned vehicles and to generate a mapping visualization. Real experiments evaluated the proposed procedure and network architecture.

Kobayashi *et al.* [51] described how to create 3D reconstructions based on images from crowdsourcing. In this method, micro-tasks are proposed to the population which takes pictures or videos of the demanded location, and the 3D reconstruction is done on a dedicated computer.

Golodetz *et al.* [52] performed a collaborative data acquisition with several different robots and proposed an incremental online 3D reconstruction process. The robots (i) collects the scene information, (ii) performs local pose scene tracking, and (iii) computes their approximate location; In addition, the base station improves the robots approximate location in the unknown environment at each iteration. Finally, the base station (server) uses the data from the robots to build the 3D model incrementally using a GPU.

Other authors focused on combing data from different sensing methods to generate more data to the 3D reconstruction algorithm. For instance,Hinzmann *et al.* [53] presented a way to fuse image and laser scanning data for the 3D reconstruction process. Surmann *et al.* [117] proposed a change in the registration method to combine data from aerial and ground unmanned vehicles, which are equipped with different sensors.

Regarding the 3D reconstruction process, changes in the procedure were proposed as well. Roters *et al.* [118] developed a new 3D reconstruction pipeline which suits better in a multi-UAV scenario. Faessler *et al.* [119] designed an autonomous single drone system able to follow

high-level commands and transmit the collected images to the base station. The proposed UAV platform can be used for both indoor and outdoor spaces since the drone can locate itself without the use of a GPS device. Finally, the base station uses an online 3D reconstruction algorithm that allows visualizing the 3D model creation in real time.

In addition to those works, other studies related to the 3D reconstruction and UV were performed. Huh *et al.* [120] described a way to simplify data in the drones. The data reduction decreases the amount of information transmitted across the drones. As a result, the battery autonomy of the UAVs is improved. Milani and Memo [121] analyzed how different UAV formations, when scanning a region, affect the 3D reconstruction quality. Furthermore, camera-in-view correction methods were proposed to remove drones inside of another drone's field-of-view (FOV). Finally, heuristic methods for path planning of the scanning step were proposed by [122] and [123].

CHAPTER 3 ORGANIZATION OF THE THESIS

The main objective of this thesis is to improve the use of swarm-powered ad-hoc clouds by optimizing the workload of tasks carried by such infrastructure. We illustrate the proposed optimization problem as well as the developed near-optimum heuristics, and data processing pipeline in the context of a 3D mapping mission for emergency response operations. In this chapter, we present how the thesis is organized and detail the way each chapter supports accomplishing the objectives listed in Chapter 1. The three main contributions of this thesis are presented in chronological submission/implementation order in Chapters 4, 5, and 6, respectively.

Chapter 4 presents the paper titled "*The Covering-Assignment Problem for Swarm-powered Ad-hoc Clouds: A Distributed 3D Mapping Use-case*" and published in *IEEE Internet of Things Journal*[1]. Swarm-powered missions are usually decomposed into the (a) photo collection and (b) data processing (i.e., 3D reconstruction) phases. Most of the works in the literature take advantage of the multiple drones only during the photo collection whereas the data processing is performed in a dedicated server or workstation. In this contribution, we aimed to efficiently exploit the computing power within of members (e.g., UAVs, unmanned ground vehicles, and fixed computing resources) responsible for the decentralized data processing phase of a mission. In order to optimize the use of swarm-powered ad-hoc clouds, we proposed a new NP-hard optimization problem: the *Covering-Assignment Problem for Swarm-powered Ad-hoc Clouds (CAPsac)*. Given a set of processing tasks required by the data-processing step, the CAPsac solution provides a workload that maps the tasks to swarm members so as to minimize its completion time. That is done by jointly optimizing the workload creation and assignment. The workload creation comprehends the creation of batches of processing tasks which must be performed all together according to clustering constraints. The workload assignment maps batches of tasks to swarm members. In the context of a swarm-powered 3D mapping mission, the workload creation splits the target region, i.e., the region to be represented in the 3D model/map, into smaller convex sub-regions, which are defined by sets of photos; the workload assignment defines how the sub-regions (set of photos) will be reconstructed across the swarm members. Preceding the NP-hardness proof of the CAPSac, we present two different Mixed-Integer Linear Programming (MILP) formulations for the CAPsac: one photo-based CAPsac (pCAPsac) and another region-based CAPsac (rCAPsac). We assessed the proposed formulations through computational experiments con-

[1] Available at [124]

ducted with a set of unweighted and weighted realistic benchmark instances, which are also created in this first contribution.

Chapter 5 refers to the paper titled "*Heuristics for optimizing 3D mapping missions over swarm-powered ad-hoc clouds*" and submitted to the *Journal of Heuristics*[2]. The CAPsac optimizes the completion time of a set of tasks offloaded to a swarm-powered ad-hoc cloud, for which exact methods to solve it were provided in our first contribution. However, given the current state-of-art of the modern MILP solvers and the amount of offloaded processing tasks, the time required to obtain the optimal workload creation and assignment (i.e., the optimal solution for the CAPsac) may be impractical for timely manner applications as in the case of an humanitarian emergency response use case. Thus, fast methods able to reach near-optimum solutions for a large range of scenarios are crucial to improve the change of success of such missions. In this contribution, we developed a mathematical programming heuristic based on decomposition and a Variable Neighborhood Search (VNS) heuristic to minimize the completion time of the 3D reconstruction process according to the CAPsac. Computational experiments were conducted with the unweighted and weighted realistic instances proposed in the first contribution to assess the performance of the proposed heuristic methods. They demonstrated that both heuristics were able to either quickly achieve near-optimal solutions or rapidly improve the best known completion times for a vast number of instances.

Chapter 6 presents the *swarm-powered Optimized 3D Mapping Pipeline* (OptiMaP) for emergency response operations. The careful design of exact and near-optimal methods is crucial to understand an application and provide relevant solutions. Nevertheless, even with highly effective techniques, there is still the challenge of ensuring that the proposed methods work well after production. Hence, in our last contribution, we devised and deployed a distributed 3D mapping pipeline for emergency response operations. We analyzed how much a distributed 3D mapping pipeline is improved by exploiting the CAPsac for optimizing the use of swarm-powered ad-hoc clouds in real-life situations. In particular, we use the VNS-based method of our second contribution for quickly solving the CAPsac instances within the OptiMaP. The proposed pipeline was deployed through a simulator able to provide realistic physics and visuals besides wireless multi-hop network emulation. We compared the VNS method with a centralized approach and a baseline greedy heuristic to assess the advantages of adopting the algorithm for workload generation and allocation step in realistic simulated scenarios.

Finally, Chapter 7 provides a summary of our work as well as a general discussion, whereas Chapter 8 presents our concluding remarks and future directions for the research carried in this thesis.

[2]Preprint available at [125]

CHAPTER 4 ARTICLE 1: THE COVERING-ASSIGNMENT PROBLEM FOR SWARM-POWERED AD-HOC CLOUDS: A DISTRIBUTED 3D MAPPING USE-CASE

Authors: Leandro R. Costa, Daniel Aloise, Luca G. Gianoli, and Andrea Lodi.
Published in *IEEE Internet of Things, 2021*[1]

Abstract: The popularity of drones is rapidly increasing across the different sectors of the economy. Aerial capabilities and relatively low costs make drones the perfect solution to improve the efficiency of operations that are typically carried out by humans. Besides automating field operations, drones acting de facto as a swarm can serve as an ad-hoc cloud infrastructure built on top of computing and storage resources available across the swarm members and other elements. Even in the absence of Internet connectivity, this cloud can serve the workloads generated by the swarm members and the field agents. By considering the practical example of a swarm-powered 3D reconstruction application on top of such cloud infrastructure, we present a new optimization problem for the efficient generation and execution of multi-node computing workloads subject to data geolocation and clustering constraints. The objective is the minimization of the overall computing times, including both networking delays caused by the inter-drone data transmission and computation delays. We prove that the problem is NP-hard and present two combinatorial formulations to model it. Computational results on the solution of the formulations show that one of them can be used to solve, within the configured time-limit, more than 50% of the considered real-world instances involving up to two hundred images and six drones.

Keywords: *Cloud Computing, Swarm, 3D Reconstruction, Workload Optimization*

4.1 Introduction

An Unmanned Aerial Vehicle (UAV) — otherwise commonly known as drone — is a flying vehicle whose weight can vary, according to the targeted applications, from a few hundreds grams to hundreds of kilos. The popularity of drones is rapidly increasing across the different sectors of the economy. Aerial capabilities and relatively low CAPEX/OPEX costs make UAVs the perfect solution to improve the efficiency of those operations that are typically carried out by humans, e.g., building inspection, photo collection, area surveillance, etc.

[1] Available at [124]

In normal operations, drones are remotely controlled by human pilots through wireless remote controls. However, by setting the UAV autopilot in *auto off-board* mode, a drone can operate in a fully autonomous manner by following the inputs generated by an on-board flight computer directly connected to the autopilot. This capability can be leveraged to create fleets of autonomous UAVs collaborating to fulfill the desired missions. This is achieved by installing a collaborative drone application on each on-board flight computer of the fleet and by connecting these latter on the same wireless network.

The possibility of organizing drones in fleets of autonomous and collaborating entities naturally attracted the attention of swarm robotics scientists [58]. Swarm robotics studies how to reproduce, with the help of artificial agents, those swarming behaviors typically observed in nature — in ant colonies, bee swarms, bird flocks, etc. Swarm behaviors have the potential to revolutionize the world of robotized applications — including UAV applications — because of their promise of jointly achieving maximum performance and maximum resilience through the power of distributed interactions that do not require any form of centralized supervision.

Swarming UAVs can be deployed to support operations in a long list of domains [97], including forestry [98–100], archaeology and architecture [101–105], environment monitoring [106–110], emergency management [111–113] and precision agriculture [114–116].

Accordingly, the operations research community has been investigating approaches to improve the efficiency of UAV-powered applications [19, 20]. In particular, decentralized optimization methods have fostered search problems [21–26], target assignment problems [27–37], node covering problems [38, 39], scheduling problems [40], and other cases [41–43].

In a complex mission, the UAV swarming component is typically dedicated to data collection duties, e.g., taking pictures, producing video feeds, sniffing wireless signals. Other technologies are then involved in processing the collected data and producing the desired output. For instance, digital photogrammetry algorithms [45, 46], can be leveraged to elaborate the images collected by the swarming UAVs, by extracting and displaying the relevant 2D/3D geometric information.

The data-processing phase — 3D processing phase in the photogrammetry use-case [48–52, 121] — is typically executed in the cloud and in a centralized fashion [2, 18, 71]. However, to mitigate Internet connectivity and network latency issues, the distributed power of the UAV swarm can be leveraged to establish an ad-hoc cloud infrastructure [7, 9, 77, 126, 127]. In fact, such infrastructure can be exploited to execute the data processing phases of the considered mission [10, 15–18, 128]. This approach aims to exploit the *power of the many* — many embedded microcomputers of limited power are installed on the swarming UAVs — to replace the computer power typically available in a powerful work station reserved in the

cloud.

This paper addresses the problem of optimizing the exploitation of a swarm-powered ad-hoc cloud, by jointly dealing with two interrelated aspects of the data-processing stage:

- the workload generation, i.e., definition of the computing application elements and of the corresponding set of data input.
- the workload scheduling/assignment, i.e., mapping of computing application elements and physical swarm members.

In particular, we consider the generation and placement of workloads whose input data are subject to geolocation/clustering constraints (e.g. [129, 130]).

Practically speaking, the collected data are bound to a location and must be processed in batch of neighboring samples: in the 3D reconstruction use-case, this means that groups of neighboring pictures have to be processed by the same computing element. This additional constraint has a non-negligible impact when dealing with a swarm powered ad-hoc cloud: due to the distributed nature of the swarm-powered data collection stage, the whole input data-set may end up being completely scattered across the UAVs of the swarm. If not properly dealt with during the workload generation/scheduling processes, this aspect may severely deteriorate the whole process performance due to unnecessary data transmission delays (input data must be received by the corresponding computing application element).

For the purpose of illustrating the applicability of our approach to a real-life application, we adjust the proposed solution to a relevant use-case scenario from the emergency response field. Our use-case is a perfect example of a real-life application subject to geolocation constraints that highly benefits from swarm-powered ad-hoc cloud infrastructure [131]. In fact, the drone swarm is able to perform 3D reconstruction of a region of interest — comprising five hundred photos — on top of twenty Raspberry Pis [132] microcomputers [133].

In such context, the 3D map of a given region of interest is used to improve the decision making process and the operator situational awareness through the availability of a 3D digital twin of the operation area, where the elements of interest can be even selectively highlighted, e.g., building or road damages, risky areas, etc. Producing the relevant 3D maps in a timely manner (near real-time), even when the cloud connectivity is not available, is crucial to increase the chances of success of an operation. To this purpose, we introduce a new optimization problem, namely the Covering-Assignment Problem for swarm-powered ad-hoc clouds (CAPsac). Given a set of geo-positioned aerial pictures (data) that are physically distributed across a set of UAVs (stored on the embedded microcomputers on the drones),

CAPsac minimizes the 3D mapping (data-processing phase) completion time by jointly computing:

- the optimal workload configuration/the optimal covering of photos, i.e., splitting the overall photographed region across multiple convex sub-regions, and
- the optimal workload scheduling/the optimal assignment of photographed sub-regions to UAVs, i.e., deciding which drone (its embedded microcomputer) is responsible for the 3D reconstruction of a photographed sub-region.

It is worth pointing out that, differently from the *decompose-then-allocate* and the *allocate-then-decompose* paradigms [44] broadly adopted in (both the cloud computing optimization and) the multi-robot task allocation literature, CAPsac is an integrated decision model that handles workload generation (photo covering or sub-region splitting) and workload assignment (sub-region to UAV assignment) at the same time. Other works employ similar principle by jointly optimizing UAVs deployment and task/position allocation [78,134]. However, they propose evolutionary heuristics which, by definition, cannot certify the optimality of the obtained solutions.

The remainder of the paper is organized as follows. The next section formally introduces the CAPsac problem, by clearly highlighting how the general problem designed for swarm-powered ad-hoc clouds is naturally applied to optimize the execution of a distributed 3D mapping application. Once the relationship between the general problem and the specific 3D mapping use-case will have been clearly proved through Section 4.2, it will be possible to present the remainder of the paper by directly referring to the latter. We believe that this editing approach will allow the reader to better grasp the details and the added value of the proposed solution. Two mathematical programming formulations to solve the CAPsac problem are described in Section 4.3 while Section 4.4 presents the NP-hardness proof for the problem. Finally, Section 4.5 presents and discusses the computational results obtained by experimenting with realistic 3D reconstruction instances, while Section 4.6 summarizes our concluding remarks.

4.2 Covering-Assignment Problem for Swarm-powered Ad-hoc Clouds - CAPsac

A swarm-powered mission can be typically decomposed in two phases:

i **Data collection**: the UAVs of the swarm dynamically collaborate to collect all the necessary information within the area of interest. In a swarm-powered 3D mapping

mission, this phase corresponds to the photo-collection process meant to shoot the required aerial photos of the selected area. Note that the set of required pictures is typically computed by a dedicated mapping software and is merely an input of the mapping mission.

ii **Data processing**: the collected data are collaboratively processed by the ad-hoc cloud built on top of the microcomputers installed on the UAVs to produce the desired output. During this process, thanks to the swarm-powered ad-hoc cloud, the computing workload can be parallelized over the available computing units. Furthermore, the collected data can be transferred over the inter-drone wireless network to satisfy the input requirements of the distributed processing tasks. In a swarm-powered 3D mapping mission, this phase corresponds to the 3D-processing process meant to compute a 3D point cloud and/or a 3D mesh of the selected area.

The proposed CAPsac problem deals with the optimization of the data (3D) processing phase and has no direct control on the data (photo) collection strategy. Given the set of data (aerial pictures) just collected by the UAVs, CAPsac aims at minimizing the overall processing time required to compute the desired output (3D map).

An explanatory CAPsac problem instance involving a swarm of four drones performing a 3D mapping mission is represented in Fig. 4.1. The full lines delimit the area of interest represented by the set P of aerial pictures just captured by the four drones during the photo collection phase. Each photo $p \in P$ was taken by a specific drone (which also stores it in memory). Furthermore, each picture must be processed during the 3D processing phase to guarantee a proper reconstruction.

In Fig. 4.1, the positions of the four drones are represented by the large symbols "$\times$", "$+$", "$\diamond$", and "$\circ$". Not all drones may be equipped with microcomputers powerful enough to support the 3D processing workload. In the example of Fig. 4.1, only two drones are considered *3D-capable*, those represented by the $+$ and the $\times$ symbols.

Each photo is characterized by its shooting location, which is denoted by the small versions of the symbols previously used to represent the UAVs: the pictures represented by a small $+$ were shot by the drone represented by the large $+$, and so on. Note that, given the dynamic nature of the decentralized decision-making process employed by the swarm of drones [44], it is impossible to know a-priori which UAV will shoot which picture.

A solution of the CAPsac problem describes how to:

- Split the processing workload into multiple processing (application) components, each

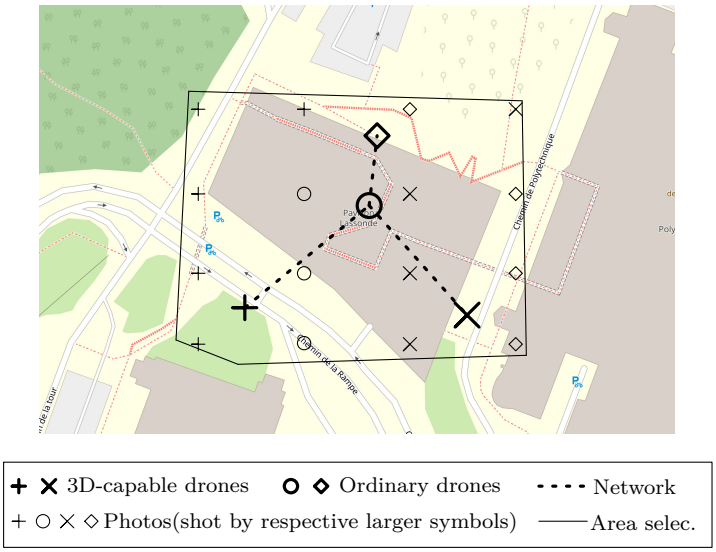


Figure 4.1 Explanatory instance of CAPsac problem accordingly to the 3D mapping use-case.

responsible for dealing with a specific subset of the collected data, e.g., of the aerial pictures. Note that in the 3D reconstruction use-case each 3D reconstruction sub-task corresponds to a specific sub-region and requires as input all the aerial pictures that belong to that sub-region.

- Assign each processing component and all its corresponding input data to at least one of the computing elements available within the swarm-powered ad-hoc cloud, i.e., the microcomputers installed on the swarming-UAVs or on any other ground element connected to the swarm itself.

The optimal solution of CAPsac minimizes the latest processing time among all the involved computing elements, which corresponds to minimizing the makespan of the whole 3D reconstruction process. Three main issues cannot be ignored when assigning the photos (and thus the sub-regions) to the optimal 3D processing drones. Note that for sake of simplicity, we consider the case of not more than one computing device available on each drone.

i A feasible region (workload) subdivision is characterized by the creation of a *spatial-convex covering*: the union of the sub-regions corresponds to the whole region and the photos associated to each sub-region must be a *spatial-convex set*. Accordingly, a photo can be assigned to a drone if and only if it lies inside the convex hull of all the photos assigned to that drone. Fig. 4.2 illustrates a set of photos which is **not** spatial-convex. The assigned photos are represented by colored "•" symbols. The "○" symbols

represent photos that do not belong to the set, which are hence assigned to other drones. Spatial-convex sets are crucial to perform the 3D mapping procedure since the presence of non-overlapping photo footprints (represented by the dashed colored rectangles in Fig. 4.2) makes the 3D reconstruction of the associated region impossible. Fig. 4.3 shows an example of a photo spatial-convex set assigned to one 3D-capable drone. It is worth pointing out that the creation of a spatial-convex covering is required by any workload operating over geo-located/clustered data to be processed in neighboring batches.

ii As the sub-regions (and their corresponding pictures) are assigned to the 3D-capable drones, a drone may need to require some input pictures (data) from the other swarm members. The CAPsac considers a pre-defined single-tree network topology built by a networking middleware running on the swarming drones [14]. Note that in Fig. 4.4, the network links are represented by the pointed lines. Besides, in a single-tree network topology, only one routing path exists to connect each pair of UAVs. A drone cannot start the 3D reconstruction of the assigned sub-region until all the required photos are received. The TCP communication protocol is widely applied in engineering to achieve reliable transmissions and flow control, and it is adopted to model the swarm communications in the CAPsac problem. According to [82], a good way to approximate the transmission behavior concerning the TCP protocol is to assume that the transmission rate allocation follows the Max-Min Fairness - MMF paradigm [79]. Thus, minimizing the makespan of the 3D reconstruction requires that all the transmission rates of the network follow the MMF paradigm. It is well known that, in multi-hop wireless networks, the allocation behavior of TCP may deviate from the ideal MMF paradigm [135]. However, when a traffic engineering problem involves elastic traffic demands, i.e., the transmission rate is autonomously determined by a distributed end-to-end congestion control scheme, the choice of considering even just an approximated form of fairness plays a crucial role to avoid those poor routing solutions that would be otherwise obtained by approximating the traffic demands as inelastic [83]. Furthermore, note that sources of MMF deviations, such as round-trip-time variance and multi-connection schemes [135], can be explicitly accounted for with the help of simple multiplicative parameters added to a specific group of constraints [83]. Another more complex source of deviation, i.e., the hidden node phenomenon, could be addressed by means of robust optimization techniques [136, 137] that fall beyond the scope for this work.

iii Finally, a *reliability factor* should be considered to immunize the CAPsac assignment

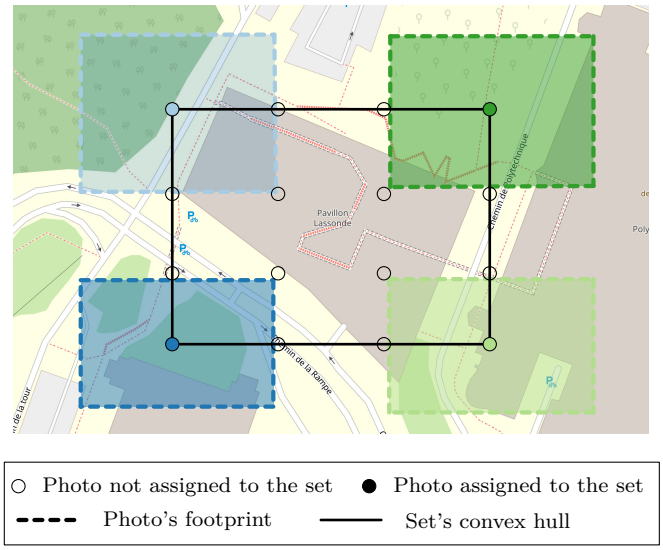


Figure 4.2 Ordinary set and the respective convex hull.

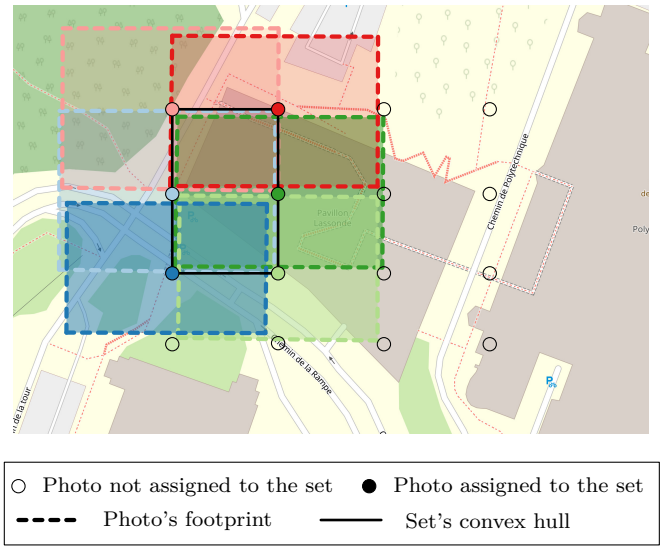


Figure 4.3 Spatial-convex set and the respective convex hull.

with respect to drone malfunctions. The reliability factor defines the minimum number of drones (computing elements) that should process each sub-region.

Fig. 4.4 shows a feasible solution to the CAPsac problem optimizing the makespan of a 3D mapping mission and considering a reliability factor equal to one. The dashed and the dashed-and-pointed lines define a feasible spatial-convex covering. The number of sets comprising the covering is equal to the number of 3D-capable drones. For instance, the covering in Fig.

4.4 has only two sets. The photos lying within the left sub-region are processed by drone +, whereas those in the right sub-region are elaborated by drone ×.

Given that the relationship between the general CAPsac and the swarm-powered 3D mission use-case has been clearly established, we will present the remainder of the paper referring directly to the specific use-case. Thus, it will be possible to the reader to better grasp the details and the added value of our proposed optimization problem.

4.3 Mathematical programming formulations

4.3.1 Common definitions

Let us consider a swarm of drones D of different types which is responsible for the 3D reconstruction of a region described by the set P of photos. Let $\bar{D} \subseteq D$, with $|\bar{D}| = m$, be the set of 3D-capable drones that have enough computing power to support the 3D reconstruction workloads. The location of all photos $p \in P$ are also known.

Given a photo $p \in P$, let λ_p and μ_p be the non-negative real parameters representing, respectively, the estimated processing time of p and the storage space occupied by p (expressed in megabytes). Also, for each drone $d \in D$, let θ_{dp} be the binary parameter equal to 1 if photo $p \in P$ is stored on drone d. The subset of photos processed by a drone directly corresponds to a specific sub-region. Therefore, note that the number of sub-regions are hence equal to the number of 3D-capable drones available in the swarm.

Some pictures may have to be transferred among different drones to respect the computed sub-region assignment configuration. The picture transmission is supported by an undirected transmission tree $T{=}(N, A)$, where the nodes of set N correspond to the swarming drones and the arcs of set A represent the device-to-device communication links (e.g., Wi-Fi links) between the drone themselves. Furthermore, let F be the set of traffic demands defined for each pair of drones $(h, d) \in D \times \bar{D}$, where f^{hd} denotes the demand (possibly null) between drones h and d. Also for each $(i, j) \in A$, denote c_{ij} the transmission capacity of the link (i, j).

Finally, the maximum allowed time for transmitting the traffic demands through the network is denoted by $\hat{T}$.

With the support of the notation just introduced — grouped in Table 4.1, we present two different Mixed-Integer Linear Programming (MILP) formulations to optimize the 3D-processing phase of 3D mapping missions with UAV swarms:

- The Photo-based CAPsac (pCAPsac), where each picture $p \in P$ is assigned to one

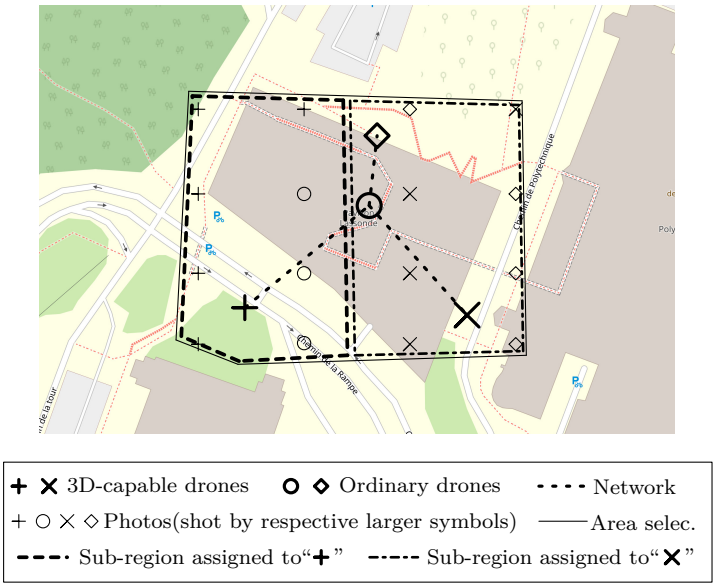


Figure 4.4 Spatial-convex covering and its assignment optimizing the makespan of a 3D mapping mission.

Table 4.1 CAPsac parameters in the context of the 3D mapping use-case.

Parameters	Description
λ_p	estimated processing time of a photo p
μ_p	amount of data of a photo p in Mb
θ_{dp}	equal to 1 if drone d has the photo p stored in its memory
F	set of traffic demands between each pair of drones
c_{ij}	transmission capacity of the link $(i, j) \in A$
$\bar{c}^{hd}$	minimum c_{ij} on the sole routing path of f^{hd}
σ	reliability factor
m	number of drones (equiv. number of sub-regions) which can perform 3D reconstruction
$\hat{T}$	maximum allowed time for exchanging photos between drones

sub-region, among a pre-defined set of initially empty sub-regions.

- The Region-based CAPsac (rCAPsac), where all the feasible rectangular sub-regions are given; the formulation is responsible for selecting the optimal set of sub-regions among those available (presented at the Appendix).

4.3.2 Photo-based CAPsac

In the pCAPsac formulation, the decision variables will be optimized to compose $R = \{1, \ldots, m\}$ sub-regions (equiv. subsets of photos) to be reconstructed by the set of drones. The formulation aims to jointly perform two assignment operations:

- each picture $p \in P$ is assigned to one sub-region $r \in R$,
- each non-empty sub-region $r \in R$ is assigned to one 3D-capable drone $d \in \bar{D}$.

To this purpose, let y_p^r and x_d^r be the binary variables equal to 1 when, respectively, photo $p \in P$ is assigned to sub-region $r \in R$, and sub-region $r \in R$ is assigned to drone $d \in \bar{D}$. Furthermore, let g_{dp}^r be the binary variables equal to 1 if drone $d \in \bar{D}$ is assigned to a sub-region $r \in R$ that contains picture $p \in P$.

Assignment constraints

To obtain a proper 3D-reconstruction, each photo must be processed at least one time, i.e., it must belong to at least one sub-region:

$$\sum_{r \in R} y_p^r \geq 1 \quad \forall p \in P. \tag{4.1}$$

Similarly, each sub-region must be assigned to at least σ 3D-capable drones, with σ representing the previously introduced reliability factor meant to immunize the system toward possible drone failures:

$$\sum_{d \in \bar{D}} x_d^r \geq \sigma \quad \forall r \in R. \tag{4.2}$$

Finally, let us introduce the group of constraints necessary to correctly compute the g variables without introducing any non-linearity:

$$g_{dp}^r \leq x_d^r \qquad \forall p \in P, \forall r \in R, \forall d \in \bar{D} \tag{4.3}$$

$$g_{dp}^r \leq y_p^r \qquad \forall p \in P, \forall r \in R, \forall d \in \bar{D} \tag{4.4}$$

$$g_{dp}^r \geq y_p^r + x_d^r - 1 \qquad \forall p \in P, \forall r \in R, \forall d \in \bar{D}. \tag{4.5}$$

That is, constraints (4.3)-(4.5) are the classical McCormick inequalities [138] such that $g_{dp}^r = x_d^r y_p^r \; \forall p \in P, \forall r \in R, \forall d \in \bar{D}$ are represented in linear form.

Spatial-convexity constraints

To properly work, state-of-the-art 3D reconstruction algorithms [87] have to deal with convex regions and/or sub-regions, which is also equivalent to work with spatial-convex sets of photos. To this purpose, we approximate the convex hull of the set of photos assigned to a drone by its smallest enclosing hyperrectangle. This is not a bad approximation since 3D mapping missions often considers 50% to 80% of photo overlapping [96].

Since the GPS position of each photo shooting point is known, let C be the set of distinct picture longitudes and let L be the set of distinct photo latitudes. Note that the following relations are always respected: $1 \leq |C| \leq |P|$ and $1 \leq |L| \leq |P|$. For each sub-region, there exists a finite set of photo latitudes and longitudes that represents the bounding rectangular box, i.e. the approximated boundaries of the sub-region.

The boundary of a sub-region r is defined by its left (α^r), right (β^r), bottom (γ^r), and top (ω^r) borders. Binary variables α_c^r, β_c^r, γ_ℓ^r, and ω_ℓ^r are used to designate the latitudes and the longitudes defining these borders:

- Binary variable α_c^r is equal to one if longitude $c \in C$ delimits the left border of sub-region $r \in R$,
- Binary variable β_c^r is equal to one if longitude $c \in C$ delimits the right border of sub-region $r \in R$,
- Binary variable γ_ℓ^r is equal to one if latitude $\ell \in L$ delimits the bottom (inferior) border of sub-region $r \in R$,
- Binary variable ω_ℓ^r is equal to one if latitude $\ell \in L$ delimits the top (superior) border of sub-region $r \in R$.

Each sub-region $r \in R$ must be associated to a unique tuple of borders:

$$\sum_{c \in C} \alpha_c^r = 1 \qquad \forall r \in R \tag{4.6}$$

$$\sum_{c \in C} \beta_c^r = 1 \qquad \forall r \in R \tag{4.7}$$

$$\sum_{\ell \in L} \gamma_l^r = 1 \qquad \forall r \in R \tag{4.8}$$

$$\sum_{\ell \in L} \omega_l^r = 1 \qquad \forall r \in R. \tag{4.9}$$

To respect the sub-region convexity, a photo $p \in P$ can be assigned to sub-region $r \in R$ if and only if p is contained within the boundary defined for r. Geometrically, such constraint is fulfilled when (i) $lng_{\alpha^r} \leq lng_p \leq lng_{\beta^r}$ and (ii) $lat_{\gamma^r} \leq lat_p \leq lat_{\omega^r}$, where the lat stands for the latitude and lng for longitude. To capture this geometrical pattern, for each photo p, the sets $\mathcal{L}^p_\alpha$, $\mathcal{L}^p_\beta$, $\mathcal{L}^p_\gamma$, $\mathcal{L}^p_\omega$ are defined as:

- $\mathcal{L}^p_\alpha = \{c \in C | lng_c \leq lng_p\}$, i.e., $\mathcal{L}^p_\alpha$ contains the longitudes on the left of lng_p,
- $\mathcal{L}^p_\beta = \{c \in C | lng_c \geq lng_p\}$, i.e., $\mathcal{L}^p_\beta$ contains the longitudes on the right of lng_p,
- $\mathcal{L}^p_\gamma = \{\ell \in L | lat_\ell \leq lat_p\}$, i.e., $\mathcal{L}^p_\gamma$ contains the latitudes below lat_p,
- $\mathcal{L}^p_\omega = \{\ell \in L | lat_\ell \geq lat_p\}$, i.e., $\mathcal{L}^p_\omega$ contains the latitudes above lat_p.

Finally, the sub-region convexity is modeled by the *Boundary Constraints* - BC_1, expressed as:

$$(BC_1^\alpha) \qquad y^r_p \leq \sum_{c \in \mathcal{L}^p_\alpha} \alpha^r_c \qquad \forall p \in P, \forall r \in R \tag{4.10}$$

$$(BC_1^\beta) \qquad y^r_p \leq \sum_{c \in \mathcal{L}^p_\beta} \beta^r_c \qquad \forall p \in P, \forall r \in R \tag{4.11}$$

$$(BC_1^\gamma) \qquad y^r_p \leq \sum_{\ell \in \mathcal{L}^p_\gamma} \gamma^r_\ell \qquad \forall p \in P, \forall r \in R \tag{4.12}$$

$$(BC_1^\omega) \qquad y^r_p \leq \sum_{\ell \in \mathcal{L}^p_\omega} \omega^r_\ell \qquad \forall p \in P, \forall r \in R. \tag{4.13}$$

Constraints (4.10) restrict the longitudes which can compose the left border α^r to the left of the photo p's longitude. Similarly, Constraints (4.11)-(4.13) impose restrictions on the right (longitude), the bottom (latitude) and the top (latitude) borders, respectively.

However, a photo p is not assigned to sub-region r if and only if it lies outside the boundary of r, i.e., if $lng_p < lng_{\alpha^r}$, or $lng_p > lng_{\beta^r}$, or $lat_p < lat_{\gamma^r}$, or $lat_p > lat_{\omega^r}$, which may be expressed by either

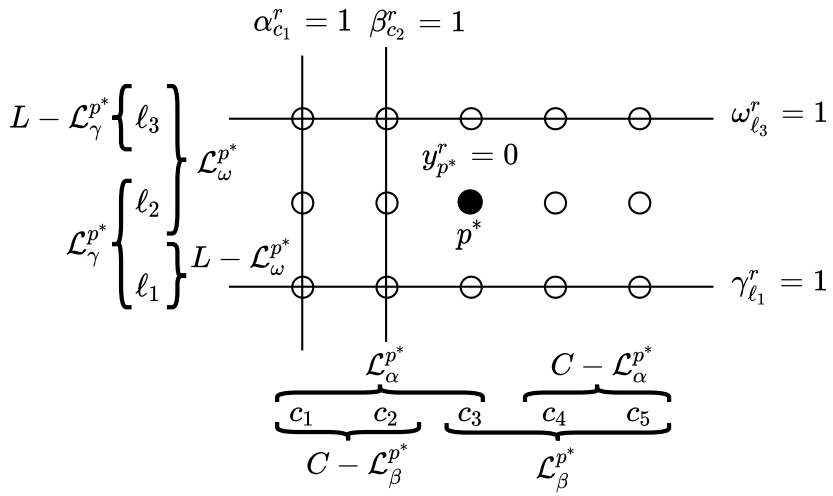
$$(\overline{BC}_0) \qquad \sum_{c \in C - \mathcal{L}^p_\alpha} \alpha^r_c + \sum_{c \in C - \mathcal{L}^p_\beta} \beta^r_c + \sum_{\ell \in L - \mathcal{L}^p_\gamma} \gamma^r_\ell + \sum_{\ell \in L - \mathcal{L}^p_\omega} \omega^r_\ell \geq 1 - y^r_p \quad \forall p \in P, \forall r \in R \tag{4.14}$$

or

$$(BC_0) \qquad \sum_{c \in \mathcal{L}^p_\alpha} \alpha^r_c + \sum_{c \in \mathcal{L}^p_\beta} \beta^r_c + \sum_{\ell \in \mathcal{L}^p_\gamma} \gamma^r_\ell + \sum_{\ell \in \mathcal{L}^p_\omega} \omega^r_\ell \leq 3 + y^r_p \qquad \forall p \in P, \forall r \in R. \tag{4.15}$$

Given constraints (4.6)-(4.9), constraints (4.14) guarantee that at least one of left-side summations is equal to one when p is not assigned to the sub-region r (i.e., $y_p^r = 0$). Consequently, at least one boundary of r makes the photo p to lie outside r. In a complimentary way, constraints (4.15) force that at most three boundaries are satisfied when the photo p is not assigned to the sub-region r. The Fig. 4.5 mathematically illustrates the behaviour of constraints (4.14) and (4.15) concerning a photo p^* (represented by "•") and a sub-region r when the photo p^* is **not** assigned to r.

Moreover, let us define valid inequalities (namely Ordering inequalities) to preemptively remove the infeasible boundaries in the continuous space for any possible sub-region. For instance, a boundary is infeasible if the right border is placed on the left side of the left



$$(\overline{BC}_0)^r_{p^*}: \sum_{c \in C - \mathcal{L}^{p^*}_\alpha} \alpha^r_c + \sum_{c \in C - \mathcal{L}^{p^*}_\beta} \beta^r_c + \sum_{\ell \in L - \mathcal{L}^{p^*}_\gamma} \gamma^r_\ell + \sum_{\ell \in L - \mathcal{L}^{p^*}_\omega} \omega^r_\ell \geq 1 - y^r_{p^*}$$

$$\Rightarrow \quad 0 \quad + \quad 1 \quad + \quad 0 \quad + \quad 0 \quad \geq 1$$

$$(BC_0)^r_{p^*}: \sum_{c \in \mathcal{L}^{p^*}_\alpha} \alpha^r_c + \sum_{c \in \mathcal{L}^{p^*}_\beta} \beta^r_c + \sum_{\ell \in \mathcal{L}^{p^*}_\gamma} \gamma^r_\ell + \sum_{\ell \in \mathcal{L}^{p^*}_\omega} \omega^r_\ell \leq 3 + y^r_{p^*}$$

$$\Rightarrow \quad 1 \quad + \quad 0 \quad + \quad 1 \quad + \quad 1 \quad \leq 3$$

Figure 4.5 Behavior of constraints (4.14) and (4.15) when the photo p_8 is **not** assigned to a sub-region r.

border. Such boundaries are removed through the following set of ordering inequalities:

$$\alpha_c^r \leq \sum_{j \in C: lng_j > lng_c} \beta_j^r \qquad \forall c \in C, \forall r \in R \tag{4.16}$$

$$\beta_c^r \leq \sum_{j \in C: lng_j < lng_c} \alpha_j^r \qquad \forall c \in C, \forall r \in R \tag{4.17}$$

$$\gamma_\ell^r \leq \sum_{j \in L: lat_j > lat_\ell} \omega_j^r \qquad \forall \ell \in L, \forall r \in R \tag{4.18}$$

$$\omega_\ell^r \leq \sum_{j \in L: lat_j < lat_\ell} \gamma_j^r \qquad \forall \ell \in L, \forall r \in R \tag{4.19}$$

Photo transmission constraints

For the purpose of minimizing the 3D processing computation time, it cannot be ignored that an additional delay is introduced any time a picture is transmitted by the drone where it is currently stored, to the drone that is responsible for reconstructing the corresponding sub-region. Given a demand $f^{hd} \in F$, let $\bar{c}^{hd}$ represent the minimum link capacity on the sole routing path of f^{hd} and let z^{hd} be the binary variable equal to 1 if traffic demand $f^{hd} > 0$ is active, i.e., if at least one picture has to be transferred from drone $h \in D$ to drone $d \in \bar{D}$. In this case, non-negative real variables ϕ^{hd} are used to represent the transmission rate achieved by traffic demand f^{hd} on its routing path.

Given the demands $f^{hd} \in F$, the following constraints are introduced to correctly activate binary variables z^{hd}:

$$z^{hd} \leq \sum_{r \in R, p \in P} g_{dp}^r \theta_{hp} \quad \forall (h,d) \in D \times \bar{D}. \tag{4.20}$$

That is, a demand f^{hd} is active if there exists a photo exchange between drones h and d. Besides, the flow variables ϕ are forced to 0 when the corresponding traffic demands are idle, or forced to the upper bound on the transmission rate otherwise. This is represented by constraints:

$$\phi^{hd} \leq \bar{c}^{hd} z^{hd} \quad \forall (h,d) \in D \times \bar{D}. \tag{4.21}$$

As mentioned in Section 4.2, the transmission times of the traffic demands are computed by considering the MMF paradigm for computing the traffic demand transmission rates. A flow (transmission rate) allocation vector is MMF if and only if there is at least one bottleneck link $(i,j) \in A$ on the routing path V^{hd} of each active traffic demand $f^{hd} \in F$ [80]. In this work, we adopt the constraints proposed in [81, 83] to impose the MMF paradigm for all the swarm communications (i.e., the photo transmissions) in pCAPsac. For the sake of simplicity,

the multiplicative parameters introduced in [81, 83] to cope with MMF deviations were all considered equal to 1.

In pCAPsac formulation, a maximum networking/transmission latency of $\hat{T}$ seconds is imposed for each activated traffic demand:

$$\hat{T} \cdot \phi^{hd} \geq \sum_{r \in R, p \in P} g^r_{dp} \theta_{hp} \mu_p \quad \forall (h,d) \in D \times \bar{D}. \tag{4.22}$$

The summation term $\sum_{r \in R, p \in P} g^r_{dp} \theta_{hp} \mu_p$ computes the overall amount of data to be transferred from drone $h \in D$ to the drone $d \in \bar{D}$. Limiting the transmission times means to ensure every drone receives all the photos belonging to the assigned sub-region within a maximum prefixed time set by the domain expert. This constraint can be relaxed by setting $\hat{T}$ to a suitable very high value.

Symmetry breaking constraints

Formulation pCAPsac suffers from symmetry in both photo-to-sub-region (i.e., y^r_p) and sub-region-to-drone (i.e., x^r_d) assignments. It is possible to partially break the symmetry of the sub-region-to-drone assignments. Each sub-region can be assigned to one distinct drone in advance. That is, m variables x^r_d are fixed where the fixed pairs $\{(r_1, d_1), \ldots, (r_m, d_m)\} \in \{R \times \bar{D}\}$ have distinct indexes. Considering Fig. 4.1, the sub-region 1 could be assigned to the drone "+" and the sub-region 2 to the drone "×" for instance. Note that setting variables x^r_d does not affect the variables y^r_p. Consequently, just redundant integer solutions are eliminated.

Complete formulation

Let $T_{\max}$ be the 3D mapping completion time, i.e., the makespan, calculated as the maximum processing time obtained from the swarm of drones. The variable $T_{\max}$ is computed by the group of constraints

$$T_{\max} \geq \sum_{r \in R, p \in P} g^r_{dp} \lambda_p \quad \forall d \in \bar{D} \tag{4.23}$$

where $\sum_{r \in R, p \in P} (g^r_{dp} \lambda_p)$ computes the required time to process all the photos assigned to drone $d \in \bar{D}$.

Finally, the pCAPsac formulation is expressed by the following MILP.

$$\min_{x,y} \quad T_{\max} \tag{4.24}$$

s.t. (4.1) − (4.23)

MMF constraints [81]

$$x_d^r, y_p^r, g_{dp}^r \in \{0,1\} \qquad \forall r \in R, \forall p \in P, \forall d \in \bar{D} \tag{4.25}$$

$$w_{ij}^{hd} \in \{0,1\} \qquad \forall (i,j) \in A, \forall (h,d) \in D \times \bar{D} \tag{4.26}$$

$$\phi^{hd} \geq 0, z^{hd} \in \{0,1\} \qquad \forall (h,d) \in D \times \bar{D} \tag{4.27}$$

$$\alpha_c^r, \beta_c^r \in \{0,1\} \qquad \forall c \in C, \forall r \in R \tag{4.28}$$

$$\gamma_\ell^r, \omega_\ell^r \in \{0,1\} \qquad \forall \ell \in L, \forall r \in R \tag{4.29}$$

$$u_{ij} \geq 0 \qquad \forall (i,j) \in A. \tag{4.30}$$

The objective function (4.24) minimizes the makespan of the whole 3D mapping procedure. The domain constraints are given by (4.25)-(4.30). As $|R| = |\bar{D}|$, the total number of constraints in the model is $O(|P| \cdot |\bar{D}|^2)$ as well as the number of its variables.

4.4 NP-Hardness of the CAPsac

The proof comes from a reduction of the decision version of the *unweighted Geometric Set-Covering Problem - GSCP*, whose objective is to assert, for a finite set of points $P' = \{p_1, p_2, ..., p_n \in \mathbb{R}^d\}$ and a finite collection $\mathcal{S}'$ of subsets of P', if there exists a covering for the points P' composed by at most $k < |\mathcal{S}'|$ sets of $\mathcal{S}'$, i.e., if there exists a $\mathcal{C} \subset \mathcal{S}'$ such that $\cup_{c \in \mathcal{C}} c = P'$ and $|\mathcal{C}| \leq k$. The collection $\mathcal{S}'$ is induced by a fixed polytope $\mathcal{T}$, that is, $\mathcal{S}'$ is formed by the points covered by the distinct placements of $\mathcal{T}$ over the coordinates of the points P'. The decision problem is NP-Complete even when $\mathcal{T}$ is a fixed square [139] or a fixed circumference [140].

Proposition 1. Given an instance I' of the $GSCP$, there exists a polynomial-time transformation from I' to an instance I of the $CAPsac$.

Proof. Consider an instance of the $GSCP$ with P' points, a collection $\mathcal{S}'$ of subsets of P' induced by a fixed square of length s, and a positive integer $k < |\mathcal{S}'|$. The instance of the $rCAPsac$ (see the Appendix) is created on polynomial-time as follows.

Let the set of photos P and their location be equal to the set of points P' (i.e., $P = P'$), and consider a set of k drones which can do the 3D reconstruction, i.e., $|D| = |\bar{D}| = k$. The communication network $T = (N, A)$ is a random tree whose links $(i,j) \in A$ have infinite

capacity. Therefore, the transmission times, the transmission rates ϕ^{hd}, and the photos storage location θ_{dp} can be dismissed. Thus, the latest completion time $T_{\max}$ is defined only by the photo processing times of the drones. Concerning the reliability factor, it is made equal to 1 ($\sigma = 1$). Indeed, the collection $\mathcal{S}'$ does not have all possible rectangular spatial-convex sets, being the $\mathcal{S} \setminus \mathcal{S}'$ missing spatial-convex sets obtained in polynomial-time by inspecting the tuples in $C \times C \times L \times L$. Finally, the photo processing times of the spatial-convex sets will be either 1 or $+\infty$. For $S \in \mathcal{S}'$, $t^S = 1$, while for the remaining $S \in \mathcal{S} \setminus \mathcal{S}'$ $t^S = +\infty$. □

Proposition 2. The *CAPsac* answers the *GSCP*.

Proof. Consider $rCAPsac(P, D, T, \mathcal{S})$ a routine which solves the *CAPsac* by the *rCAPsac* formulation (see the Appendix). Let its optimal solution be comprised by the optimal set Q^* of variables q_d^S and the optimal completion time $T^*_{\max}$. Given a solution of an instance of the *CAPsac* created from an instance of the *GSCP*, evaluating $T^*_{\max}$ is enough to answer the *GSCP*. If $T^*_{\max} = 1$, reply *yes*. Otherwise, reply *no*. For an optimal solution whose $T^*_{\max} = 1$, the covering $\mathcal{C}$ of the P', with $|\mathcal{C}| \leq k$, can be extracted from Q^*. □

Theorem 1. The *CAPsac* is NP-Hard.

Proof. Given the propositions 1 and 2, one can state that the *GSCP* is no harder than the *CAPsac*. Since the *GSCP* is NP-complete, the *CAPsac* is NP-Hard. □

4.5 Computational experiments

Our experimental analysis assessed (i) the effectiveness of the ordering inequalities and the branching strategies for the pCAPsac formulation; (ii) the performance of formulation pCAPsac; (iii) the sensitivity of the pCAPsac formulation with respect to both reliability factor σ and maximum allowed transmission time $\hat{T}$. We used CPLEX v12.8 as general-purpose MILP solver. All experiments were carried exploiting a single core on a machine powered by an Intel E5-2683 v4 Broadwell 2.1GHz with 20Gb of RAM, and running the CentOS Linux 7.5.1804 OS.

We do not present computational experiments for formulation rCAPsac (see the Appendix). As demonstrated in the Appendix, the number of variables of that formulation is bounded by $O(|P|^4)$, which can rapidly increase. Column Generation (CG) strategy has been extensively applied to formulations with a massive number of variables [141]. The great advantage of employing CG is to solve the Linear Programming (LP) continuous relaxation considering only a relevant subset of variables. Such LP with reduced number of variables is called the *restricted master problem* (RMP). The subset of relevant variables is iteratively created as

needed by solving the so-called *pricing subproblem* (PS). Usually, a CG iteration comprises: i) solving the current RMP to obtain current primal optimal solution and its associated dual variables, and ii) optimizing the PS to find new variables with negative reduced costs (when considering minimization problems). The CG terminates when the PS does not find any variable with negative reduced cost, i.e., when the optimality of the current RMP has been proved. Preliminary experiments (restricted to $\sigma = 1$ and $\hat{T} = +\infty$) were performed applying a vanilla column generation on the *rCAPsac*. Unfortunately, *rCAPsac* has proved highly degenerate requiring several CG iterations to prove optimality. Therefore, its performance for solving CAPsac was largely inferior to that of using the pCAPsac formulation. Finally, note that the above degeneracy is a common CG drawback that leads the resulting algorithms (and codes) to be difficult to tune and somehow delicate, thus incompatible for the applied context we deal with.

4.5.1 MILP technology

Whenever a new problem can be described by a compact MILP formulation, it is sensible (if not necessary) to first assess the viability of solving it by a general-purpose MILP solver like CPLEX. The reason is that MILP technology is very mature, with essentially all advances in algorithmic discrete optimization now part of the arsenal of the main solvers [142]. In other words, even the design of an *ad-hoc* exact or heuristic algorithm for a new discrete optimization problem always needs to be accompanied by a thoughtful analysis of a MILP solver performance on a possibly non-trivial formulation on the problem. This is the path we have followed and in this section, we briefly describe the basic ingredients that make the MILP technology so successful and mature.[2]

MILP solvers implement a sophisticated version of the divide-and-conquer algorithm called branch and bound. The basic idea is to relax the complicated constraints of the problem, namely the integrality requirements, and solve the continuous (or LP) relaxation to compute a lower bound (in our minimization case) on the optimal solution value. If none of the removed constraints is violated (in this case, all discrete variables are integer) in the LP solution, then the solution is feasible and the problem is solved. Otherwise, one of the variables taking a fractional value in the continuous relaxation is rounded up and down and two subproblems are created by imposing either that the variable must be smaller or equal than the rounded-down value or greater or equal than the rounded-up value, respectively. This step is called branching and it is easy to see that for both subproblems the solution of the LP relaxation

[2]The interested reader is referred to [142] for more details and for all necessary references that we avoid to add to the section to keep it compact.

is infeasible and that the optimal solution of the original problem must belong to one of the two subproblems. It is easy to visualize the branching process by a tree in which the original MILP is the “root” node and each subproblem is a “child" node. The algorithm iterates by the selection (according to different policies) of one of the subproblems / nodes and a node can be fathomed either because integer feasible, or because LP infeasible or, finally, because the local lower bound is provably worse than the incumbent solution, i.e., the best upper bound value found so far in the search.

This basic scheme clearly leads to potentially explore an exponentially-large search tree, which is to be expected in the worst case (MILP is theoretically NP-hard) but obviously undesirable. The quality of the formulation is measured with respect to the gap between its LP relaxation and the optimal solution values,[3] and MILP technology includes a number of enhancements to this basic scheme to help avoid the tree growing too large. Namely, (i) preprocessing and probing techniques are applied to simplify the formulation by fixing variables and improving coefficients, (ii) redundant linear inequalities exploiting integrality of the variables are added to improve the LP relaxation,[4] (iii) effective rules to select, at every node, the variable to branch on are used and (iv) heuristic methods are applied to compute better and better incumbent solutions (and, consequently, fathom more nodes).

4.5.2 Tested instances and computational settings

The instances, which were constructed from realistic data, comprise two scenarios:

i *Unweighted*: all photos require the same amount of processing time λ.

ii *Weighted*: each photo $p \in P$ requires a certain amount of processing time λ_p.

The λ_p are acquired from the equivalent unweighted case: $\lceil |P| \times 0.1 \rceil$ groups of nine adjacent photos are randomly selected, and then, for each of those groups, a single λ_p is drawn from a normal distribution ($\mu = 26.72$ seconds and $\sigma = 5.0$) and attributed to all photos of that group. Changing the photo-processing time in that fashion allows to represent the 3D reconstruction of distinct complex objects in the region of interest. The name of the instances follows the notation X-PYYDZ%$\bar{D}WW$ where “X” is “u” for the unweighted instances and “w” for the weighted instances, “YY” stands for the number of photos in the instance, “Z” specifies the number of drones in the swarm, and “WW” informs the percentage of drones that can do 3D reconstruction. The number of drones able to perform 3D reconstruction, called

[3] We specify and use the notion of gap extensively in the computational evaluation.

[4] Those inequalities are referred to as “cutting planes” or “cuts” and this is the reason the overall algorithm is often referred to as “branch and cut” instead of “branch and bound".

$|\bar{D}|$, is always equal to $\lfloor Z \times \frac{WW}{100} \rfloor$. The characteristics of the tested instances are listed in Table 4.2. Many factors directly impact the number of photos required to obtain 3D maps with good resolution, e.g., perspective, lens quality, overlap, coverage, and object geometry [87]. In fact, the same number of photos is suitable to areas with different dimensions when varying those parameters [96]. As an illustration, according to [133], an area surrounding a football field of approximate size of 160m $\times$ 80m was reconstructed with about 200 pictures. Hence, the number of photos used in our instances is comparable to those of real-world applications.

The tables presented hereafter report the instance employed at each row (column "Instance"), the corresponding formulation (column "Form."), the dual gap (in percentage) w.r.t. the optimal solution found at the root node (column "gap_0"), the number of cuts added by CPLEX at the root node (column "cuts"), the number of nodes explored by the CPLEX's branch-and-cut method (column "Nodes"), and the dual gap (in percentage) w.r.t. the optimal solution (best known, see below) at the end of the branch-and-cut enumeration (column "*gap*"). CPU times spent in the solution of the root node and by the branch-and-cut algorithm are also reported (column "*sec.*").

Note that dual gaps are computed with respect to the best upper bound solution found whenever the optimal solutions are not obtained by CPLEX within one day of execution. These situations are represented in the tables by the symbol "$*$".

4.5.3 pCAPsac experiments

This section evaluates the performance of the proposed pCAPsac formulation. It investigates the effectiveness of valid inequalities (4.16)-(4.19), whose aim is, for any possible sub-region, to clean the search space from all the infeasible boundary configuration. Also, the experiments analyze how different branching priorities can influence the branch-and-cut method. All the experiments of this section are for $\sigma = 1$ and $\hat{T}$=$+\infty$.

This section also reports the performance profiles [143] concerning the CPU times of the formulations. Let us define I as the set of all instances and $\mathcal{F}$ as the set comprising all formulations to be compared. Given an instance $i \in I$, the performance profile compares the CPU time of a formulation $form \in \mathcal{F}$, denoted $cpu_{i,form}$, with the best CPU time obtained across all formulations in $\mathcal{F}$ when solving that instance i. It is done by computing the performance ratio $r_{i,form}$ [143]

$$r_{i,form} = \frac{cpu_{i,form}}{\min\limits_{form \in \mathcal{F}} \{cpu_{i,form}\}} \quad \forall i \in I, form \in \mathcal{F}.$$

Table 4.2 Characteristics of the tested instances.

Photos(P)	200, 400
Drones(D)	5, 7, 10
%3D-capable drones(%$\bar{D}$)	50%, 70%, 90%

Thus, the overall assessment of performance is captured by the cumulative distribution function

$$\rho_{form}(\tau) = \frac{|\{r_{i,form} \leq \tau | i \in I\}|}{|I|}.$$

Finally, $\rho_{form}(\tau)$ is the probability for the formulation $form$ that the ratio $r_{i,form}$ is at most τ of the best ratio found [143].

pCAPsac performance

The pCAPsac formulation using the $\overline{BC_0}$ constraints (4.14), called "PB:$\overline{BC_0}$", and the pCAPsac formulation using the BC_0 constraints (4.15), named "PB:BC_0", are compared in Tables 4.3 and 4.4. For those tables, the dual gaps in the column "gap_0" indicate how strong is the formulation. Small dual gaps mean that the value of the optimal objective function is close to the value of the optimal objective function obtained when solving the continuous relaxation of the formulation. Small "gap_0" values lead to the exploration of fewer nodes by the branch-and-cut method, reducing its overall computing time. As the dual gaps "gap_0" are equal across the proposed formulations, one should focus on the execution time (column "sec") and number of nodes explored (column "nodes") of the whole branch-and-cut method.[5] The results related to lines "PB:$\overline{BC_0}$+Ord." are discussed in the following section.

[5]Note that the time and nodes can vary even if the gap is initially equal because CPLEX evolves differently, for example, because it adds different cutting planes (see, column "cuts").

Table 4.3 CPLEX results when solving unweighted instances for the "$PB{:}\overline{BC_0}$", the "$PB{:}BC_0$", and "$PB{:}\overline{BC_0}$+$Ord.$" formulations.

Instance	Form.	Root Node			Branch-and-Cut		
		$\mathbf{gap_0}$	**cuts**	**sec.**	**Nodes**	**gap**	**sec.**
u-P200D5%$\bar{D}$70	$PB{:}\overline{BC}_0$	4.76	28	1.47	182	0.00	15.81
	$PB{:}BC_0$	4.76	110	2.84	300	0.00	33.83
	$PB{:}\overline{BC}_0$+Ord.	4.76	242	1.15	345	0.00	19.02
u-P200D7%$\bar{D}$50	$PB{:}\overline{BC}_0$	4.76	17	1.25	395	0.00	27.48
	$PB{:}BC_0$	4.76	62	1.14	383	0.00	19.70
	$PB{:}\overline{BC}_0$+Ord.	4.76	189	1.46	239	0.00	11.45
u-P400D5%$\bar{D}$70	$PB{:}\overline{BC}_0$	1.23	28	3.33	386	0.00	87.16
	$PB{:}BC_0$	1.23	74	3.57	264	0.00	57.94
	$PB{:}\overline{BC}_0$+Ord.	1.23	183	3.38	500	0.00	91.56
u-P400D7%$\bar{D}$50	$PB{:}\overline{BC}_0$	1.23	27	3.42	556	0.00	156.15
	$PB{:}BC_0$	1.23	50	3.16	1178	0.00	181.14
	$PB{:}\overline{BC}_0$+Ord.	1.23	188	4.31	707	0.00	133.70
u-P200D5%$\bar{D}$90	$PB{:}\overline{BC}_0$	0.00	9	2.16	433	0.00	24.73
	$PB{:}BC_0$	0.00	31	3.21	41	0.00	13.98
	$PB{:}\overline{BC}_0$+Ord.	0.00	227	1.99	359	0.00	22.60
u-P200D7%$\bar{D}$70	$PB{:}\overline{BC}_0$	0.00	23	2.14	371	0.00	27.49
	$PB{:}BC_0$	0.00	22	4.07	175	0.00	24.74
	$PB{:}\overline{BC}_0$+Ord.	0.00	226	2.33	699	0.00	45.39
u-P400D5%$\bar{D}$90	$PB{:}\overline{BC}_0$	0.00	4	5.94	631	0.00	440.53
	$PB{:}BC_0$	0.00	81	4.09	121	0.00	44.60
	$PB{:}\overline{BC}_0$+Ord.	0.00	267	5.00	49	0.00	35.02
u-P400D7%$\bar{D}$70	$PB{:}\overline{BC}_0$	0.00	71	5.04	790	0.00	141.43
	$PB{:}BC_0$	0.00	84	5.06	1631	0.00	696.21
	$PB{:}\overline{BC}_0$+Ord.	0.00	175	6.23	7011	0.00	6665.71
u-P200D10%$\bar{D}$50	$PB{:}\overline{BC}_0$	0.00	81	3.63	6591	0.00	1036.81
	$PB{:}BC_0$	0.00	95	2.72	18840	0.00	3141.16
	$PB{:}\overline{BC}_0$+Ord.	0.00	312	4.99	3546	0.00	556.97
u-P400D10%$\bar{D}$50	$PB{:}\overline{BC}_0$	0.00	38	9.99	4363	0.00	1737.28
	$PB{:}BC_0$	0.00	127	10.64	16216	0.00	7200.00
	$PB{:}\overline{BC}_0$+Ord.	0.00	46	6.35	990	0.00	219.40
u-P200D7%$\bar{D}$90	$PB{:}\overline{BC}_0$	1.96	24	5.17	8448	0.00	1354.41
	$PB{:}BC_0$	1.96	47	9.60	13835	1.96	7200.00
	$PB{:}\overline{BC}_0$+Ord.	1.96	130	8.80	521	0.00	49.52
u-P400D7%$\bar{D}$90	$PB{:}\overline{BC}_0$	*1.96	130	12.28	12362	*1.96	7200.00
	$PB{:}BC_0$	*1.96	84	13.56	9063	*1.96	7200.00
	$PB{:}\overline{BC}_0$+Ord.	*1.96	50	17.94	14897	*1.96	7200.00

Table 4.4 CPLEX results when solving weighted instances for the "$PB{:}\overline{BC_0}$", the "$PB{:}BC_0$", "$PB{:}\overline{BC_0}$+*Ord.*" formulations.

Instance	Form.	Root Node			Branch-and-Cut		
		gap_0	cuts	sec.	Nodes	gap	sec.
w-P200D5%$\bar{D}$70	$PB{:}\overline{BC_0}$	3.36	28	1.40	226	0.00	19.13
	$PB{:}BC_0$	3.36	19	2.14	450	0.00	45.84
	$PB{:}\overline{BC_0}$+Ord.	3.36	93	1.25	300	0.00	16.87
w-P200D7%$\bar{D}$50	$PB{:}\overline{BC_0}$	3.67	22	1.34	961	0.00	50.73
	$PB{:}BC_0$	3.67	40	1.27	1057	0.00	65.62
	$PB{:}\overline{BC_0}$+Ord.	3.67	111	1.49	549	0.00	31.52
w-P400D5%$\bar{D}$70	$PB{:}\overline{BC_0}$	0.86	9	2.57	483	0.00	97.08
	$PB{:}BC_0$	0.86	45	3.19	910	0.00	148.94
	$PB{:}\overline{BC_0}$+Ord.	0.86	190	3.48	414	0.00	79.99
w-P400D7%$\bar{D}$50	$PB{:}\overline{BC_0}$	2.02	44	2.92	1008	0.00	191.33
	$PB{:}BC_0$	2.02	22	3.47	930	0.00	121.78
	$PB{:}\overline{BC_0}$+Ord.	2.02	264	3.40	974	0.00	241.08
w-P200D5%$\bar{D}$90	$PB{:}\overline{BC_0}$	2.96	13	2.37	5080	0.00	748.98
	$PB{:}BC_0$	2.96	42	4.76	24186	0.00	5199.65
	$PB{:}\overline{BC_0}$+Ord.	2.96	84	2.30	3427	0.00	427.78
w-P200D7%$\bar{D}$70	$PB{:}\overline{BC_0}$	2.69	58	2.55	4227	0.00	453.18
	$PB{:}BC_0$	2.69	21	3.95	18204	0.00	4191.87
	$PB{:}\overline{BC_0}$+Ord.	2.69	87	3.01	3736	0.00	514.63
w-P400D5%$\bar{D}$90	$PB{:}\overline{BC_0}$	1.26	19	4.85	3161	0.00	1425.80
	$PB{:}BC_0$	1.26	51	5.29	18195	1.24	7200.00
	$PB{:}\overline{BC_0}$+Ord.	1.26	442	6.14	3048	0.00	1532.36
w-P400D7%$\bar{D}$70	$PB{:}\overline{BC_0}$	0.68	34	4.97	4529	0.00	1805.83
	$PB{:}BC_0$	0.68	13	4.63	19444	0.68	7200.00
	$PB{:}\overline{BC_0}$+Ord.	0.68	628	6.68	9429	0.00	6031.09
w-P200D10%$\bar{D}$50	$PB{:}\overline{BC_0}$	1.65	15	5.42	32327	1.65	7200.00
	$PB{:}BC_0$	1.65	63	3.62	42035	1.65	7200.00
	$PB{:}\overline{BC_0}$+Ord.	1.65	80	5.55	19491	1.65	7200.00
w-P400D10%$\bar{D}$50	$PB{:}\overline{BC_0}$	*3.38	129	10.59	9416	*3.38	7200.00
	$PB{:}BC_0$	*3.38	82	8.88	7364	*3.38	7200.00
	$PB{:}\overline{BC_0}$+Ord.	*3.38	194	7.03	10625	*3.38	7200.00
w-P200D7%$\bar{D}$90	$PB{:}\overline{BC_0}$	*3.81	1	8.28	54078	*3.81	7200.00
	$PB{:}BC_0$	*3.81	71	6.17	19819	*3.81	7200.00
	$PB{:}\overline{BC_0}$+Ord.	*3.81	57	10.48	23206	*3.81	7200.00
w-P400D7%$\bar{D}$90	$PB{:}\overline{BC_0}$	*3.18	15	9.42	13316	*3.18	7200.00
	$PB{:}BC_0$	*3.18	83	17.06	10581	*3.18	7200.00
	$PB{:}\overline{BC_0}$+Ord.	*3.18	226	13.09	9753	*3.18	7200.00

The results in both Tables 4.3 and 4.4 clearly show that the "PB:$\overline{BC_0}$" solves faster than "PB:BC_0" formulation (T=-2.877 and p-val=0.008 via "PB:$\overline{BC_0}$" vs. "PB:BC_0" paired t-test [144]). The performance profile in Fig. 4.6 confirms the better performance of the "PB:$\overline{BC_0}$". In fact, "PB:$\overline{BC_0}$" has the largest probability (0.75) to have the best performance ratio (point when $\tau = 1$).

We can observe that the dual gaps at the root node are equal to zero for the unweighted instances whenever the number of photos is divisible by the number of drones that can do the 3D reconstruction. Consequently, the objective function value of the optimum solution coincides with the dual bound already at the root node for instances "u-P200D5%$\bar{D}$90", "u-P200D7%$\bar{D}$70", "u-P400D5%$\bar{D}$90", "u-P400D7%$\bar{D}$70", "u-P200D10%$\bar{D}$50" and "u-P400D10 %$\bar{D}$50".

Finally, based on tables 4.3 and 4.4, Figs. 4.7, 4.8, 4.9, and 4.10 present how the CPU time (column "sec.") is affected (on average) with the increase of the number of 3D-capable drones in the unweighted and weighted instances. Figs. 4.7, 4.8, 4.9, and 4.10 group the average CPU times — computed w.r.t. the collection of instances with the same number of 3D-capable drones and number of photos — obtained by formulations "PB:BC_0" (green lines), "PB:$\overline{BC_0}$" (blue dashed lines), and "PB:$\overline{BC_0}$+$Ord.$" (orange dotted lines). We observe that the average CPU time increases when the instances have more 3D-capable drones. Also, the average CPU time of the instances with 200 photos tends to be shorter than the average

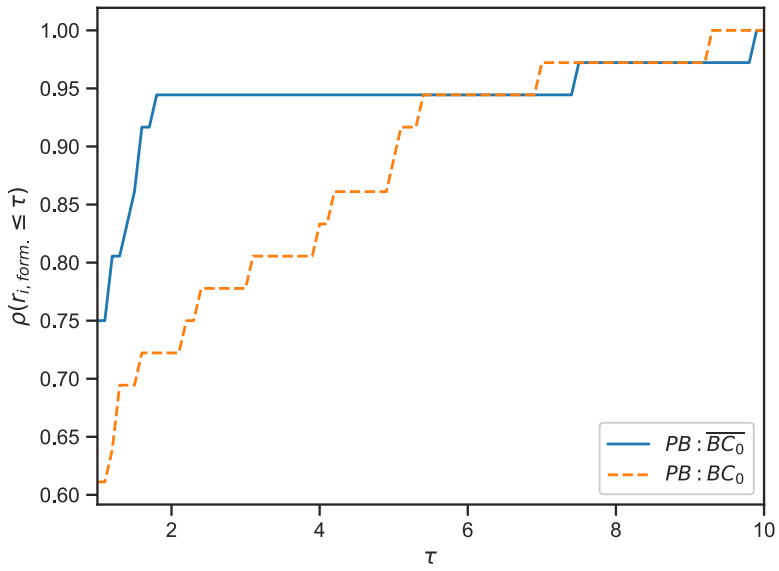


Figure 4.6 Performance profile w.r.t. the CPU times of the "PB:$\overline{BC_0}$" and "PB:BC_0".

CPU time of the instances with 400 photos.

Ordering inequalities effectiveness

The effect of adding all the ordering inequalities (4.16)-(4.19) into the "$PB{:}\overline{BC_0}$" formulation is analyzed in Tables 4.3 and 4.4. The inclusion of the ordering inequalities is identified by the "$+Ord.$" in the formulation's name.

The valid inequalities (4.16)-(4.19) eliminate infeasible boundaries in the continuous solution space whereas not necessarily excluding the continuous optimum solution. Consequently, these inequalities are not guaranteed to increase the dual bound obtained. In fact, gap_0 was never improved in our experiments after adding the ordering inequalities. Nevertheless, the insertion of (4.16)-(4.19) improved the CPU time required to reach the optimum solution of 11 out to 19 instances solved to optimality (considering 2h of execution). In fact, paired t-tests show significant improvements (T= 1.907 p-val= 0.034) by adding them into the "$PB{:}\overline{BC_0}$" formulation, except for instances "P400D7%$\bar{D}$70". This is also confirmed by the performance profile in Fig. 4.11 in which "$PB{:}\overline{BC_0}{+}Ord.$" has the best CPU times approximately 75% of the times ($\tau = 1$). However, when limited to weighted cases, there is no significant improvement on reducing the enumeration CPU time. For these cases, 11 seconds of improvement is obtained when comparing the average computing CPU time of non-inserting against inserting constraints (4.16)-(4.19). Finally, we observed that the number of cuts added by CPLEX at the root node increased considerably when the ordering inequalities were employed.

Branching priority

Different branching priorities for the selection of the boundary assignment (i.e., α_c^r, β_c^r, γ_ℓ^r, and ω_ℓ^r) and the photo assignment (i.e., y_p^r) variables were also explored in formulation "$PB{:}\overline{BC_0}{+}Ord.$" (simply denoted "$PB{:}\overline{BC_0}$" at this section) and the results reported in Tables 4.5 and 4.6. The distinct branching priorities are denoted as "$b{>}y$" and "$y{>}b$". The default option of CPLEX is identified by the absence of those notations. The "$b{>}y$" is used when the boundary assignment variables are given higher priority over the photo assignment variables, which still have higher priority over the remaining variables. The "$y{>}b$" refers to the opposite case, that is, when photo assignment variables are rather branched over boundary assignment variables. Since the values in the column "gap" are equal in most of the cases, the number of nodes explored (column "nodes") and execution times (column "gap") should be used as the comparison metric. This means that smaller values in those columns stand for more efficient enumeration performed by CPLEX.

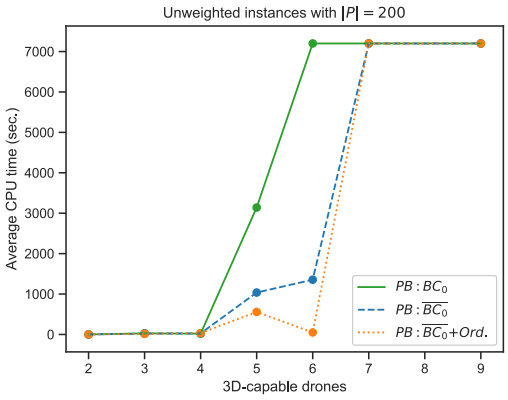


Figure 4.7 Average CPU times when solving unweighted instances with 200 photos according to "PB:BC_0", "PB:$\overline{BC_0}$", and "PB:$\overline{BC_0}$+$Ord.$".

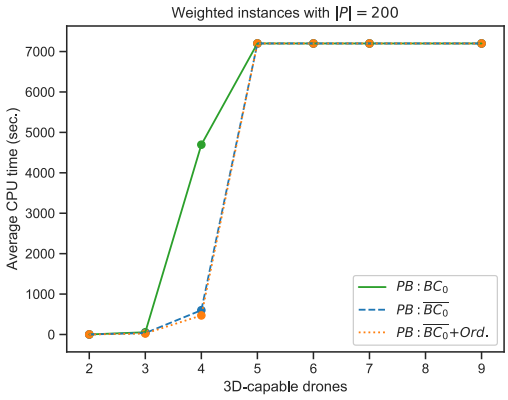


Figure 4.8 Average CPU times when solving weighted instances with 200 photos according to "PB:BC_0", "PB:$\overline{BC_0}$", and "PB:$\overline{BC_0}$+$Ord.$".

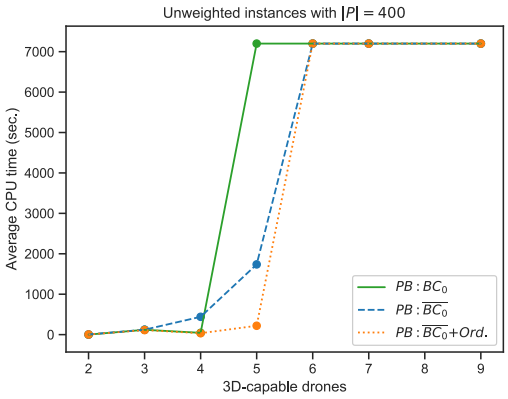


Figure 4.9 Average CPU times when solving unweighted instances with 400 photos according to "PB:BC_0", "PB:$\overline{BC_0}$", and "PB:$\overline{BC_0}$+$Ord.$".

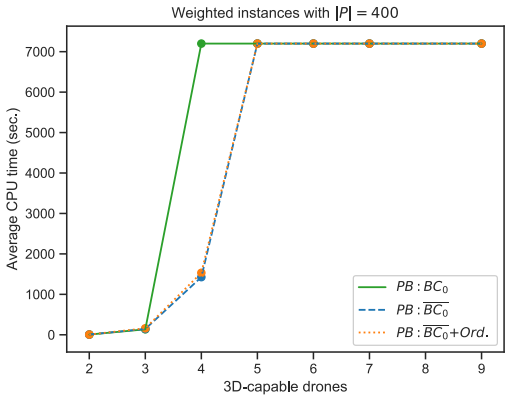


Figure 4.10 Average CPU times when solving weighted instances with 400 photos according to "PB:BC_0", "PB:$\overline{BC_0}$", and "PB:$\overline{BC_0}$+$Ord.$".

The adoption of different branching priorities improves the CPU times to solve some instances. In fact, the "y>b" strategy achieves better or equivalent CPU times in 10 out to 11 unweighted instances solved to optimality (within 2h of execution). Regarding weighted cases, the "b>y" strategy results in better or equivalent CPU times in 5 out to 8 instances solved to optimality (considering 2h of execution). However, paired t-tests do not show significant improvement for both "b>y" (T= 0.889 p-val= 0.191) or "y>b" (T=1.295 p-val= 0.103) strategies considering overall cases.

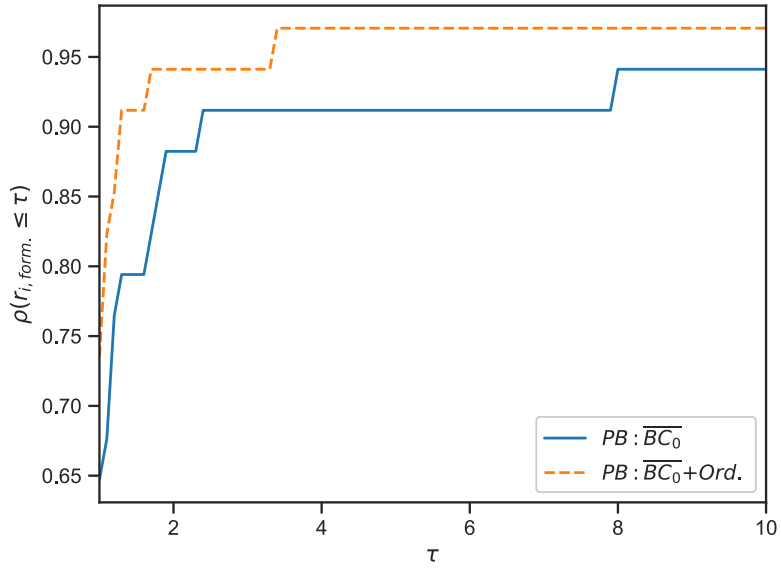


Figure 4.11 Performance profile w.r.t. the CPU times of the "PB:$\overline{BC_0}$" and "PB:$\overline{BC_0}$+$Ord.$".

4.5.4 Sensitivity of the pCAPsac formulation

This section evaluates the sensitivity of pCAPsac formulation with respect to both reliability factor σ and to maximum transmission time allowed $\hat{T}$.

Reliability factor sensitivity

Tables 4.7 and 4.8 report results for various values of σ, ranging from 1 to $|\bar{D}| - 1$. The subset of instances used in this experiment consists of those for which CPLEX was able to solve within 2h of execution the associated problem with $\sigma = 1$. Besides, the communication constraints concerning $\hat{T}$ were relaxed, i.e. $\hat{T}$=$+\infty$. For this analysis, one should concentrate on how much varying σ affects the initial dual gap (column "gap_0"), which might lead to a large number of explored nodes and execution times.

Tables 4.7 and 4.8 show that initial dual gaps are largely affected by σ. Those large dual gaps result from the increase of the optimum solution values with σ not accompanied by the increase in the dual bounds obtained at the root node. This is illustrated in Fig. 4.12 and Fig. 4.13, where the changes in the optimal objective function value (named "$T_{\max}$") given the increase of σ are presented for unweighted instances "u-P200D7%$\bar{D}$70", "u-P200D10%$\bar{D}$50", "u-P200D7%$\bar{D}$90", and for weighted instances "w-P200D7%$\bar{D}$70", "w-P200D10%$\bar{D}$50", "w-P200D7%$\bar{D}$90". Those instances were selected to include distinct values of $|\bar{D}|$.

Table 4.5 CPLEX results when solving unweighted instances for the "$PB{:}\overline{BC_0}$", "$PB{:}\overline{BC_0}{-}b{>}y$" and "$PB{:}\overline{BC_0}{-}y{>}b$" branching priority strategies.

Instance	Form.	Branch-and-Cut		
		Nodes	gap	sec.
u-P200D5%$\bar{D}$70	$PB{:}\overline{BC_0}$	345	0.00	19.02
	$PB{:}\overline{BC_0}{-}b{>}y$	529	0.00	25.46
	$PB{:}\overline{BC_0}{-}y{>}b$	178	0.00	15.05
u-P200D7%$\bar{D}$50	$PB{:}\overline{BC_0}$	239	0.00	11.45
	$PB{:}\overline{BC_0}{-}b{>}y$	744	0.00	33.73
	$PB{:}\overline{BC_0}{-}y{>}b$	232	0.00	11.47
u-P400D5%$\bar{D}$70	$PB{:}\overline{BC_0}$	500	0.00	91.56
	$PB{:}\overline{BC_0}{-}b{>}y$	1993	0.00	255.79
	$PB{:}\overline{BC_0}{-}y{>}b$	179	0.00	50.86
u-P400D7%$\bar{D}$50	$PB{:}\overline{BC_0}$	707	0.00	133.70
	$PB{:}\overline{BC_0}{-}b{>}y$	1011	0.00	207.03
	$PB{:}\overline{BC_0}{-}y{>}b$	202	0.00	68.28
u-P200D5%$\bar{D}$90	$PB{:}\overline{BC_0}$	359	0.00	22.60
	$PB{:}\overline{BC_0}{-}b{>}y$	471	0.00	41.63
	$PB{:}\overline{BC_0}{-}y{>}b$	72	0.00	7.06
u-P200D7%$\bar{D}$70	$PB{:}\overline{BC_0}$	699	0.00	45.39
	$PB{:}\overline{BC_0}{-}b{>}y$	1096	0.00	110.24
	$PB{:}\overline{BC_0}{-}y{>}b$	393	0.00	25.89
u-P400D5%$\bar{D}$90	$PB{:}\overline{BC_0}$	49	0.00	35.02
	$PB{:}\overline{BC_0}{-}b{>}y$	812	0.00	215.49
	$PB{:}\overline{BC_0}{-}y{>}b$	49	0.00	35.16
u-P400D7%$\bar{D}$70	$PB{:}\overline{BC_0}$	7011	0.00	6665.71
	$PB{:}\overline{BC_0}{-}b{>}y$	1330	0.00	235.25
	$PB{:}\overline{BC_0}{-}y{>}b$	430	0.00	59.71
u-P200D10%$\bar{D}$50	$PB{:}\overline{BC_0}$	3546	0.00	556.97
	$PB{:}\overline{BC_0}{-}b{>}y$	2231	0.00	217.96
	$PB{:}\overline{BC_0}{-}y{>}b$	1328	0.00	144.06
u-P400D10%$\bar{D}$50	$PB{:}\overline{BC_0}$	990	0.00	219.40
	$PB{:}\overline{BC_0}{-}b{>}y$	1022	0.00	107.68
	$PB{:}\overline{BC_0}{-}y{>}b$	922	0.00	199.24
u-P200D7%$\bar{D}$90	$PB{:}\overline{BC_0}$	521	0.00	49.52
	$PB{:}\overline{BC_0}{-}b{>}y$	18419	0.00	3891.44
	$PB{:}\overline{BC_0}{-}y{>}b$	2466	0.00	557.22
u-P400D7%$\bar{D}$90	$PB{:}\overline{BC_0}$	14897	*1.96	7200.00
	$PB{:}\overline{BC_0}{-}b{>}y$	30016	*1.96	7200.00
	$PB{:}\overline{BC_0}{-}y{>}b$	7749	*1.96	7200.00

Maximum transmission time sensitivity

The sensitivity analysis of formulation "PB" to parameter $\hat{T}$ is performed by decreasing its values progressively (the value of σ is fixed to 1 in this set of experiments). The first value of $\hat{T}$ tested corresponds to the allocated communication time between the drones when no time limit is imposed for their communication, i.e., $\hat{T}{=}{+}\infty$. From that value, $\hat{T}$ is decreased

Table 4.6 CPLEX results when solving weighted instances for the "$PB{:}\overline{BC_0}$", "$PB{:}\overline{BC_0} - b{>}y$" and "$PB{:}\overline{BC_0} - y{>}b$" branching priority strategies.

Instance	**Form.**	**Branch-and-Cut**		
		Nodes	**gap**	**sec.**
w-P200D5%$\bar{D}$70	$PB{:}\overline{BC_0}$	300	0.00	16.87
	$PB{:}\overline{BC_0}{-}b{>}y$	620	0.00	38.68
	$PB{:}\overline{BC_0}{-}y{>}b$	279	0.00	20.40
w-P200D7%$\bar{D}$50	$PB{:}\overline{BC_0}$	549	0.00	31.52
	$PB{:}\overline{BC_0}{-}b{>}y$	600	0.00	28.76
	$PB{:}\overline{BC_0}{-}y{>}b$	327	0.00	29.57
w-P400D5%$\bar{D}$70	$PB{:}\overline{BC_0}$	414	0.00	79.99
	$PB{:}\overline{BC_0}{-}b{>}y$	1918	0.00	330.83
	$PB{:}\overline{BC_0}{-}y{>}b$	299	0.00	80.89
w-P400D7%$\bar{D}$50	$PB{:}\overline{BC_0}$	974	0.00	241.08
	$PB{:}\overline{BC_0}{-}b{>}y$	602	0.00	94.42
	$PB{:}\overline{BC_0}{-}y{>}b$	315	0.00	126.37
w-P200D5%$\bar{D}$90	$PB{:}\overline{BC_0}$	3427	0.00	427.78
	$PB{:}\overline{BC_0}{-}b{>}y$	4281	0.00	409.38
	$PB{:}\overline{BC_0}{-}y{>}b$	3833	0.00	560.64
w-P200D7%$\bar{D}$70	$PB{:}\overline{BC_0}$	3736	0.00	514.63
	$PB{:}\overline{BC_0}{-}b{>}y$	5879	0.00	663.39
	$PB{:}\overline{BC_0}{-}y{>}b$	4680	0.00	679.89
w-P400D5%$\bar{D}$90	$PB{:}\overline{BC_0}$	3048	0.00	1532.36
	$PB{:}\overline{BC_0}{-}b{>}y$	2189	0.00	925.85
	$PB{:}\overline{BC_0}{-}y{>}b$	3146	0.00	2052.07
w-P400D7%$\bar{D}$70	$PB{:}\overline{BC_0}$	9429	0.00	6031.09
	$PB{:}\overline{BC_0}{-}b{>}y$	1397	0.00	605.48
	$PB{:}\overline{BC_0}{-}y{>}b$	3341	0.00	2097.93
w-P200D10%$\bar{D}$50	$PB{:}\overline{BC_0}$	19491	1.65	7200.00
	$PB{:}\overline{BC_0}{-}b{>}y$	46502	1.08	7200.00
	$PB{:}\overline{BC_0}{-}y{>}b$	25588	1.65	7200.00
w-P400D10%$\bar{D}$50	$PB{:}\overline{BC_0}$	10625	*3.38	7200.00
	$PB{:}\overline{BC_0}{-}b{>}y$	10981	*3.38	7200.00
	$PB{:}\overline{BC_0}{-}y{>}b$	8520	*3.38	7200.00
w-P200D7%$\bar{D}$90	$PB{:}\overline{BC_0}$	23206	*3.81	7200.00
	$PB{:}\overline{BC_0}{-}b{>}y$	33427	*3.81	7200.00
	$PB{:}\overline{BC_0}{-}y{>}b$	29215	*3.81	7200.00
w-P400D7%$\bar{D}$90	$PB{:}\overline{BC_0}$	9753	*3.18	7200.00
	$PB{:}\overline{BC_0}{-}b{>}y$	20758	*3.18	7200.00
	$PB{:}\overline{BC_0}{-}y{>}b$	9417	*3.18	7200.00

by 0.5 seconds until formulation "PB" becomes infeasible. Fig. 4.14 and Fig. 4.15 present results for instances "u-P200D5%$\bar{D}$70" and "w-P200D5%$\bar{D}$70". For them, the first value of $\hat{T}$ is 48 seconds, being decreased down to 23.5 when both problems become infeasible. The reported number of nodes explored by CPLEX branch-and-cut method and the optimum objective function value are obtained within 2h of computing time.

Table 4.7 CPLEX results when solving unweighted instances for the PB formulation with $\sigma \in \{1, \ldots, |\bar{D}| - 1\}$.

Instance	σ	**Root Node**			**Branch-and-Cut**		
		gap$_0$	**cuts**	**sec.**	**Nodes**	**gap**	**sec.**
u-P200D5%$\bar{D}$70	1	4.76	242	1.15	345	0.00	19.02
	2	50.62	752	10.21	1113	0.00	138.18
u-P200D7%$\bar{D}$50	1	4.76	189	1.46	239	0.00	11.45
	2	50.62	1053	9.16	902	0.00	101.71
u-P400D5%$\bar{D}$70	1	1.23	183	3.38	500	0.00	91.56
	2	50.25	667	2.90	1912	0.00	539.52
u-P400D7%$\bar{D}$50	1	1.23	188	4.31	707	0.00	133.70
	2	50.25	366	2.80	3070	0.00	1723.51
u-P200D5%$\bar{D}$90	1	0.00	227	1.99	359	0.00	22.60
	2	50.00	228	3.47	14578	0.00	5685.71
	3	66.67	194	2.46	2392	0.00	1014.65
u-P200D7%$\bar{D}$70	1	0.00	226	2.33	699	0.00	45.39
	2	*50.00	119	3.71	16579	*33.33	7200.00
	3	66.67	402	3.52	2209	0.00	1150.97
u-P400D5%$\bar{D}$90	1	0.00	267	5.00	49	0.00	35.02
	2	*50.00	349	6.82	3573	*33.33	7200.00
	3	66.67	656	5.65	2631	0.00	4593.69
u-P400D7%$\bar{D}$70	1	0.00	175	6.23	7011	0.00	6665.71
	2	*50.00	403	6.59	3019	*49.50	7200.00
	3	*66.67	749	7.69	3275	*11.11	7200.00
u-P200D10%$\bar{D}$50	1	0.00	312	4.99	3546	0.00	556.97
	2	50.00	522	8.13	11659	16.67	7200.00
	3	*67.48	202	9.37	9759	*59.32	7200.00
	4	75.00	2419	967.58	1145	26.00	7200.00

The decreasing of $\hat{T}$ tends to reduce the number of nodes explored in the branch-and-cut enumeration whereas the objective function value increases until the problem becomes infeasible. For example, in Fig. 4.14, the optimum objective function value increases from 1870.4 to 2939.2 starting at $\hat{T} = 33.5$. The problem becomes infeasible for $\hat{T}$ smaller than 24s.

Table 4.8 CPLEX results when solving weighted instances for the PB formulation with $\sigma \in \{1, \ldots, |\bar{D}| - 1\}$.

Instance	σ	Root Node			Branch-and-Cut		
		$\mathbf{gap_0}$	**cuts**	**sec.**	**Nodes**	**gap**	**sec.**
w-P200D5%$\bar{D}$70	1	3.36	93	1.25	300	0.00	16.87
	2	50.71	831	9.81	1663	0.00	172.46
w-P200D7%$\bar{D}$50	1	3.67	111	1.49	549	0.00	31.52
	2	51.44	670	11.53	4013	0.00	665.60
w-P400D5%$\bar{D}$70	1	0.86	190	3.48	414	0.00	79.99
	2	50.34	480	2.74	1652	0.00	891.45
w-P400D7%$\bar{D}$50	1	2.02	264	3.40	974	0.00	241.08
	2	50.29	566	3.15	2819	0.00	1557.25
w-P200D5%$\bar{D}$90	1	2.96	84	2.30	3427	0.00	427.78
	2	50.00	293	3.73	12045	32.56	7200.00
	3	*67.03	289	4.70	20671	*1.09	7200.00
w-P200D7%$\bar{D}$70	1	2.69	87	3.01	3736	0.00	514.63
	2	50.00	428	3.89	6632	0.00	2647.32
	3	*66.82	382	3.03	25358	*0.46	7200.00
w-P400D5%$\bar{D}$90	1	1.26	442	6.14	3048	0.00	1532.36
	2	*50.01	381	4.95	3415	*48.87	7200.00
	3	*66.86	430	10.48	2994	*11.63	7200.00
w-P400D7%$\bar{D}$70	1	0.68	628	6.68	9429	0.00	6031.09
	2	*50.00	387	7.68	2605	*49.52	7200.00
	3	*66.78	650	6.11	2089	*33.02	7200.00

4.6 Conclusion

A swarm of drones (UAVs) can be used to automate a wide range of missions, from surveillance to search and rescue, from 3D mapping to telecommunication enhancement. While UAVs are typically responsible for the mission phases related to data collection — thanks to their flying capabilities and to the availability of embedded sensors — most of the data processing is offloaded to dedicated machines (virtual or bare-metal) placed in the cloud. However, when the communication bandwidth between the swarm and the cloud is limited, an ad-hoc cloud established on top of the UAVs' computing resources (and those of other

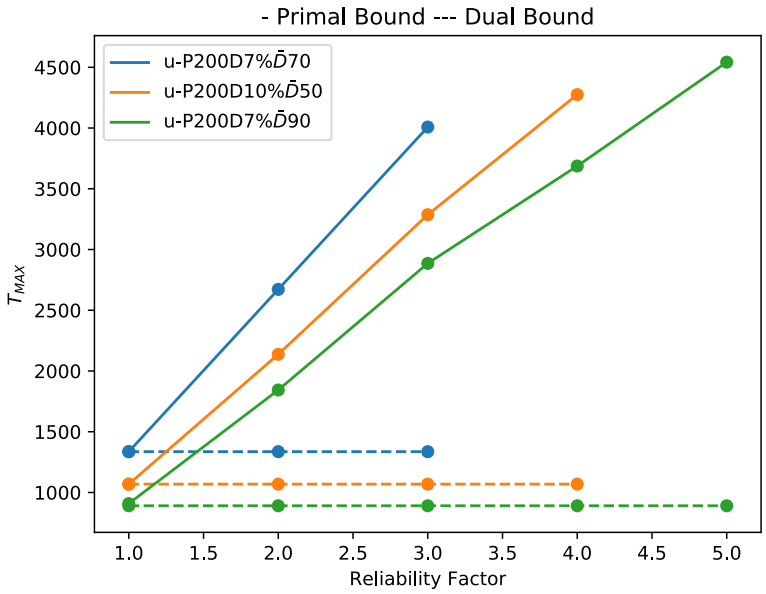


Figure 4.12 *Primal bound* and *dual bound* on increasing σ for unweighted instances.

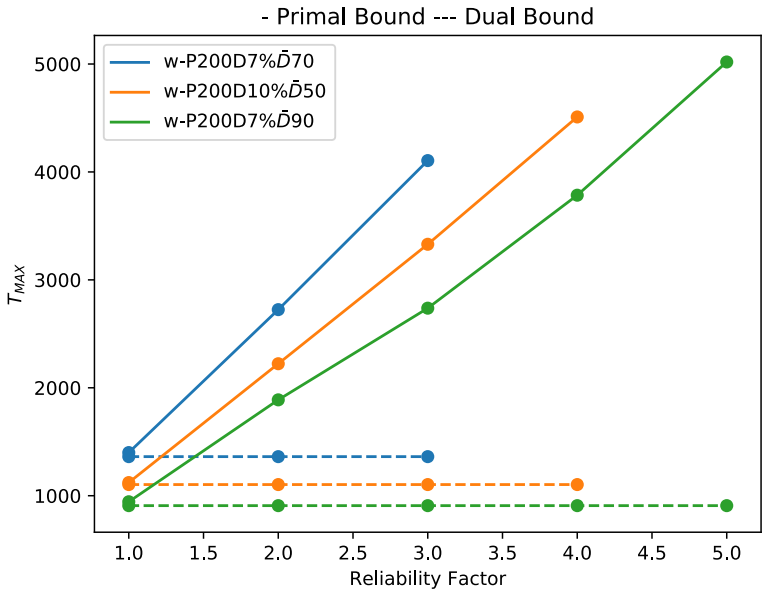


Figure 4.13 *Primal bound* and *dual bound* on increasing σ for weighted instances.

elements available in the area) can be leveraged to replace the cloud and keep data processing local.

For the purpose of optimizing the use of such ad-hoc cloud infrastructure powered by the

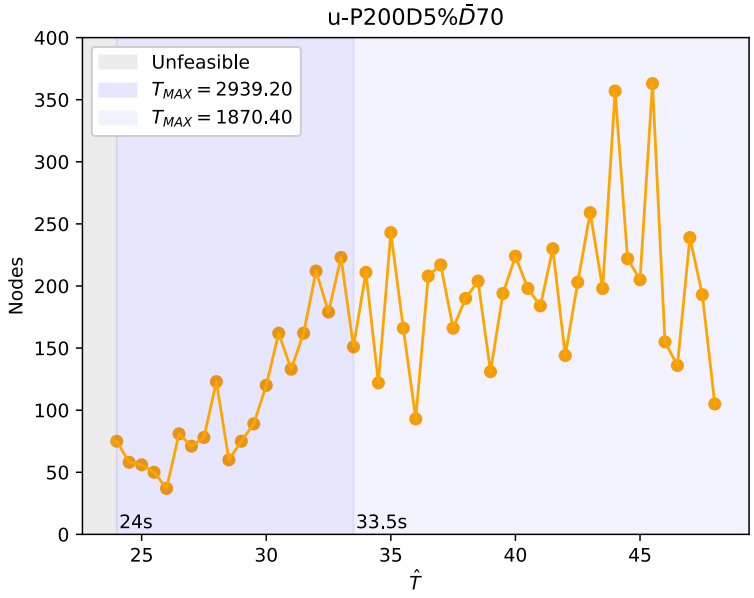


Figure 4.14 Number of nodes explored by CPLEX on varying $\hat{T}$ for the instance u-P200D5%$\bar{D}$70.

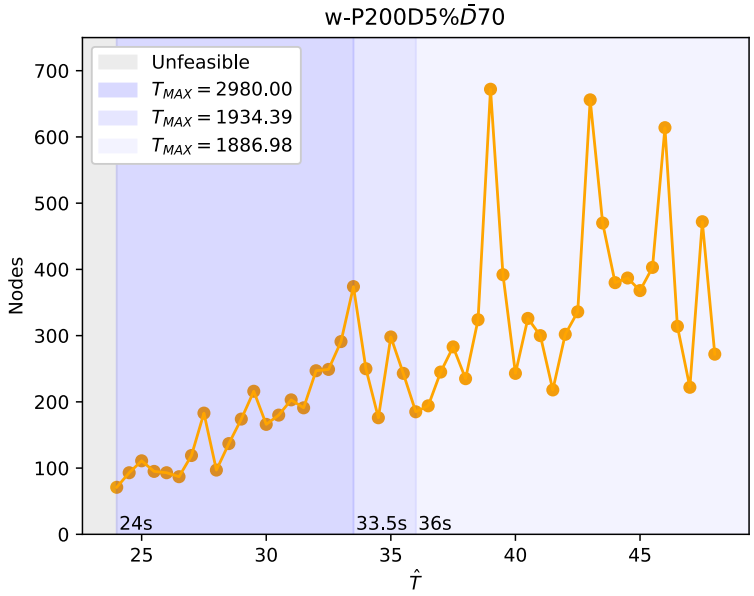


Figure 4.15 Number of nodes explored by CPLEX on varying $\hat{T}$ for the instance w-P200D5%$\bar{D}$70.

swarming UAVs, we introduced a new optimization problem, namely the Covering-Assignment Problem for swarm-powered ad-hoc clouds - CAPsac, based on a real-life use-case in the emer-

gency management field: swarm-powered distributed 3D reconstruction for humanitarian emergency response. After having established the relationship between the general problem and the specific use-case, we presented the NP-Hardness proof of the CAPsac and described two MILP formulations for it.

Given a set of geo-positioned aerial pictures (data) that are subject to geolocation/clustering constraints, CAPsac minimizes the 3D mapping (data-processing phase) completion time by jointly computing: (i) the optimal covering of photos (workload configuration), and (ii) the optimal assignment of photographed sub-regions (workload assignment) to UAVs (computing elements). Besides being a way to provide optimal solutions for the problem, our integrated decision model contrasts with the *decompose-then-allocate* and the *allocate-then-decompose* paradigms usually seen in (both the cloud computing optimization and) the multi-robot task allocation literature. Finally, modeling CAPsac in this way is flexible and amendable to take into account any other additional ground computing elements connected to the swarm itself.

In order to assess the proposed formulations, a series of computational experiments was conducted with a set of unweighted and weighted realistic benchmark instances available online (`https://github.com/ds4dm/CAPsac`). The experiments revealed that the photo-based formulation "PB" was more efficient by using ordering inequalities that remove from the feasible continuous search space those sub-regions whose boundaries are not regular (e.g. left boundary at the right of a right boundary). However, the different branching priority strategies and row generation methods have not proven to yield a performance gain while solving "PB". Column Generation was employed in the region-based formulation "RB", but the presence of highly degenerate optimums led to long execution times.

Finally, the sensitivity analysis of the formulation "PB" showed that it becomes more difficult to solve as the reliability factor σ increases. Tests with varying values for the maximum allowed transmission time $\hat{T}$ also presented a slight gain of performance as $\hat{T}$ approaches a limit when the problem becomes infeasible.

Appendix: Region-based CAPsac

The *CAPsac* problem can be addressed by explicitly considering the set $\mathcal{S}$ of all feasible rectangular subsets of photos, such that each element of $\mathcal{S}$ corresponds to a possible rectangular sub-region to be 3D-reconstructed. It is important to remark that the cardinality of $\mathcal{S}$ is polynomial and bounded by $O(|C|^2|L|^2)$, which is $O(|P|^4)$ in the worst scenario:

Proposition 3. *Let $\mathcal{S}$ be the set of all feasible rectangular sub-regions to a CAPsac instance. Then, $|\mathcal{S}|$ is bounded by $O(|C|^2|L|^2)$, which is $O(|P|^4)$ in the worst case.*

Proof. As in Section 4.3.2, any feasible hyperectangle $S \in \mathcal{S}$ is defined by a tuple (α^S, β^S, γ^S, ω^S) of latitudes and longitudes corresponding to the left, right, bottom, and top borders of S, respectively, with $\alpha^S, \beta^S \in C$ and $\gamma^S, \omega^S \in L$, and such that $\alpha^S \leq \beta^S$ and $\gamma^S \leq \omega^S$. Therefore, $\mathcal{S} = C \times C \times L \times L$. Since $1 \leq |C| \leq |P|$ and $1 \leq |L| \leq |P|$, $|\mathcal{S}|$ is bounded by $O(|P|^4)$. □

In particular, the photos are commonly spread across the target region in a grid pattern to fulfill photo footprint overlapping constraints [96]. Consequently, $|C|$ and $|L|$ are usually far smaller than $|P|$, and hence, $|C|^2 \cdot |L|^2$ is in practice usually significantly smaller than $|P|^4$.

Let $\mathcal{S}_p$ be the collection of rectangular subsets $S \in \mathcal{S}$ that cover photo $p \in P$. For each set $S \in \mathcal{S}$, denote t^S the photo processing time of S, and μ_S^{hd} the amount of data to transfer from drone $h \in D$ to the drone $d \in \bar{D}$ if S is selected. Let q_d^S be the binary variable equal to 1 if S is allocated to drone $d \in \bar{D}$. Finally, let us denote o^S the auxiliary binary variable that is equal to 1 if S is selected, and 0 otherwise.

The region-based formulation of the *CAPsac* is expressed as follows:

$$\min_{q,o} \quad T_{\max} \tag{4.31}$$

$$\text{s.t. } T_{\max} \geq \sum_{S \in \mathcal{S}} t^S q_d^S \qquad \forall d \in \bar{D} \tag{4.32}$$

$$\hat{T} \cdot \phi^{hd} \geq \sum_{S \in \mathcal{S}} \mu_S^{hd} q_d^S \qquad \forall (h,d) \in D \times \bar{D} \tag{4.33}$$

$$\sum_{d \in \bar{D}} q_d^S \geq \sigma o^S \qquad \forall S \in \mathcal{S} \tag{4.34}$$

$$\sum_{S \in \mathcal{S}_p} o^S \geq 1 \qquad \forall p \in P \tag{4.35}$$

$$\sum_{S \in \mathcal{S}} o^S = m \tag{4.36}$$

$$z^{hd} \leq \sum_{S \in \mathcal{S}} \mu_S^{hd} q_d^S \qquad \forall (h,d) \in D \times \bar{D} \tag{4.37}$$

$$\phi^{hd} \leq \bar{c}^{hd} z^{hd} \qquad \forall (h,d) \in D \times \bar{D} \tag{4.38}$$

MMF constraints [81]

$$o^S, q_d^S \in \{0,1\} \qquad \forall S \in \mathcal{S}, \forall d \in \bar{D} \tag{4.39}$$

$$w_{ij}^{hd} \in \{0,1\} \qquad \forall (i,j) \in A, \forall (h,d) \in D \times \bar{D} \tag{4.40}$$

$$\phi^{hd} \geq 0, z^{hd} \in \{0,1\} \qquad \forall (h,d) \in D \times \bar{D} \tag{4.41}$$

$$u_{ij} \geq 0 \qquad \forall (i,j) \in A. \tag{4.42}$$

The objective function (4.31) minimizes the makespan $T_{\max}$, which is computed by constraints

(4.32). Constraints (4.33) limit the networking delay for each photo transmission traffic demand. Constraints (4.34) impose that the selected subsets in $\mathcal{S}$ are assigned to σ drones that can do the 3D reconstruction. The set of constraints (4.35) ensures that each photo $p \in P$ is covered at least once, and constraint (4.36) defines the number of selected subsets to m, i.e., the number of drones that can perform the 3D reconstruction. The transmission rates are defined by (4.37) and the ones in [81], following MMF rate allocation, as explained in Section 4.3.2. Finally, domain constraints are given in (4.39)-(4.42).

The cardinality of $\mathcal{S}$ is bounded by $O(|P|^4)$ (Proposition 3). Therefore, the number of constraints in the formulation is bounded by $O(|P|^4)$ due to the amount of constraints (4.34). The number of variables is bounded by $O(|D| \cdot |P|^4)$ due to the number of variables q_d^S.

CHAPTER 5 ARTICLE 2: HEURISTICS FOR OPTIMIZING 3D MAPPING MISSIONS OVER SWARM-POWERED AD-HOC CLOUDS

Authors: Leandro R. Costa, Daniel Aloise, Luca G. Gianoli, and Andrea Lodi.
Submitted to the *Journal of Heuristics*[1].

Abstract: Drones have been getting more and more popular in many economy sectors. Both scientific and industrial communities aim at making the impact of drones even more disruptive by empowering collaborative autonomous behaviors — also known as swarming behaviors — within fleets of multiple drones. In swarming-powered 3D mapping missions, unmanned aerial vehicles typically collect the aerial pictures of the target area whereas the 3D reconstruction process is performed in a centralized manner. However, such approaches do not leverage computational and storage resources from the swarm members. We address the optimization of a swarm-powered distributed 3D mapping mission for a real-life humanitarian emergency response application through the exploitation of a swarm-powered ad-hoc cloud. Producing the relevant 3D maps in a timely manner, even when the cloud connectivity is not available, is crucial to increase the chances of success of the operation. In this work, we present a mathematical programming heuristic based on decomposition and a variable neighborhood search heuristic to minimize the completion time of the 3D reconstruction process necessary in such missions. Our computational results reveal that the proposed heuristics either quickly reach optimality or improve the best known solutions for almost all tested realistic instances comprising up to 1000 images and fifteen drones.

Keywords: *Cloud Computing, Swarm, 3D Reconstruction, Workload Optimization*

5.1 Introduction

Unmanned Aerial Vehicles (UAVs), which are also referred to as drones, are a remotely operated aircraft. Their aerial capabilities and low cost made them an attractive option for operations as building inspection, photo collection, and area surveillance. That explains their popularity and adoption in a multitude of sectors of the economy.

Besides being remotely operated by pilots, drones can also operate autonomously when obeying an on-board flight controller. This is handled by employing a collaborative intelligence

[1]Preprint available at [125]

program on each agent of the fleet while keeping them connected on the same wireless network. This collaborative capability that simulates natural swarms [58] (e.g., ant colonies, bee swarms, and bird flocks) is further leveraged by applying swarm robotics to conceive a fleet of fully autonomous drones focusing on fulfilling a common mission. Swarm robotics studies how to coordinate, in a distributed and decentralized manner, a large group of simple embodied robots to perform collective tasks and maximize the swarm performance [12]. Such swarm behavior is a powerful tool to foster UAV applications, given its capability to deliver both performance and resilience without requiring any centralized control. Swarming UAVs are deployed to support operations in a long list of domains [97], including forestry [98–100], archaeology and architecture [101–105], environmental monitoring [106–110], emergency management [111–113] and precision agriculture [114–116].

Likewise, the swarm robotics has drawn the attention of the operations research community as an attractive opportunity to improve the efficiency of swarm-powered missions [19,20]. For instance, decentralized optimization methods have been leveraging search problems [21–26], target assignment problems [27–37], node covering problems [38,39], scheduling problems [40], etc.

UAV swarming solutions are typically used in synergy with other technologies such as digital photogrammetry [45, 46], which focuses on extracting and displaying the relevant 2D/3D geometric information from the portrayed physical environment. Given a set of pictures that are fairly distributed across the area of interest — multiple shooting points and shooting perspectives should be considered for improved performances — it is possible to extract a 3D model of the region itself by performing a so-called *three-dimensional reconstruction*. In terms of synergy between swarming and photogrammetry technologies, the 3D reconstruction process is typically included in a swarm-powered mission pipeline where swarming UAVs are responsible for collecting, as fast as possible, the set of aerial images required to properly build the 3D map of the desired area.

Although the aerial photo collection is typically carried out in parallel by multiple agents, current 3D mapping literature addresses the 3D-reconstruction process in a centralized manner [48–52,121]. Meyer *et al.* [47] employed 3D reconstruction solutions which were conducted in a centralized base-station to survey a heritage site in Mexico, while other authors focused on multi-sensor data fusion to feed the 3D reconstruction algorithms with more accurate data [53, 117].

Such approaches are susceptible to internet connectivity and network latency issues. Thus, the distributed power within the UAV swarm can be exploited establishing an ad-hoc cloud infrastructure able to safely perform the 3D reconstruction of the considered 3D mapping

mission [15–18]. This strategy profits from the computational power in the microcomputers installed on the swarming UAVs instead of the computational resources present in a powerful computer (e.g. [145]).

The motivation of this paper arises from a real-life problem in the emergency response field. The decision-making and situation awareness during a first response operation are highly boosted by 3D maps of the affected region. Such 3D models allow the first responders to detect relevant threats like damage in roads and buildings or insecure zones. Thus, creating 3D maps must be quickly done regardless internet connectivity to enhance the safety and efficiency of the first responders.

In [124], the authors introduced the Covering-Assignment Problem for swarm-powered ad-hoc clouds (CAPsac), whose objective, in the context of a 3D mapping UAV mission, is to optimally generate and place the multi-node computing workload that will be responsible for performing the 3D reconstruction process. Since each computing node is responsible for reconstructing a specific sub-region on a specific drone, the optimal solution of the problem describes how to minimize the processing time by optimally i) splitting the set of available photos to form multiple sub-regions and then ii) assigning each sub-region to a specific UAV.

During emergency field operations, each minute counts and nearly optimal solutions of the CAPsac problem must be computed as quickly as possible. In this way, 3D maps can be promptly put in the hands of the emergency responders, while leaving the UAVs available to perform additional critical tasks — including other 3D mapping missions. Furthermore, a fast CAPsac solution allows the flying UAVs to preserve the battery life by limiting the idle flying periods spent waiting for 3D processing instructions. Note that the autonomy of typical commercial drones does not go beyond one hour.

In this paper, we propose a Variable Neighborhood Search (VNS) heuristic [146] based on *sub-tree reconstruction*, *splitting hyperplane reallocation*, *sub-region transfer*, and *sub-region swap* neighborhoods for quickly optimizing a swarm-powered 3D mapping mission according to CAPsac. We also develop a mathematical programming-based heuristic, namely *Decomposition-based heuristic*, to assess the performance of matheuristics.

The paper is organized as follows. The next section describes how to cast a swarm-powered 3D mapping mission as CAPsac problem. The proposed mathematical programming-based heuristic is presented in Section 5.3. Likewise, the VNS fundamentals and the proposed VNS-based heuristic are exposed in Section 5.4. Finally, Section 5.5 shows and analyzes the computational results obtained over realistic instances, while Section 5.6 outlines our concluding remarks.

5.2 Swarm-powered 3D mapping mission as a CAPsac

A swarm-powered 3D mapping mission is decomposed in two main phases [124]:

1. **Photo collection**: the UAVs of the swarm dynamically collaborate to collect all the necessary aerial pictures of the area of interest. Note that the set of required pictures is typically computed by dedicated mapping software and is merely an input of the mapping mission [96].

2. **3D processing** [87]: the collected pictures are collaboratively processed by the microcomputers installed on the UAVs to produce a 3D map. During this process, the computing workload can be parallelized over the available computing units. Furthermore, pictures can be transferred over the inter-drone wireless network to satisfy the input requirements of the distributed reconstruction tasks.

By casting a swarm-powered 3D mapping mission as a CAPsac, we aim to optimize the 3D processing phase only, with no direct control over the photo collection step. Therefore, we consider that the set of photos taken by the drones as well as their locations are input parameters for CAPsac solution. A swarm-powered 3D processing application employing a swarm of four UAVs is illustrated in Figure 5.1 [124].

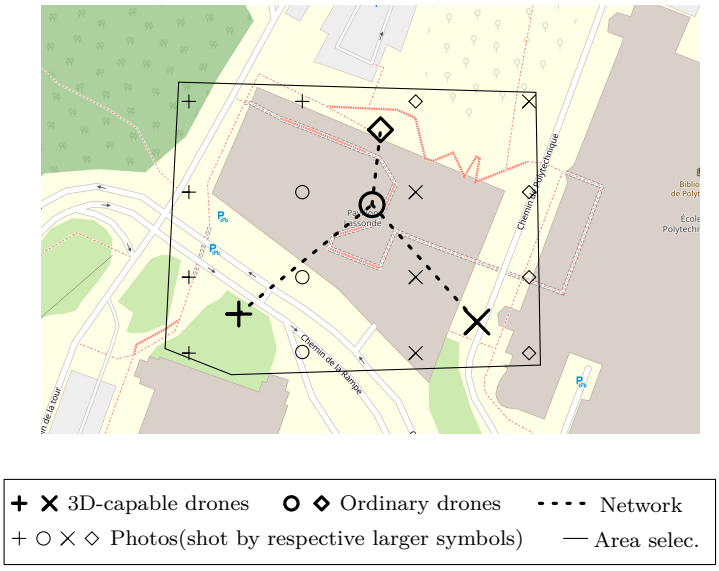


Figure 5.1 Swarm-powered 3D mapping mission as an instance of CAPsac. ©2020 IEEE [124].

In that example, the large "×", "+", "◇", and "○" identify the drones. Only drones with powerful microcomputers able to sustain the 3D reconstruction methods are considered *3D-capable* [124]. The 3D-capable drones are represented by the + and the × symbols. That is, + and × are responsible for performing the 3D processing stage. The region portrayed by the set of photos P is bounded by the continuous lines. Further, the photos are represented by the small "×", "+", "◇", and "○". Their symbols match with the UAV where the photo is stored, for instance, pictures identified by a small + are stored in the UAV represented by the large +.

A solution to the CAPsac problem describes how to parallelize optimally a massive 3D reconstruction task into smaller 3D reconstruction sub-tasks to be distributed across the 3D-capable drones. This means that a 3D reconstruction sub-task is associated with a specific sub-region of the target region. Therefore, the optimal solution of CAPsac minimizes the completion time (i.e., makespan) of the whole 3D reconstruction phase. When optimizing the CAPsac, three constraints cannot be neglected [124]: the 3D reconstruction sub-regions must be a *spatial-convex covering*; the communication delays follow the *Max-Min Fairness* (MMF) paradigm [79]; the 3D reconstruction sub-tasks are distributed across the 3D-capable drones according to a *reliability factor*.

More precisely, the sub-regions form a *spatial-convex covering* if and only if the union of all sub-regions (subsets of photos) is equal to the target region and all sub-regions are a *spatial-convex set* — all photos lying inside the set's convex hull are allocated to that same sub-region [124]. Figure 5.2 shows an example of a spatial-convex set, whereas Figure 5.3 illustrates a set of photos which is *not* a spatial-convex set. In those figures, photos allocated to the set are represented by the filled "•" and the empty "○" represent photos not allocated to the set.

The communication delays must be taken into account since the photo collection stage is not optimized in the CAPsac, and the 3D reconstruction of a sub-region cannot start until its assigned drone has in its hard disk all photos belonging to that sub-region, that is, all 3D reconstruction input photos. Thus, a drone must request to the other drones the photos to complete its designed sub-region. A single tree network topology (dotted lines in Figure 5.1), on top of which establishing TCP sessions — one per photo transfer, is adopted as inter-drone communication means. The MMF paradigm is considered to approximate the TCP-based rate allocation for multiple photo transfer sessions occurring within the swarm communication tree [82].

Finally, the sub-region (3D reconstruction sub-task) to drone assignments must be robust to drone malfunction. This is addressed by introducing a *reliability factor*, which dictates the

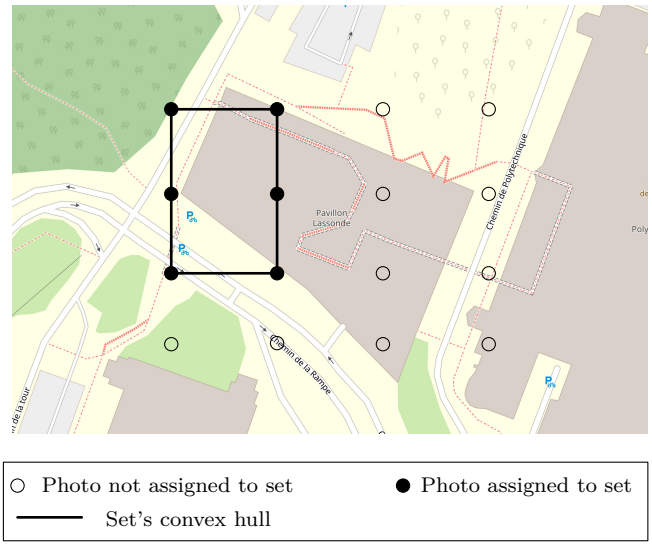


Figure 5.2 Spatial-convex set and the respective convex hull. Adapted ©2020 IEEE [124].

minimal number of drones each sub-region must be assigned. For instance, a solution for a reliability factor equal to one is illustrated in Figure 5.4. There, the two spatial-convex sets (the same number as the 3D-capable drones) are represented by the dashed (left) and the dashed-and-dotted lines (right). The sub-region on the left is assigned to the UAV "+" and the sub-region on the right is assigned to the UAV "×".

Let us now define the notation for CAPsac. We consider the region photographed by the set of photos P as the region targeted by the 3D processing stage. The CAPsac considers that photo positioning was performed beforehand, and then, the locations of all captured photos in P are fixed and known. Further, let λ_p and μ_p be the estimated photo processing time and the data size of photo $p \in P$, respectively. Besides, a set of drones D is at disposal but only a subset $\bar{D} \subseteq D$, where $|\bar{D}| = m$, is able to perform the 3D reconstruction. Therefore, the number of sub-regions constructed by the CAPsac is equal to m. Given a drone $d \in D$, the binary parameter θ_{dp} indicates if d has the photo $p \in P$ in its hard disk.

The drones are disposed in an undirected tree topology $T = (N, A)$, in which N corresponds to the set of nodes where each drone is located and A comprises the sets of arcs linking the nodes in N. The set F is composed by all transmission demands across pairs of drones, such that $F = \{f^{hd} | (h, d) \in D \times \bar{D}\}$, in which a specific transmission demand from the drone h to the drone d is represented by f^{hd}. For each demand f^{hd}, let V^{hd} be the set of arcs $(i, j) \in A$ in the unique routing path between h and d, and $\overline{F}_{ij}$ the set of demands using the arc (i, j), i.e., $\overline{F}_{ij} = \{f^{hd} \in F | (i, j) \in V^{hd}\}$. Given an arc $(i, j) \in A$, let c_{ij} be the capacity of the arc (i, j). Also, denote by $\bar{c}^{hd}$ the minimum c_{ij} in V^{hd}. Finally, each transmission demand is

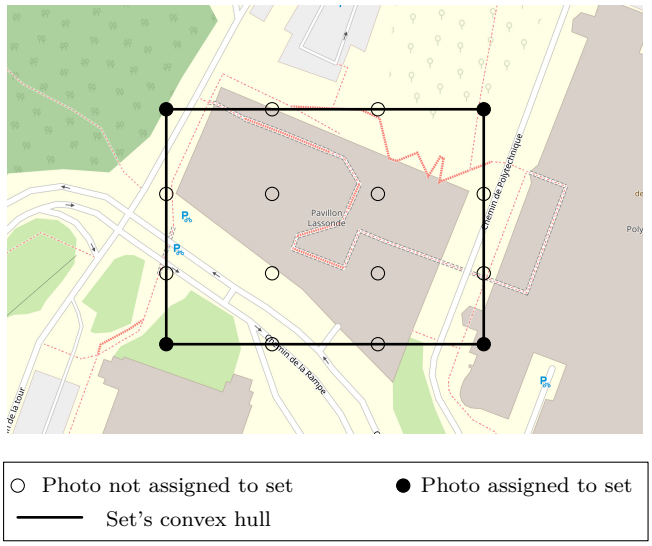


Figure 5.3 Ordinary set and the respective convex hull. ©2020 IEEE [124].

allowed within a time limit $\hat{T}$, after which it is considered as infeasible.

All the notation of the presented parameters is presented in Table 5.1.

Table 5.1 CAPsac parameters for a 3D mapping mission. Adapted ©2020 IEEE [124].

Parameters	Description
λ_p	estimated processing time of photo p
μ_p	amount of data of photo p
θ_{dp}	equal to 1 if drone d has the photo p stored in its memory
F	set of traffic demands between each pair of drones
V^{ab}	routing path of a demand $f^{ab} \in F$ from the drone a to the drone b
c_{ij}	transmission capacity of the link $(i,j) \in A$
$\bar{c}^{ab}$	minimum c_{ij} for $(i,j) \in V^{ab}$
$\overline{F}_{ij}$	set of demands that use link $(i,j) \in A$
σ	reliability factor
m	number of 3D-capable drones (equiv. number of sub-regions)
$\hat{T}$	maximum allowed time for transmitting photos between drones

5.2.1 Mathematical Formulation

Mathematical formulations to solve the *CAPsac* were proposed in [124]. In this work, we adopt the region-based formulation (*rCAPsac*) [124]. The *rCAPsac* formulation minimizes the completion time of the 3D processing stage $T_{\max}$, and it exploits the set $\mathcal{S}$ containing all

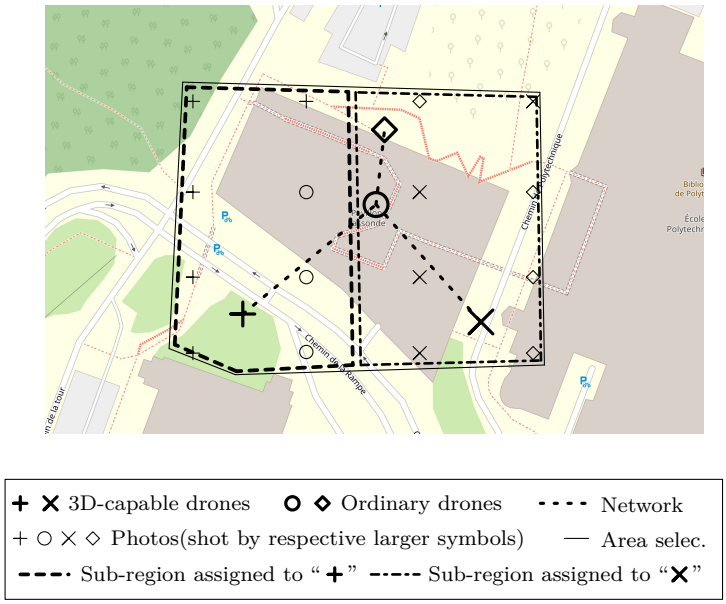


Figure 5.4 Spatial-convex covering and its assignment optimizing the makespan of a 3D mapping mission.©2020 IEEE [124].

rectangular spatial-convex sets of photos in P. Given a photo $p \in P$, the set $\mathcal{S}_p$ comprises all sub-regions $S \in \mathcal{S}$ which contain photo p. The binary variables q_d^S indicate if $S \in \mathcal{S}$ is selected and its 3D reconstruction is allocated to the drone $d \in \bar{D}$. Remark that all selected sub-regions must be assigned to at least σ (reliability factor) drones. Also, for each $S \in \mathcal{S}$, the binary variable o^S indicates (i.e., $o^S = 1$) if the sub-region S is used in the solution; the t^S represents the time required to perform the 3D reconstruction of S; the μ_S^{hd} expresses the amount of data added into the transmission demand f^{hd} when S is selected.

Since 3D reconstruction cannot start until a drone has all its input pictures, the communication time required to exchange photos among the drones cannot be neglected. For each $f^{hd} \in F$, we denote by z^{hd} the binary variable which indicates when the demand f^{hd} is active — if there exists any data to be exchanged through f^{hd}. Accordingly, continuous variables ϕ^{hd} correspond to the transmission rate performed by the demand f^{hd}.

As mentioned, in the case of the CAPsac, the transmission rate allocation follows the MMF paradigm. Such rate allocation attends the MMF if and only if there is at least one bottleneck link $(i, j) \in A$ on the routing path V^{hd} of each active demand $f^{hd} \in F$ [80]. Furthermore, a link (i, j) is a bottleneck of the demand f^{hd} if and only if [80]

(i) its capacity is saturated, i.e., $\sum_{f^{ab} \in \overline{F}_{ij}} \phi^{ab} = c_{ij}$ and

(ii) the transmission rate ϕ^{hd} of traffic demand f^{hd} is the highest among the traffic demands routed over link (i,j), i.e., $\phi^{hd} \geq \phi^{ab} \quad \forall f^{ab} \in \overline{F}_{ij}$.

Given a demand f^{hd}, let w_{ij}^{hd} be the binary variable equal to 1 if the link (i,j) is a bottleneck of f^{hd}. We denote by u_{ij} the highest transmission rate among the traffic demands carried by the link $(i,j) \in A$, that is, $u_{ij} = \max_{f^{ab} \in \overline{F}_{ij}} \{\phi^{ab}\}$.

The *rCAPsac* formulation of the *CAPsac* is described by the following MILP [124].

$$\min_{q,o} \quad T_{\max} \tag{5.1}$$

$$\text{s.t. } T_{\max} \geq \sum_{S \in \mathcal{S}} t^S q_d^S \qquad \forall d \in \bar{D} \tag{5.2}$$

$$\hat{T} \cdot \phi^{hd} \geq \sum_{S \in \mathcal{S}} \mu_S^{hd} q_d^S \qquad \forall f^{hd} \in F \tag{5.3}$$

$$\sum_{d \in \bar{D}} q_d^S \geq \sigma o^S \qquad \forall S \in \mathcal{S} \tag{5.4}$$

$$\sum_{S \in \mathcal{S}_p} o^S \geq 1 \qquad \forall p \in P \tag{5.5}$$

$$\sum_{S \in \mathcal{S}} o^S = m \tag{5.6}$$

$$z^{hd} \leq \sum_{S \in \mathcal{S}} \mu_S^{hd} q_d^S \qquad \forall f^{hd} \in F \tag{5.7}$$

$$\phi^{hd} \leq \bar{c}^{hd} z^{hd} \qquad \forall f^{hd} \in F \tag{5.8}$$

$$\sum_{(i,j) \in V^{hd}} w_{ij}^{hd} \geq z^{hd} \qquad \forall f^{hd} \in F \tag{5.9}$$

$$\sum_{f^{ab} \in \overline{F}_{ij}} \phi^{ab} \leq c_{ij} \qquad \forall (i,j) \in A \tag{5.10}$$

$$\sum_{f^{ab} \in \overline{F}_{ij}} \phi^{ab} \geq c_{ij} w_{ij}^{hd} \qquad \forall (i,j) \in A, \forall f^{hd} \in \overline{F}_{ij} \tag{5.11}$$

$$u_{ij} \geq \phi^{hd} \qquad \forall (i,j) \in A, \forall f^{hd} \in \overline{F}_{ij} \tag{5.12}$$

$$\phi^{hd} \geq u_{ij} - c_{ij}(1 - w_{ij}^{hd}) \qquad \forall (i,j) \in A, \forall f^{hd} \in \overline{F}_{ij} \tag{5.13}$$

$$o^S, q_d^S \in \{0,1\} \qquad \forall S \in \mathcal{S}, \forall d \in \bar{D} \tag{5.14}$$

$$w_{ij}^{hd} \in \{0,1\} \qquad \forall (i,j) \in A, \forall f^{hd} \in F \tag{5.15}$$

$$\phi^{hd} \geq 0, z^{hd} \in \{0,1\} \qquad \forall f^{hd} \in F \tag{5.16}$$

$$u_{ij} \geq 0 \qquad \forall (i,j) \in A. \tag{5.17}$$

The objective function (5.1) minimizes the makespan of the 3D processing phase, i.e., the maximum processing time $T_{\max}$ across all 3D-capable drones in the swarm. Accordingly,

$T_{\max}$ is computed by the constraints (5.2), in which, the summation $\sum_{S\in\mathcal{S}} t^S q_d^S$ determines the processing time of a given drone $d \in \bar{D}$. Constraints (5.3) restrict the latency of all transmission demands F up to $\hat{T}$. Constraints (5.4) enforce the reliability factor over all selected sub-regions. That is, when a sub-region S is selected, it is assigned to at least σ 3D-capable drones. The photos $p \in P$ are always covered given the constraints (5.5). The constraint (5.6) establishes m (number of 3D-capable drones) as the number of sub-regions being selected. The MMF transmission rate allocation is computed by the constraints (5.7)-(5.13). Constraints (5.7) allow a transmission to be active only if there exist data to be sent from drone h to drone d. Constraints (5.8) set the variables ϕ^{hd} to 0 when f^{hd} is inactive. All active demands have at least one bottleneck link in their routing path according to inequalities (5.9). Constraints (5.10) ensure that all link capacities are respected. Constraints (5.10) and (5.11) jointly force that all bottleneck links will be saturated. That covers the first condition to a link $(i, j) \in A$ to be a bottleneck link. The second bottleneck link condition is ensured by constraints (5.12) and (5.13). Constraints (5.12) make u_{ij} greater or equal to the highest transmission rate passing through the link (i, j). If $(i, j) \in A$ is a bottleneck link of the demand f^{hd}, constraints (5.13) ensure that ϕ^{hd} will not be exceeded by any other transmission rate passing thought link (i, j). The domain constraints are (5.14)-(5.17). The number of variables of the *rCAPsac* formulation can rapidly increase even though it is polynomial bounded by $O(|P|^4)$.

The CAPsac was proved NP-hard in [124]. A straightforward lower bound for the CAPsac is the perfect workload division among the available processing resources. In the context of a 3D mapping mission by UAVs, this means to equally spread the photo processing time of the whole target region by the number of 3D-capable drones. Given a set of photos P, a set of 3D-capable drones $\bar{D}$, and a reliability factor σ, the lower bound $Lb(P, \bar{D}, \sigma)$ is given by

$$Lb(P, \bar{D}, \sigma) = \frac{\sigma \sum_{p\in P} \lambda_p}{|\bar{D}|}. \tag{5.18}$$

5.3 Decomposition-based heuristic

The formulation given in the previous section can be used to devise heuristics for the CAPsac. However, before that, let us define $\bar{n}$ as the perfect workload division w.r.t. the number of photos per sub-region

$$\bar{n} = \frac{|P|}{|\bar{D}|}.$$

Inspired by the observation that good solutions have their workload w.r.t. the number of photos per sub-region close to $\bar{n}$, we decompose the problem by exploring first those solutions

whose sub-regions have cardinality close to $\bar{n}$.

To that purpose, we restrict the solution space of *rCAPsac* to sub-regions whose photo cardinalities lie inside an interval $[n_\ell, n_u]$, which is iteratively increased. That interval is initialized with $n_\ell = \lfloor \bar{n} \rfloor$ and $n_u = \lceil \bar{n} \rceil$. As solving this restricted *rCAPsac* formulation leads to possibly infeasible or suboptimal solutions, one can iteratively increase the interval $[n_\ell, n_u]$, and solve the new restricted CAPsac until the objective function value ($T_{\max}$) does not change between two consecutive iterations. This stopping condition is indeed sub-optimal. For example, in the presence of very contrasting photo-processing times λ_p among the photos $p \in P$, the optimal solution might contain sub-regions with small processing times and large cardinalities.

Algorithm 2 summarizes the decomposition-based heuristic. It considers Ω the set of distinct photo cardinalities from sub-regions in $\mathcal{S}$, and a mapping function ω (implemented as a hash table) which maps the cardinalities in Ω to their respective sub-regions in S. Step 1

Algorithm 2 Decomposition-based heuristic

1: Enumerate rectangular sub-regions $\mathcal{S}$;
2: Construct and sort Ω;
3: Build ω according to $\mathcal{S}$ and Ω;
4: $n_\ell \leftarrow \lfloor \bar{n} \rfloor$; $n_u \leftarrow \lceil \bar{n} \rceil$; $\mathcal{S}^0 \leftarrow \{\varnothing\}$; $T^0_{\max} \leftarrow \infty$; $i \leftarrow 0$;
5: **repeat**
6: $\quad i \leftarrow i + 1$;
7: $\quad \mathcal{S}^i \leftarrow \mathcal{S}^{i-1} + \omega(n_\ell) + \omega(n_u)$
8: $\quad T^i_{\max} \leftarrow$ solve *rCAPsac* encompassing sub-regions $\mathcal{S}^i$;
9: $\quad$ decrease n_ℓ to its closest smaller $n \in \Omega$;
10: $\quad$ increase n_u to its closest larger $n \in \Omega$;
11: **until** $T^{i-1}_{\max} = T^i_{\max}$ *or* $\mathcal{S}^i = \mathcal{S}$

enumerates the sub-regions $\mathcal{S}$ on $O(|P|^4)$ operations [124]. Step 2 constructs the sorted array Ω whereas step 3 creates ω. The values of n_ℓ, n_u, $\mathcal{S}^0$, $T^0_{\max}$, and i are properly initialized at step 4. For each iteration (steps 5-11): i is incremented in step 6. Then, the set $\mathcal{S}^i$ is updated to include the sub-regions of cardinality equal to n_ℓ and n_u in step 7. Then, step 8 solves the restricted *rCAPsac* formulation encompassing all sub-regions in $\mathcal{S}^i$. The new interval $[n_\ell, n_u]$ is obtained in steps 9-10 by updating n_ℓ and n_u. The algorithm iterates (lines 5-11) until the objective function value ($T_{\max}$) does not change between two consecutive iterations, or that $\mathcal{S}^i$ contains all the sub-regions $\mathcal{S}$. We remark that the algorithm also iterates (lines 5-11) while no feasible solutions have been found yet. Accordingly, if the algorithm does not find a feasible solution and $\mathcal{S}^i = \mathcal{S}$, the problem is proven infeasible. As a result, the heuristic takes at most $O(|\Omega|)$ iterations to finish in the worst case. In such worst scenario, the full problem

(i.e., the complete *rCAPsac* formulation) is solved once the interval $[n_\ell, n_u]$ encompasses all sub-regions in $\mathcal{S}$. It is worth pointing out that the optimal solution in one iteration remains feasible in the next one since increasing the interval $[n_\ell, n_u]$ is equivalent to simply adding new columns to the *rCAPsac* formulation. Consequently, it can be used as a feasible upper bound solution for the next iteration.

5.4 Variable Neighborhood Search for CAPsac

Our VNS heuristic exploits a *spatial partition tree* as a spatial data representation to ensure the spatial-convexity of sub-regions in a feasible solution. Furthermore, our proposed VNS is based on four neighborhoods, namely: *sub-tree reconstruction*, *splitting hyperplane reallocation*, *sub-region transfer*, and *sub-region swap*. They are explained together with the VNS fundamentals and our VNS implementation in the following sections. Finally, we present how the VNS heuristic ensures communication delays feasibility during its search.

5.4.1 Variable Neighborhood Search fundamentals

Variable Neighborhood Search is a stochastic search metaheuristic, i.e., a local optimal evading framework based on heuristics, which aims at reaching optimal or near-optimal solutions for global combinatorial optimization problems. VNS has been successfully applied to a vast range of NP-hard problems [147].

A generic combinatorial optimization problem can be formally defined as follows. Let $N = \{1, \ldots, n\}$ be a finite set and let $c = \{c_1, \ldots, c_n\}$ be an n-dimensional vector. Denote by $c(F) = \sum_{i \in F} c_i$ the cost associated to the finite set $F \subseteq N$. Given a collection of subsets $\mathcal{F}$ of N, a *binary combinatorial optimization* problem $\mathcal{C} = (N, \mathcal{F}, c)$ can be expressed as

$$\min\{c(f) : f \in \mathcal{F}\} \tag{5.19}$$

Let the n-dimensional binary vector $X^f = \{x_1^f, \ldots, x_n^f\}$ characterize a feasible solution such that $x_i^f = 1$ if $i \in N$ belongs to $f \in \mathcal{F}$, and $x_i^f = 0$ otherwise. Thus, $\mathcal{C}$ can be seen as minimizing over a polytope, i.e.,

$$\min\left\{c^T x | x \in conv\left\{X^f \in \{0,1\}^N | f \in \mathcal{F}\right\}\right\}. \tag{5.20}$$

A local minimum x^* of (5.20) is defined as

$$c^T x^* \leq c^T x, \forall x \in \mathcal{N}(x^*), \tag{5.21}$$

where $\mathcal{N}(x)$ is the *neighborhood* of x. A neighborhood $\mathcal{N}(x)$ is defined as the set of neighboring solutions x' obtained from x by systematic changes in the components of x (e.g., complementing elements of x). Acknowledging that a local minimum w.r.t. a neighborhood is not necessarily the local optimum for another neighborhood, VNS employs several different neighborhoods to escape local minima and reach global optimality. Note that a global minimum is a local minimum regardless of the considered neighborhood.

Likewise simulated annealing and tabu search [148], VNS keeps a single solution x during the whole execution of the algorithm. However, if differs from other metaheuristics mainly by its search mechanism that searches for better solutions in increasingly wider neighborhoods of x, according to the parameters $k \in \mathbb{Z}^+$ and $k_{\max} \in \mathbb{Z}^+$. Let us define the neighborhoods adopted by the VNS as the set $\boldsymbol{\mathcal{N}} = \{\mathcal{N}_1, \ldots, \mathcal{N}_{k_{\max}}\}$ such that neighborhoods $\mathcal{N}_k \in \boldsymbol{\mathcal{N}}$ are sorted according to their size (i.e., cardinality of $\mathcal{N}_k(x)$). First, starting with $k = 1$, a random neighboring solution x' is obtained from neighborhood $\mathcal{N}_k(x)$. Then, a local descent method is performed from x' leading to another local minimum x''. If x'' is worse than (or equal to) x, it is dismissed and the local descent method starts from a new random neighbor x' concerning the next neighborhood of x, i.e., $\mathcal{N}_{k+1}(x)$. Otherwise, x'' replaces x and the algorithm resets k to 1, i.e., the search is resumed in the neighborhood $\mathcal{N}_1$ of the new best solution. Every time the maximum neighborhood $\mathcal{N}_{k_{max}}$ is reached (i.e.,$k = k_{\max}$), VNS restarts from the first neighborhood ($k = 1$). The VNS iterates until the stopping condition (e.g., maximum CPU time) is met. Given the order of neighborhoods in $\boldsymbol{\mathcal{N}}$, the method favors the exploration of solutions in small neighborhoods of x, increasing the size of the neighborhood if necessary. The basic steps of VNS are given in Algorithm 3.

5.4.2 Spatial partition tree

The *spatial partition tree* allows the VNS to efficiently explore the solution space without directly handling spatial-convexity constraints. Inspired by the *k-dimensional trees* [149], the spatial partition tree is a (**not** necessarily complete) binary tree which recursively splits the metric space under consideration. In this kind of tree, all nodes are directly associated with a portion (sub-region) of the target region.

In the CAPsac context, the target region is discretized by the positions of the photos P. Let us define L as the set of distinct photo latitudes and let C be the set of distinct photo longitudes, such that $1 \leq |C| \leq |P|$ and $1 \leq |L| \leq |P|$. The sets C and L offer a simple way to partition the space into rectangular sub-regions, such that any rectangular sub-region r can be represented by its left $c_< \in C$, right $c_> \in C$, inferior $\ell_\vee \in L$, and superior $\ell_\wedge \in L$ borders. As a result, constructing a spatial partition tree splits the target region and forms

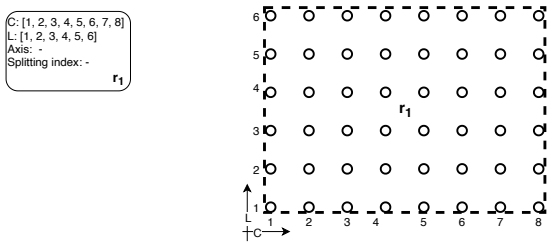
Algorithm 3 Variable Neighborhood Search - VNS

Select a set of neighborhoods $\mathcal{N}_k$, for $k = 1, \ldots, k_{max}$;
Find an initial solution x;
repeat
  $k \leftarrow 1$
  repeat
    (*Diversification step*): Choose a random neighboring solution x' from $\mathcal{N}_k$;
    (*Intensification step*): Apply a local descent method from x', obtaining x'';
    if $cost(x'') < cost(x)$ **then**
      $x \leftarrow x''$;
      $k \leftarrow 1$;
    else
      $k \leftarrow k + 1$;
    end if
  until $k > k_{max}$
until a stopping criterion is met

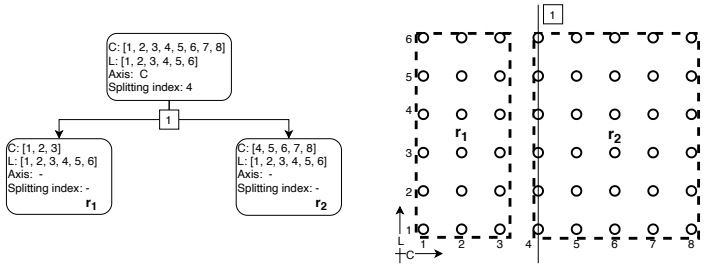
a partition of rectangular sub-regions.

A spatial partition tree is successively constructed by splitting the region of interest into new sub-regions as illustrated in Figure 5.5. At first, the spatial partition tree contains only the root node corresponding to the whole target region as shown in Figure 5.5a. The photos P (target region), identified by the "$\circ$", are enclosed by a unique rectangular region r_1 (dashed lines). All nodes in the tree (round-cornered rectangles) have (i) their associated enclosed longitudes on axis C ("C"); (ii) their associated enclosed latitudes on axis L ("L"); (iii) the axis selected to guide the splitting ("Axis"); (iv) the index on the selected axis chosen to define the splitting hyperplane ("Splitting index"). It is worth to mention that the leaf nodes have the attributes (iii) and (iv) empty, and the photos inside a sub-region are implicitly given by the lists C and L.

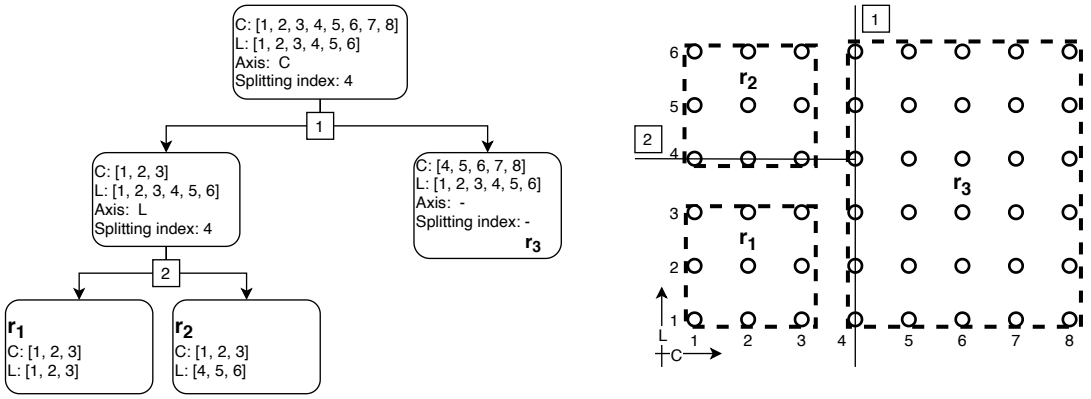
In each successive iteration (splitting step), a leaf node (rectangular sub-region) is selected to be split by means of an axis-aligned splitting hyperplane. The selected splitting hyperplane (denoted by "$\square$" and represented by lines) is chosen from the location of the photos associated with a sub-region being split. The splitting step is repeated $|\bar{D}| - 1$ times, that is, until $|\bar{D}|$ sub-regions are formed. For instance, starting from the root node in Figure 5.5a, the construction of a partition with four sub-regions (i.e., $|\bar{D}| = 4$) is done after three iterations presented in Figures 5.5b, 5.5c, and 5.5d, respectively. Finally, Figure 5.5d illustrates the final spatial-partition tree and its respective partition and splitting hyperplanes, where the photos P are split into the four rectangular sub-regions r_1, r_2, r_3, and r_4. This data structure guarantees the creation of sub-regions that are always spatial-convex sets.



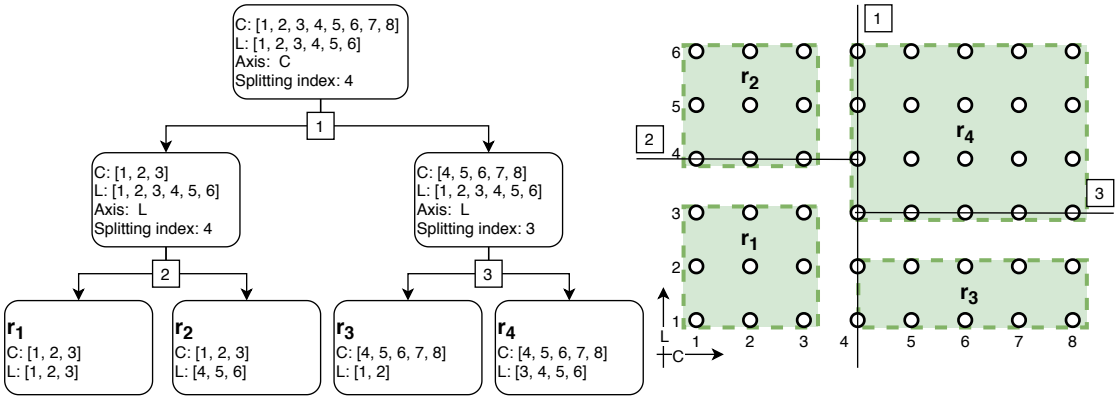
(a) Initial step. Spatial-convex tree (left) and the target region (right).



(b) Iteration 1. Spatial-convex tree (left) and its respective partition (right).



(c) Iteration 2. Spatial-convex tree (left) and its respective partition (right).



(d) Final iteration resulting in the spatial-partition tree (left) and its respective partition and splitting hyperplanes (right).

Figure 5.5 Illustrative construction of a spatial-convex tree with four sub-regions.

5.4.3 Sub-tree reconstruction neighborhood

Given a spatial partition tree T, denote by $D(T)$ the maximum depth of the nodes of T. Let $N(d)$ be the set of all nodes at the depth d of T where $0 \leq d \leq D(T)$. The *sub-tree reconstruction* neighborhood randomly reconstructs (as explained in Section 5.4.2) a sub-tree rooted at a non-leaf node $n \in N(d)$. Figure 5.6 illustrates a neighboring solution in the right obtained from reconstructing the sub-tree rooted at the grey node (in depth 0) of the spatial partition tree in the left.

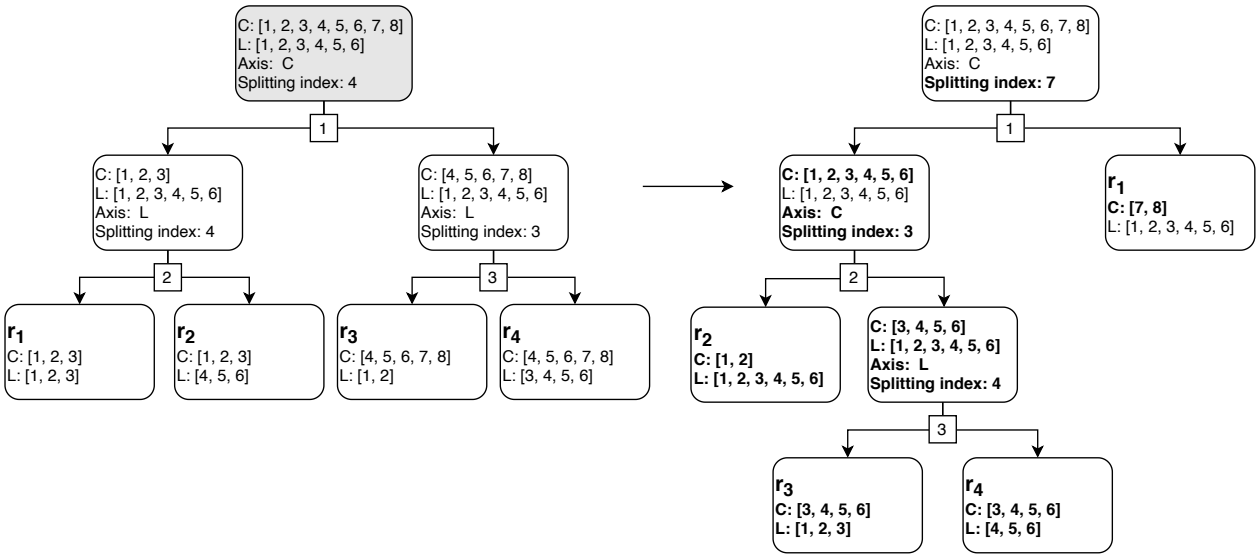


Figure 5.6 Neighboring solution (right) obtained from a spatial partition tree (left) in the sub-tree reconstruction neighborhood.

5.4.4 Splitting hyperplane reallocation neighborhood

Given a spatial partition tree T and a node n that is a parent of a leaf node, this neighborhood includes all feasible solutions obtained by reallocating the splitting hyperplanes of the non-leaf nodes of the sub-tree rooted in n (named T_n). Furthermore, such reallocation is made changing the splitting index (component "Splitting index") and/or the axis orienting the hyperplane (component "Axis"). Only feasible changes of the splitting index are allowed in order to avoid nodes associated with empty portions of target region. Unlike the sub-tree reconstruction neighborhood, the *splitting hyperplane reallocation* neighborhood does not restrict n to be a node in a specific depth $0 \leq d \leq D(T)$. Moreover, it keeps the depth of the resulting sub-tree rooted in n equal to that of the original sub-tree. Figure 5.7 illustrates that procedure in which the splitting index of the (colored) node n (on the left) is changed to originate the neighboring tree on the right.

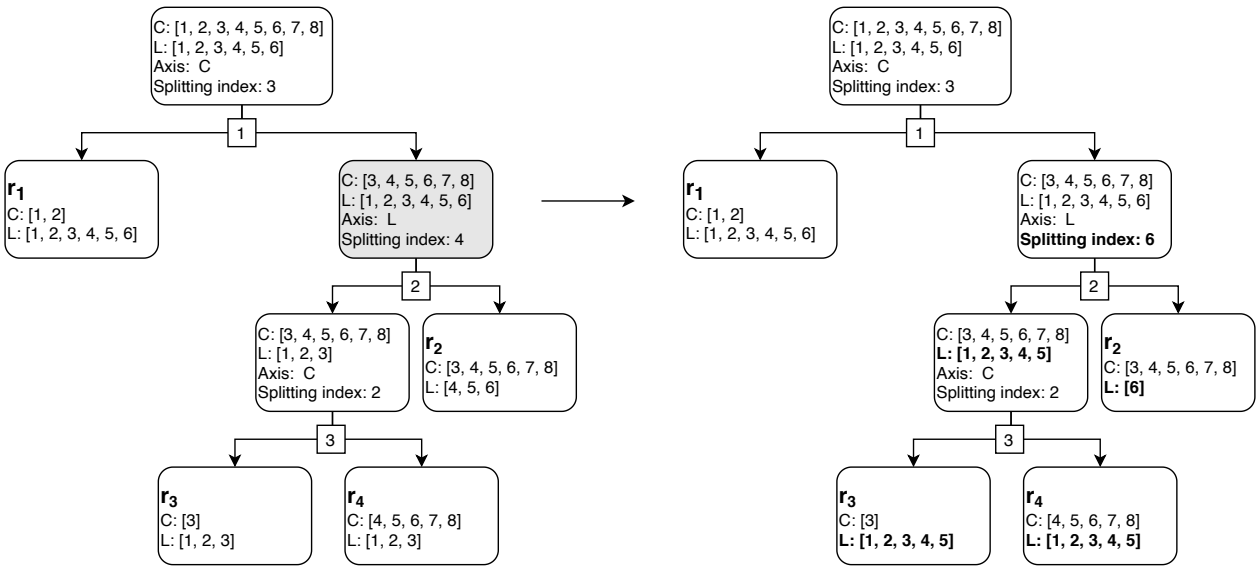


Figure 5.7 Neighboring tree (right) obtained from a spatial partition tree (left) in the splitting hyperplane neighborhood.

5.4.5 Sub-region transfer neighborhood

This neighborhood encompasses all solutions obtained by transferring one sub-region assignment (r_ℓ, d_i) from a drone d_i to a drone d_j such that $i \neq j$. In Figure 5.8, the transfer of the sub-region r_3 from drone d_1 to drone d_2 generates a neighboring solution according to the sub-region transfer neighborhood for a reliability factor σ equal to two.

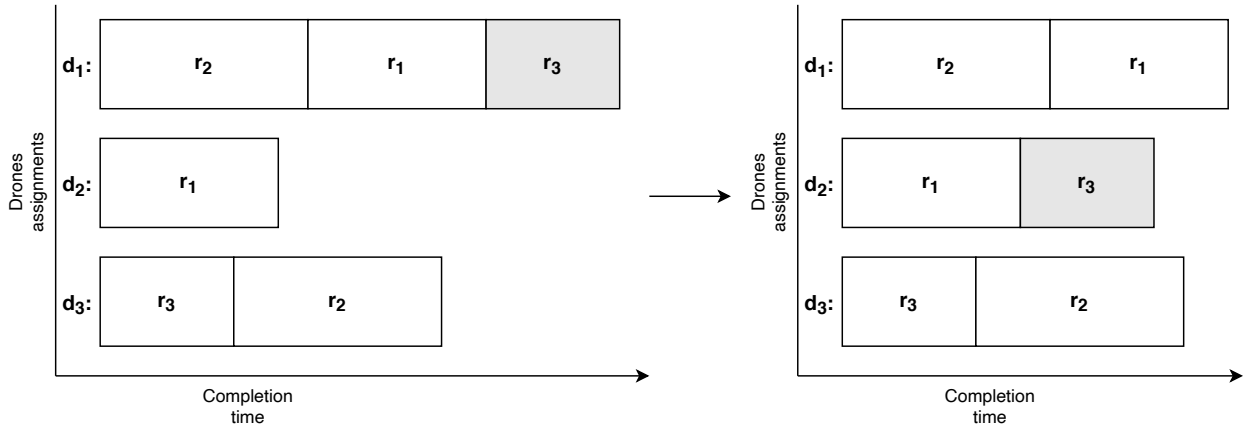


Figure 5.8 Neighboring solution (right) obtained from a spatial partition tree (left) in the sub-region transfer neighborhood for $\sigma = 2$.

5.4.6 Sub-region swap neighborhood

Given a solution, its neighbors in the sub-region swap neighborhood are obtained by the swap of sub-regions assignments (r_ℓ, d_i) and (r_m, d_j) between drones d_i and d_j. Such swap yields the sub-region assignments (r_ℓ, d_j) and (r_m, d_i). An example of neighboring solution (for $\sigma = 2$) in the sub-region swap neighborhood is given in Figure 5.9, where the assignments (r_2, d_3) and (r_1, d_4) are swapped between drones d_3 and d_4.

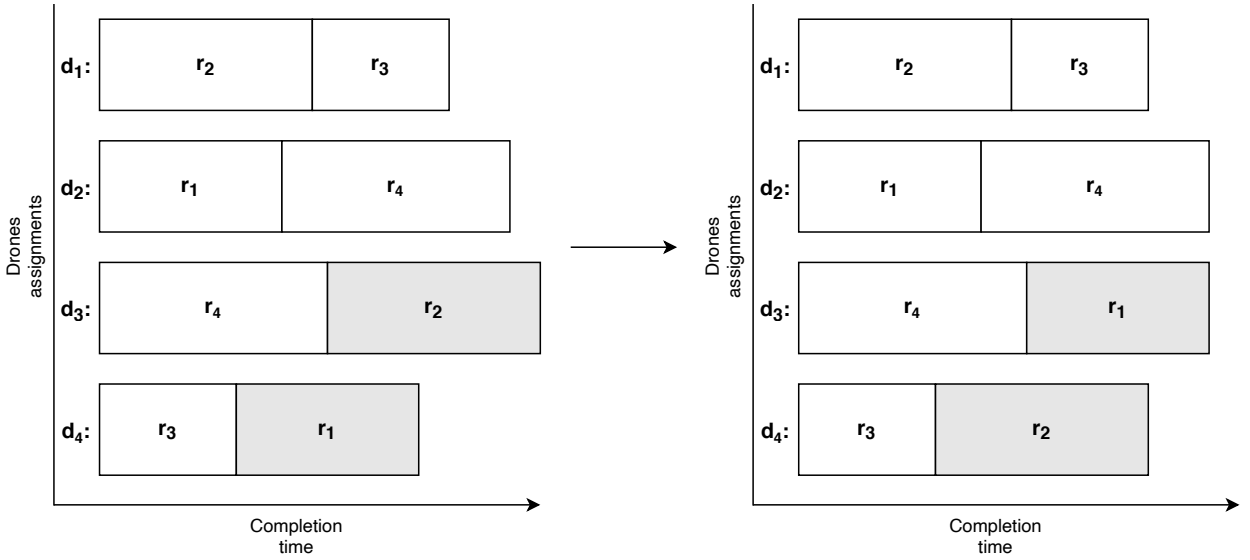


Figure 5.9 Neighboring solution (right) obtained from a spatial partition tree (left) in the sub-region swap neighborhood for $\sigma = 2$

5.4.7 Computation of the processing time and transmission data

In order to efficiently explore the neighborhoods during VNS' local descent, it is important to adequately compute the photo processing time of a sub-region and the amount of data involved in the data transfers between drones. Given the boundaries of a sub-region and the drone $d \in \bar{D}$ dealing with the reconstruction of that sub-region, this information can be computed in constant time as follows.

The computation of the photo processing time of a sub-region is based on the sets of photos

S_c^{left}, S_c^{right}, S_ℓ^{up}, S_ℓ^{down} for $c \in C$ and $\ell \in L$. Thus,

$$S_c^{left} = \{p \in P | lng(p) < lng(c)\}, \tag{5.22}$$
$$S_c^{right} = \{p \in P | lng(p) > lng(c)\}, \tag{5.23}$$
$$S_\ell^{up} = \{p \in P | lat(p) < lat(\ell)\}, \tag{5.24}$$
$$S_\ell^{down} = \{p \in P | lat(p) > lat(\ell)\}, \tag{5.25}$$

where $lat()$ and $lng()$ stand for the latitude and longitude respectively. Thus, S_c^{left} encompasses all photos on the left of c whereas S_c^{right} contains the photos on the right of c. Similarly, S_ℓ^{up} groups the photos above ℓ and S_ℓ^{down} the photos below ℓ.

We define $T(\overline{P})$ as the overall processing time of all photos in a set $\overline{P} \subseteq P$ such that

$$T(\overline{P}) = \sum_{p \in \overline{P}} \lambda_p. \tag{5.26}$$

Finally, let $c_<$, $c_>$, $\ell_\vee$, $\ell_\wedge$ represent, respectively, the left, the right, the bottom (inferior), and the top (superior) boundaries of a sub-region r. The photo processing time T^r of a sub-region r delimited by $c_<$, $c_>$, $\ell_\vee$, and $\ell_\wedge$ is then given by

$$T^r = T\left(\{p \in P \mid lng(c_<) \leq lng(p) \leq lng(c_>)\ ,\ lat(\ell_\vee) \leq lat(p) \leq lat(\ell_\wedge)\}\right).$$

Equivalently, one may subtract the processing times of the photos outside the sub-region r from the overall processing time $T(P)$, i.e.,

$$T^r = T(P) - T(S_{c_<}^{left} \cup S_{c_>}^{right} \cup S_{\ell_\vee}^{down} \cup S_{\ell_\wedge}^{up}). \tag{5.27}$$

Now, for each pair $(c, \ell) \in C \times L$, let us define $Q_{c\ell}^1$, $Q_{c\ell}^2$, $Q_{c\ell}^3$, $Q_{c\ell}^4$, respectively, as the set of photos lying in the first, second, third, and fourth quadrants concerning $lng(c)$ and $lat(\ell)$, which are expressed as

$$Q_{c\ell}^1 = \{p \in P \mid lng(p) > lng(c), lat(p) > lat(\ell)\}, \tag{5.28}$$
$$Q_{c\ell}^2 = \{p \in P \mid lng(p) < lng(c), lat(p) > lat(\ell)\}, \tag{5.29}$$
$$Q_{c\ell}^3 = \{p \in P \mid lng(p) < lng(c), lat(p) < lat(\ell)\}, \text{ and} \tag{5.30}$$
$$Q_{c\ell}^4 = \{p \in P \mid lng(p) > lng(c), lat(p) < lat(\ell)\}. \tag{5.31}$$

Thus, it is possible to express equation (5.27) as

$$\begin{aligned} T^r = T(P) - T(S^{left}_{c<}) - T(S^{right}_{c>}) - T(S^{down}_{\ell\vee}) - T(S^{up}_{\ell\wedge}) \\ + T(Q^1_{c>\ell\wedge}) + T(Q^2_{c<\ell\wedge}) + T(Q^3_{c<\ell\vee}) + T(Q^4_{c>\ell\vee}). \end{aligned} \tag{5.32}$$

Given the boundaries of a sub-region r, equation (5.32) allows to compute T^r based only on the processing times of $S^{left}_{c<}$, $S^{right}_{c>}$, $S^{down}_{\ell\vee}$, $S^{up}_{\ell\wedge}$, $Q^1_{c>\ell\wedge}$, $Q^2_{c<\ell\wedge}$, $Q^3_{c<\ell\vee}$, $Q^4_{c>\ell\vee}$, and $T(P)$. By precomputing these values in time $O(|C| \cdot |L| \cdot |P|)$ (equiv. $O(|P|^3)$), there is no need of keeping in memory the index of photos belonging to r. Thus, the photo processing time of any sub-region r can be computed by (5.32) in constant time $O(1)$.

Likewise, the amount of data μ^{hd}_r on a certain data transmission $f^{hd} \in F$ concerning only the sub-region r can be computed by a closed form expression. Given the transmission demand f^{hd} and a set of photos $\overline{P}$, let $\mu^{hd}(\overline{P})$ be the amount of data in $\overline{P}$ exchanged from drone h to the drone d, where d is the photo-processing drone of $\overline{P}$, which is expressed as

$$\mu^{hd}(\overline{P}) = \sum_{p \in \overline{P}} \theta_{hp} \mu_p. \tag{5.33}$$

Since the boundaries of the sub-region r and the transmission demand f^{hd} are known, the μ^{hd}_r can be computed by the equation

$$\begin{aligned} \mu^{hd}_r = &\ \mu^{hd}\left(\{p \in P \mid lng(c_<) \leq lng(p) \leq lng(c_>)\ ,\ lat(\ell_\vee) \leq lat(p) \leq lat(\ell_\wedge)\}\right) \\ \mu^{hd}_r = &\ \mu^{hd}(P) - \mu^{hd}(S^{left}_{c<} \cup S^{right}_{c>} \cup S^{down}_{\ell\vee} \cup S^{up}_{\ell\wedge}) \\ \mu^{hd}_r = &\ \mu^{hd}(P) - \mu^{hd}(S^{left}_{c<}) - \mu^{hd}(S^{right}_{c>}) - \mu^{hd}(S^{down}_{\ell\vee}) - \mu^{hd}(S^{up}_{\ell\wedge}) \\ &+ \mu^{hd}(Q^1_{c>\ell\wedge}) + \mu^{hd}(Q^2_{c<\ell\wedge}) + \mu^{hd}(Q^3_{c<\ell\vee}) + \mu^{hd}(Q^4_{c>\ell\vee}). \end{aligned} \tag{5.34}$$

The terms within the equation (5.34) can all be precomputed in time $O(|D|^2 \cdot |C| \cdot |L| \cdot |P|)$ operations (equiv. $|D|^2 \cdot |P|^3$). Thus, given the boundaries of the sub-region r, the amount of data exchanged in f^{hd} is obtained on constant time.

Note that both equations (5.32) and (5.34) are suited to the way sub-regions are represented by a spatial partition tree. It only suffices to know the boundaries of a sub-region to query the terms in (5.32) and (5.34).

5.4.8 Proposed VNS heuristic

According to Algorithm 3, we need to define how to create an initial solution as well as the diversification and intensification steps. These are described in the following.

Initial solution

The initial solution is created by (i) randomly building a spatial-partition tree with $\bar{D}$ sub-regions (as defined in Section 5.4.2); and by (ii) randomly assigning the obtained sub-regions to the set of 3D-capable drones $\bar{D}$ according to the reliability factor σ. Thus, each 3D-capable drone has at least one sub-region to process.

Diversification step

The diversification step uses the *sub-tree reconstruction* neighborhood. The VNS parameter k establishes the depth from which a node will be selected in the spatial-partition tree T. More precisely, a node from depth $D(T)-k$ is randomly selected to be reconstructed. Finally, we adopt a k_{step} equal to one and k_{max} equal to the $D(T)$ of the current best solution.

Intensification step

Our local descent method is a Variable Neighborhood Descent (VND) [146]. The proposed VND minimizes $T_{\max}$ by sequentially exploring the *sub-region transfer*, *sub-region swap*, and *splitting hyperplane reallocation* neighborhoods. In order to accelerate the local searches, we limit neighborhood exploration to transfer/swapping/reallocation of sub-regions assigned to the set of drones, named $D_{\max}$, holding the incumbent longest total processing time. That is, $D_{\max}$ comprises the drones whose total photo-processing time is equal to the makespan ($T_{\max}$). By denoting $R_{\hat{D}}$ as the set of all sub-regions assigned to drones $d \in \hat{D} \subseteq D$, the local searches are implemented in such a way that

i $\mathcal{N}_1$ - **sub-region transfer**: switches sub-regions $r \in R_{D_{\max}}$ to other drones $d \in \bar{D} - D_{\max}$. The size of this neighborhood is bounded by $\Theta(|R_{D_{\max}}| \times |\bar{D} - D_{max}|)$.

ii $\mathcal{N}_2$ - **sub-region swap**: only considers sub-region swaps (r_i, r_j) such that $r_i \in R_{D_{\max}}$. Consequently, this neighborhood has $\Theta(|R_{D_{\max}}| \times |R_{\bar{D}-D_{\max}}|)$ neighboring solutions.

iii $\mathcal{N}_3$ - **splitting hyperplane reallocation**: for each $n \in N_{D_{\max}}$, where $N_{D_{\max}}$ is composed by the parent nodes of leaf nodes representing the sub-regions $r \in R_{D_{\max}}$, this neighborhood reallocates the splitting hyperplanes (i.e., their positions and/or orientations) of non-leaf nodes in the sub-tree rooted at n (denoted T_n). Let us define L_n and C_n as the set of enclosed latitudes on axis L and longitudes on axis C at node n, respectively. In the worst case, the size of this neighborhood is limited by $O\left(\sum_{n \in N_{D_{\max}}} (|L_n| + |C_n|)^{|N_{T_n}|}\right)$, where N_{T_n} is the set of non-leaf nodes in the sub-tree T_n.

Employing such sequence of neighborhoods grants higher priority to adjustments in the sub-region to drone assignments, which are encompassed in smaller neighborhoods that cause less changes to solution's structure.

Besides minimizing the latest processing time $T_{\max}$, the local searches seek solutions respecting the maximum allowed transmission time $\hat{T}$. Thus, the transmission times t^{hd} of all active transmission demands $f^{hd} \in F$ must be at most $\hat{T}$ (according to constraints (5.3)). This means that we need to check whether

$$t^{hd} \leq \hat{T} \qquad \forall f^{hd} \in F \tag{5.35}$$

in order to define if a neighboring solution is feasible. Further, the transmission times t^{hd} are computed by the equation

$$t^{hd} = \frac{\mu^{hd}}{\phi^{hd}} = \frac{\sum_{r \in R_d} \mu_r^{hd}}{\phi^{hd}}. \tag{5.36}$$

As described in Section 5.4.7, likewise the processing times T^r, it is possible to compute μ_r^{hd} in constant time. However, computing ϕ^{hd} means solving an MMF rate allocation problem. Due to the fixed tree topology, one can solve an MMF rate allocation problem in $O(|D|^2 \times (|D|-1))$ employing the *water filling* method [79].

Algorithm 4 describes the local search used in this paper. It uses a first improvement exploration strategy, which pursues the first improvement direction in the local descent method. Limited computational experiments revealed that a best improving strategy in which the local descent used the best estimated improving direction was not better than its first improving counterpart. The algorithm requires an initial solution x and a neighborhood $\mathcal{N}_t$ as input and return the local optimum w.r.t. $\mathcal{N}_t$. Also, let us define Φ as the hash table mapping a set of active transmission demands $\bar{F}$ to its respective MMF rate allocation ϕ. This data structure accelerates the local search since it prevents the method from solving the same MMF rate allocation problem more than once for the same set of active transmission demands. Finally, let us define the function $T_{\max}(\cdot)$ to return the makespan (i.e., latest completion time) of a solution in its argument.

Regarding Algorithm 4, step 1 initializes the flag *Stop* to false. While the local optimum is not reached (steps 2-18), the local search explores the neighboring solutions $x' \in \mathcal{N}_t(x)$ (steps 4-17). If for neighboring solution x' $T_{\max}(x') < T_{\max}(x)$ (steps 5-16), the algorithm proceeds to checking whether x' has feasible transmission times associated with it according to expression (5.35). First, it identifies the active demands $\bar{F}$ in x' (step 6). A demand between a pair of drones is active if there exists any data to be exchanged between them (Section 5.2.1). Thus, step 6 checks whether $\mu^{hd} = \sum_{r \in R_d} \mu_r^{hd} > 0$ for each $f^{hd} \in F$. Then, it

Algorithm 4 Local search using first improvement search strategy.

Require: x and $\mathcal{N}_t$
1: $Stop \leftarrow false$;
2: **while** not $Stop$ **do**
3: $Stop \leftarrow True$;
4: **for all** $x' \in \mathcal{N}_t(x)$ **do**
5: **if** $T_{\max}(x') < T_{\max}(x)$ **then**
6: identify the active demands $\bar{F}(x')$;
7: **if** $\nexists\ \Phi(\bar{F}(x'))$ **then**
8: $\Phi(\bar{F}(x')) \leftarrow$ *water filling algorithm*$(\bar{F}(x')$;
9: **end if**
10: $\bar{T} \leftarrow$ compute communication delays $(\Phi(\bar{F}(x')), \mu(x'))$;
11: **if** $t^{hd} \leq \hat{T} \quad \forall t^{hd} \in \bar{T}$ **then**
12: $Stop \leftarrow False$;
13: $x \leftarrow x'$;
14: **break**;
15: **end if**
16: **end if**
17: **end for**
18: **end while**
19: **return** x;

computes the MMF rate allocation problem for $\bar{F}$ (through the *water filling algorithm*) if and only if $\bar{F}$ is not already a stored key of Φ (steps 7-9). The set of communication delays $\bar{T} = \{t^{hd} | f^{hd} \in F\}$ is computed in step 10 according to the equation (5.36) given the transmission rates $\Phi(\bar{F}(x'))$ and the amount of data $\mu = \{\mu^{hd} | f^{hd} \in F\}$ in solution x'. If x' has all communication delays $t^{hd} \in \bar{T}$ smaller or equal to $\hat{T}$ (steps 11-15), the local search (i) marks x as sub-optimum (step 12); (ii) sets x' as the new x (step 13), and (iii) breaks the for loop to explore the neighborhood w.r.t. the new incumbent x (step 14). Once reached, the local optimum concerning $\mathcal{N}_t$ is returned in step 19.

Finally, the proposed VND is summarized in Algorithm 5. In the steps 2-5 of the algorithm, we choose the initial value of t according to the reliability factor (σ) used. It is set to $t = 1$ for $\sigma > 1$ and $t = 2$ otherwise. That is, the local search applying the *sub-region transfer* neighborhood ($\mathcal{N}_1$) is used only for instances with $\sigma > 1$ since it cannot find better solutions for $\sigma = 1$. This is due to the fact that each 3D-capable drone must have at least one sub-region assigned to it. The VND proceeds by applying the local searches according to the current neighborhood $\mathcal{N}_t$ until it reaches the local optimum w.r.t. all neighborhoods in $\mathcal{N}$ (steps 6-17). Each new solution x'' (obtained at step 7 by Algorithm 4) with better $T_{\max}$ than x' is kept and t is reset likewise steps 2-5 (steps 8-13). Otherwise, t is increased by one

Algorithm 5 Variable Neighborhood Descend (VND) for the CAPsac

1: $\mathcal{N} = \{$*sub-region transfer*, *sub-region swap*, *splitting hyperplane reallocation* $\}$
2: $t \leftarrow 1$
3: **if** $\sigma = 1$ **then**
4: $\quad t \leftarrow 2$
5: **end if**
6: **repeat**
7: $\quad x'' \leftarrow$ perform LocalSearch(x', $\mathcal{N}_t$);
8: $\quad$ **if** $T_{\max}(x'') < T_{\max}(x')$ **then**
9: $\quad\quad x' \leftarrow x''$;
10: $\quad\quad t \leftarrow 1$
11: $\quad\quad$ **if** $\sigma = 1$ **then**
12: $\quad\quad\quad t \leftarrow 2$
13: $\quad\quad$ **end if**
14: $\quad$ **else**
15: $\quad\quad t \leftarrow t + 1$;
16: $\quad$ **end if**
17: **until** $t > 3$

(steps 14-16), and the algorithm iterates.

5.5 Computational experiments

Our experimental analysis aims to evaluate (i) the effectiveness of the decomposition strategy; (ii) the effectiveness of employing different neighborhoods in our variable neighborhood method; and (iii) the performance of the proposed methods and sensitivity regarding both the reliability factor σ and the maximum transmission time allowed $\hat{T}$. We used CPLEX v12.10 as general-purpose integer linear programming solver and C++ as programming language compiled with gcc v5.4.0. All experiments were carried exploiting a single core on a machine powered by an Intel E5-2683 v4 Broadwell 2.1GHz with 20Gb of RAM, and running the CentOS Linux 7.5.1804 OS.

5.5.1 Instances and notation adopted

Our experimental analysis is carried on the instances proposed in [124] (available at `https://github.com/ds4dm/CAPsac`). All instances are based on realistic data and comprise two scenarios: (i) one in which all photos are processed in the same amount of time λ, named *unweighted*; and (ii) another one in which each photo has a different processing time λ_p, named *weighted*. The name of the instances follows the notation X-P$\,Y$D$Z\%\bar{D}\,W$ where "X"

is "u" for the unweighted instances and "w" for the weighted instances, "Y" stands for the number of photos in the instance, "Z" specifies the number of drones in the swarm, and "W" informs the percentage of drones able to perform 3D reconstruction. Remark that the number of 3D-capable drones ($|\bar{D}|$) is always equal to $\left\lfloor Z \times \frac{W}{100} \right\rfloor$. The characteristics of the tested instances are listed in Table 5.2.

Optimal makespans are obtained by solving the CAPsac formulation [124] with CPLEX within one day of execution. Whenever CPLEX is not able to prove optimality within this time horizon, the best feasible solution obtained in [124] is reported instead, which is indicated by an "$*$" in the tables. Average and best results are then computed according to that best known solution.

5.5.2 Effectiveness of the decomposition strategy

This section assesses the effectiveness of the decomposition strategy exploited by the proposed decomposition-based heuristic.

Tables 5.3 and 5.4 report for each instance (named under column "Instance") the following values regarding the optimal (or best known) solutions:

- the smallest cardinality found among the sub-regions (column n^*_ℓ);
- the largest cardinality found among the sub-regions (column n^*_u);
- the cardinality of the sub-region which yields the makespan (column n^*_{mks});
- the associated makespan value (column OPT).

Further, we report the percentage deviations (columns "Dev.(%)") achieved by the complete $rCAPsac$ formulation and the decomposition method with respect to the best known makespan. We also report the total computing CPU times spent by both methods under column ("sec"). Finally, concerning the decomposition-based heuristic, we report the last interval $[n_\ell, n_u]$ used before the algorithm is halted by its stopping condition. For all experiments, we set a time limit of 2 hours for the MILP formulation solved by CPLEX in each

Table 5.2 Characteristics of the tested instances.

Images(P)	200, 400, 500, 750, 1000
Drones(D)	5, 7, 10, 15
%3D-capable drones($\%\bar{D}$)	50%, 70%, 90%

iteration of the decomposition-based heuristic, and 24 hours as maximum time for solving rCAPsac by CPLEX. A trace ("−") is reported whenever CPLEX is not able to find a feasible solution within this time limit.

Table 5.3 Percentage deviations of the complete *rCAPsac* formulation and the decomposition method when solving unweighted instances.

$\sigma = 1, \hat{T} = \infty$, stop condition= 86400$s$				*rCAPsac*			Decomposition			
Instance	n^*_ℓ	n^*_u	n^*_{mks}	OPT	Dev.(%)	sec.	n_ℓ	n_u	Dev.(%)	sec.
u-P200D5%$\bar{D}$70	65	70	70	1870.40	0.00	179.37	64	72	0.00	0.28
u-P200D7%$\bar{D}$50	60	70	70	1870.40	0.00	548.55	64	72	0.00	0.50
u-P400D5%$\bar{D}$70	132	135	135	3607.20	0.00	6996.32	128	140	0.00	1.04
u-P400D7%$\bar{D}$50	130	135	135	3607.20	0.00	11355.80	128	140	0.00	1.55
u-P200D5%$\bar{D}$90	50	50	50	1336.00	0.00	244.25	49	51	0.00	0.11
u-P200D7%$\bar{D}$70	50	50	50	1336.00	0.00	256.03	49	51	0.00	0.23
u-P400D5%$\bar{D}$90	100	100	100	2672.00	0.00	9346.54	99	102	0.00	0.46
u-P400D7%$\bar{D}$70	100	100	100	2672.00	0.00	45059.71	99	102	0.00	0.55
u-P200D10%$\bar{D}$50	40	40	40	1068.80	0.00	1838.53	39	42	0.00	1.46
u-P400D10%$\bar{D}$50	80	80	80	2137.60	0.00	43171.40	78	81	0.00	5.44
u-P200D7%$\bar{D}$90	30	34	34	908.48	0.00	1901.86	28	38	0.00	8.00
u-P400D7%$\bar{D}$90	60	68	68	*1816.96	32.35	86400.00	64	70	1.47	13.28
u-P200D10%$\bar{D}$70	24	30	30	*801.60	33.33	86400.00	27	32	0.00	115.18
u-P400D10%$\bar{D}$70	50	60	60	*1603.20	300.00	86400.00	54	63	-3.33	2013.98
u-P500D15%$\bar{D}$50	70	72	72	1923.84	-	86400.00	69	75	0.00	333.59
u-P750D15%$\bar{D}$50	108	117	117	*3126.24	-	86400.00	104	110	-7.69	262.73
u-P1000D15%$\bar{D}$50	143	144	144	*3847.68	-	86400.00	136	145	0.00	7434.67
u-P200D10%$\bar{D}$90	18	24	24	*641.28	8.33	86400.00	21	25	0.00	3366.98
u-P400D10%$\bar{D}$90	42	45	45	1202.40	-	86400.00	42	46	0.00	3262.06

The decomposition strategy proves effective and yields equivalent or better deviation values than solving the complete *rCAPsac* formulation by CPLEX within the defined time limit. In fact, it is able to find a best known makespan for several instances, as indicated by the negative deviations in the table. However, for a couple of instances, i.e., ("$u-P400D7\%D90$", "$w-P200D7\%D50$", "$w-P400D10\%D50$", "$w-P200D10\%D70$"), the decomposition algorithm is not able to attain the best known solutions obtained in [124]. For those cases — with percentage deviation greater than zero, we observe that the intervals $[n_\ell, n_u]$ must be increased to allow subsets of photos whose cardinalities are out of the defined bounds. For example, for instance $u-P400D7\%D90$, the best known solution contains a sub-region with 60 photos while the last iteration of the decomposition method was executed for $[n_\ell, n_u] = [64, 70]$. Finally, the decomposition method demands significant smaller execution times than solving the complete *rCAPsac* formulation by CPLEX as well as, in general, the formulations in [124].

Table 5.4 Percentage deviations of the complete *rCAPsac* formulation and the decomposition method when solving weighted instances.

$\sigma = 1, \hat{T} = \infty$, stop condition= 86400$s$					*rCAPsac*				Decomposition	
Instance	n^*_ℓ	n^*_u	n^*_{mks}	OPT	Dev.(%)	sec.	n_ℓ	n_u	Dev.(%)	sec.
w-P200D5%$\bar{D}$70	65	70	70	1886.98	0.00	242.54	64	72	0.00	0.36
w-P200D7%$\bar{D}$50	60	70	70	1885.31	0.00	784.93	64	72	3.26	0.39
w-P400D5%$\bar{D}$70	130	135	135	3735.32	0.00	10209.88	128	140	0.00	1.21
w-P400D7%$\bar{D}$50	130	135	135	3773.12	0.00	5987.39	128	140	0.00	1.93
w-P200D5%$\bar{D}$90	50	50	50	1409.48	0.00	212.44	49	51	0.00	0.30
w-P200D7%$\bar{D}$70	50	50	50	1399.82	0.00	802.39	49	51	0.00	0.56
w-P400D5%$\bar{D}$90	100	100	100	2812.93	0.00	21189.45	99	102	0.00	0.57
w-P400D7%$\bar{D}$70	100	100	100	2791.66	0.00	40872.49	99	102	0.00	1.49
w-P200D10%$\bar{D}$50	40	40	40	1121.89	0.00	9106.51	39	42	0.00	13.92
w-P400D10%$\bar{D}$50	72	88	88	2290.48	0.00	40443.02	78	81	1.15	34.01
w-P200D7%$\bar{D}$90	30	35	35	944.06	0.00	8669.90	28	38	0.00	17.77
w-P400D7%$\bar{D}$90	60	75	75	*1909.18	19.17	86400.00	63	72	-0.94	25.17
w-P200D10%$\bar{D}$70	26	30	30	820.40	0.00	62142.01	27	32	0.29	1086.76
w-P400D10%$\bar{D}$70	56	63	60	*1671.12	123.18	86400.00	55	62	-2.30	7232.05
w-P500D15%$\bar{D}$50	66	78	72	*2028.11	-	86400.00	69	75	-2.40	1791.89
w-P750D15%$\bar{D}$50	96	119	117	*3138.73	-	86400.00	102	112	-4.91	2562.77
w-P1000D15%$\bar{D}$50	132	160	160	*4343.26	-	86400.00	135	147	-9.80	7762.67
w-P200D10%$\bar{D}$90	18	24	24	*665.54	20.03	86400.00	19	27	-2.85	15655.03
w-P400D10%$\bar{D}$90	40	50	50	*1302.82	-	86400.00	39	49	-2.89	9645.66

5.5.3 Effectiveness of the neighborhoods

To assess the sequence of neighborhoods in our proposed VND, we compare the performance of distinct neighborhood configurations — $\mathcal{N}_1$, $\mathcal{N}_1 + \mathcal{N}_2$, $\mathcal{N}_1 + \mathcal{N}_3$, $\mathcal{N}_1 + \mathcal{N}_2 + \mathcal{N}_3$ — within our proposed VNS method, where $\mathcal{N}_1$ refers to the *sub-region transfer* neighborhood, $\mathcal{N}_2$ refers to *sub-region swap* neighborhood, and $\mathcal{N}_3$ refers to the *splitting hyperplane reallocation* neighborhood.

Tables 5.5 and 5.6 report the results of the unweighted and weighted instances, respectively. For each "Instance" and reliability factor "σ", the column "MIN." presents the shortest makespan found across all neighborhood configurations. The average percentage deviation from "MIN." achieved by each neighborhood configuration is reported in columns "Avg(%)".

The results are computed for 20 executions of the VNS-based heuristics within 300s of CPU execution time. These experiments are conducted over different reliability factors $\sigma > 1$ so as that the *sub-region transfer* neighborhood could be evaluated. Moreover, we do not present results for instances with fewer than five 3D-capable drones since all neighborhood configurations performed equivalently.

Table 5.5 Percentage deviations of employing distinct neighborhoods in the VND when solving unweighted instances.

$\hat{T}=\infty$, stop condition= $300s$			VNS			
			$\mathcal{N}_1$	$\mathcal{N}_1+\mathcal{N}_2$	$\mathcal{N}_1+\mathcal{N}_3$	$\mathcal{N}_1+\mathcal{N}_2+\mathcal{N}_3$
Instance	σ	MIN.	Avg.(%)	Avg.(%)	Avg.(%)	Avg.(%)
u-P200D10%$\bar{D}$50	2	2137.60	**0.00**	**0.00**	**0.00**	**0.00**
	3	3206.40	**0.00**	**0.00**	**0.00**	**0.00**
	4	4275.20	**0.00**	**0.00**	**0.00**	**0.00**
u-P400D10%$\bar{D}$50	2	4275.20	**0.00**	**0.00**	**0.00**	**0.00**
	3	6412.80	**0.00**	**0.00**	**0.00**	**0.00**
	4	8550.40	**0.00**	**0.00**	**0.00**	**0.00**
u-P200D7%$\bar{D}$90	2	1790.24	**0.00**	**0.00**	**0.00**	**0.00**
	3	2672.00	**0.00**	**0.00**	**0.00**	**0.00**
	4	3580.48	**0.00**	**0.00**	**0.00**	**0.00**
u-P400D7%$\bar{D}$90	2	3580.48	**0.00**	**0.00**	**0.00**	**0.00**
	3	5344.00	**0.00**	**0.00**	**0.00**	**0.00**
	4	7134.24	**0.00**	**0.00**	**0.00**	**0.00**
u-P200D10%$\bar{D}$70	2	1549.76	0.86	0.09	**0.00**	**0.00**
	3	2297.92	0.64	0.35	0.35	**0.06**
	4	3072.80	**0.04**	0.17	0.09	**0.04**
u-P400D10%$\bar{D}$70	2	3072.80	0.43	0.52	0.04	**0.00**
	3	4595.84	0.52	1.98	0.23	**0.12**
	4	6118.88	0.39	0.44	0.39	**0.31**
u-P500D15%$\bar{D}$50	2	3847.68	0.42	0.62	**0.00**	**0.00**
	3	5744.80	0.56	1.23	0.23	**0.05**
	4	7641.92	0.89	0.77	0.70	**0.61**
u-P750D15%$\bar{D}$50	2	5744.80	1.02	1.26	**0.44**	0.47
	3	8630.56	1.24	0.46	0.26	**0.20**
	4	11489.60	0.64	0.42	0.23	**0.16**
u-P1000D15%$\bar{D}$50	2	7641.92	1.17	1.01	0.65	**0.52**
	3	11462.88	1.42	0.65	**0.37**	1.25
	4	15283.84	0.62	0.64	0.52	**0.43**
u-P200D10%$\bar{D}$90	2	1202.40	0.56	0.11	**0.00**	0.11
	3	1790.24	**0.00**	**0.00**	**0.00**	**0.00**
	4	2378.08	1.12	1.01	1.12	**0.96**
u-P400D10%$\bar{D}$90	2	2404.80	0.22	0.11	**0.00**	**0.00**
	3	3580.48	**0.00**	**0.00**	**0.00**	**0.00**
	4	4756.16	0.79	0.56	0.53	**0.45**
		Average	0.41	0.38	0.19	**0.17**

Table 5.6 Percentage deviations of employing distinct neighborhoods in the VND when solving weighted instances.

$\hat{T}=\infty$, stop condition= $300s$			VNS			
			$\mathcal{N}_1$	$\mathcal{N}_1+\mathcal{N}_2$	$\mathcal{N}_1+\mathcal{N}_3$	$\mathcal{N}_1$+$\mathcal{N}_2$+$\mathcal{N}_3$
Instance	σ	MIN.	Avg.(%)	Avg.(%)	Avg.(%)	Avg.(%)
	2	2212.64	0.18	0.16	0.11	**0.06**
w-P200D10%$\bar{D}$50	3	3314.91	0.16	0.18	**0.15**	0.16
	4	4436.74	0.03	0.02	**0.02**	**0.02**
	2	4433.53	0.19	0.22	**0.08**	0.09
w-P400D10%$\bar{D}$50	3	6648.11	0.22	0.24	**0.13**	0.15
	4	8901.19	**0.01**	0.03	0.03	0.02
	2	1816.30	0.14	0.16	0.04	**0.03**
w-P200D7%$\bar{D}$90	3	2724.19	0.00	0.00	**0.00**	**0.00**
	4	3632.45	0.06	0.06	0.03	**0.02**
	2	3697.18	0.11	0.11	**0.02**	**0.02**
w-P400D7%$\bar{D}$90	3	5545.48	**0.00**	**0.00**	**0.00**	**0.00**
	4	7394.18	0.06	0.06	**0.03**	**0.03**
	2	1577.87	0.97	0.84	0.40	**0.30**
w-P200D10%$\bar{D}$70	3	2365.78	0.63	0.49	0.43	**0.26**
	4	3155.85	0.66	0.40	0.34	**0.33**
	2	3164.05	0.65	0.91	0.47	**0.33**
w-P400D10%$\bar{D}$70	3	4747.60	0.52	0.35	0.37	**0.18**
	4	6326.53	0.57	0.40	0.35	**0.32**
	2	3902.67	0.79	0.68	**0.27**	**0.27**
w-P500D15%$\bar{D}$50	3	5852.50	1.33	1.18	0.27	**0.21**
	4	7801.65	0.72	0.54	0.31	**0.22**
	2	5843.52	0.67	0.99	0.28	**0.27**
w-P750D15%$\bar{D}$50	3	8762.21	0.58	0.44	0.32	**0.24**
	4	11679.59	0.70	0.55	0.32	**0.28**
	2	7724.25	1.08	1.04	**0.34**	0.37
w-P1000D15%$\bar{D}$50	3	11590.54	0.58	1.11	0.32	**0.21**
	4	15451.40	0.58	0.36	**0.19**	0.26
	2	1230.42	1.10	1.08	**0.66**	0.74
w-P200D10%$\bar{D}$90	3	1838.98	0.13	0.16	**0.02**	**0.02**
	4	2454.44	0.51	0.28	0.42	**0.22**
	2	2463.59	1.05	0.69	0.62	**0.48**
w-P400D10%$\bar{D}$90	3	3688.59	0.10	0.13	**0.01**	**0.01**
	4	4922.46	0.56	0.26	0.30	**0.17**
		Average	0.47	0.43	0.23	**0.19**

We can observe that the *splitting hyperplane reallocation* neighborhood ($\mathcal{N}_3$) is an important neighborhood for improving the quality of the solutions obtained by our VNS. This is not surprising given that it is the only one able to change photo-processing times of sub-regions. The best results are usually obtained when including the neighborhoods $\mathcal{N}_1$ and $\mathcal{N}_2$ before $\mathcal{N}_3$ (i.e., $\mathcal{N}_1+\mathcal{N}_2+\mathcal{N}_3$ configuration). This happens since the $\mathcal{N}_1+\mathcal{N}_2+\mathcal{N}_3$ configuration adjusts both the sub-region to drone assignments and the sub-region's photo processing times. In fact, this configuration yields the best average deviations for 57 out of 66 cases (including unweighted and weighted instances), i.e., 86.36% of the cases. Therefore, we will consider hereafter the configuration $\mathcal{N}_1 + \mathcal{N}_2 + \mathcal{N}_3$ within our VNS heuristic.

5.5.4 Comparison of the proposed methods

This section evaluates the performance of the proposed heuristics as well as their sensitivity with respect to the reliability factor σ and to the maximum allowed transmission time $\hat{T}$. For the decomposition-based heuristic (named "Decomposition"), we set a time limit of 2 hours for the MILP formulation solved in each iteration and the overall execution time of the decomposition is limited to 24 hours (i.e., 86,400s). In contrast, the VNS was allowed to run for 300 seconds.

The makespan values in seconds (columns "$T_{\max}$") obtained by the proposed methods for the unweighted and weighted instances are reported in Tables 5.7 and 5.8, respectively. Results are reported for each instance for various values of σ, ranging from 1 to $|\bar{D}| - 1$ (presented under column labelled "σ"). A trace ("$-$") is placed whenever a method cannot find a feasible solution until before being halted. For this first series of experiments the communication constraints were relaxed, i.e., $\hat{T}$ is set to a value sufficiently large. Both tables report the instance employed at each row (column "Instance"), the optimal/best known makespan for an instance configuration (column "OPT"), the CPU time (in seconds) spent by CPLEX and the decomposition heuristic when solving an instance (columns "sec."), the best and average makespan values found by the VNS heuristic out of 20 executions (column "Best $T_{\max}$" and "Avg$T_{\max}$", respectively), and the average CPU time (in seconds) spent by the VNS heuristic to reach the solution returned at the end of its execution (column "Avg.(s)").

Table 5.7 Makespan($T_{\max}$) in seconds when solving unweighted instances and varying σ.

parameters: $\hat{T} = \infty$				Decomposition		VNS		
				time limit = 86400s		stop condition = 300s		
Instance	σ	OPT	sec.	$T_{\max}$	sec.	Best $T_{\max}$	Avg.$T_{\max}$	Avg.(s)
u-P200D5%$\bar{D}$70	1	1870.40	15.05	1870.40	0.26	1870.40	1870.40	0.01
	2	3607.20	112.83	3607.20	0.31	3607.20	3607.20	0.01
u-P200D7%$\bar{D}$50	1	1870.40	10.25	1870.40	0.42	1870.40	1870.40	0.01
	2	3607.20	100.99	3607.20	0.59	3607.20	3607.20	0.01
u-P400D5%$\bar{D}$70	1	3607.20	41.22	3607.20	1.03	3607.20	3607.20	0.01
	2	7214.40	539.52	7214.40	1.14	7214.40	7214.40	0.02
u-P400D7%$\bar{D}$50	1	3607.20	68.28	3607.20	1.50	3607.20	3607.20	0.02
	2	7214.40	1637.54	7214.40	1.54	7214.40	7214.40	0.03
u-P200D5%$\bar{D}$90	1	1336.00	7.06	1336.00	0.10	1336.00	1336.00	0.01
	2	2672.00	1680.46	2672.00	0.12	2672.00	2672.00	0.01
	3	4008.00	870.76	4008.00	0.11	4008.00	4008.00	0.50
u-P200D7%$\bar{D}$70	1	1336.00	6.41	1336.00	0.19	1336.00	1336.00	0.04
	2	2672.00	2296.34	2672.00	0.26	2672.00	2672.00	0.01
	3	4008.00	1150.97	4008.00	0.21	4008.00	4008.00	0.71
u-P400D5%$\bar{D}$90	1	2672.00	35.02	2672.00	0.47	2672.00	2672.00	0.04
	2	*5344.00	86400.00	5344.00	0.44	5344.00	5344.00	0.01
	3	8016.00	4593.69	8016.00	0.46	8016.00	8016.00	4.66
u-P400D7%$\bar{D}$70	1	2672.00	59.71	2672.00	0.58	2672.00	2672.00	0.12
	2	*5344.00	86400.00	5344.00	0.76	5344.00	5344.00	0.01
	3	8016.00	5954.75	8016.00	0.79	8016.00	8016.00	6.29
u-P200D10%$\bar{D}$50	1	1068.80	38.94	1068.80	1.46	1068.80	1068.80	0.08
	2	2137.60	20602.10	2137.60	0.71	2137.60	2137.60	0.15
	3	*3286.56	86400.00	**3206.40**	1.79	**3206.40**	**3206.40**	0.17
	4	4275.20	56226.05	4275.20	8.87	4275.20	4275.20	2.21
u-P400D10%$\bar{D}$50	1	2137.60	107.68	2137.60	5.75	2137.60	2137.60	0.14
	2	*4275.20	86400.00	4275.20	2.87	4275.20	4275.20	0.24
	3	*6519.68	86400.00	**6412.80**	3.64	**6412.80**	**6412.80**	0.90
	4	*8550.40	86400.00	8550.40	2.93	8550.40	8550.40	5.27

The table continues on next page

Table 5.7 Makespan($T_{\max}$) in seconds when solving unweighted instances and varying σ (Continued).

parameters: $\hat{T} = \infty$				Decomposition		VNS		
				time limit = 86400s		stop condition = 300s		
Instance	σ	OPT	sec.	$T_{\max}$	sec.	Best $T_{\max}$	Avg.$T_{\max}$	Avg.(s)
u-P200D7%$\bar{D}$90	1	908.48	49.52	908.48	8.12	908.48	908.48	1.73
	2	*1816.96	86400.00	**1790.24**	17.56	**1790.24**	**1790.24**	0.12
	3	*2672.00	86400.00	2672.00	14.76	2672.00	2672.00	0.01
	4	*3607.20	86400.00	**3580.48**	23.89	**3580.48**	**3580.48**	0.46
	5	*4542.40	86400.00	**4488.96**	444.98	**4488.96**	**4526.37**	21.96
u-P400D7%$\bar{D}$90	1	*1816.96	86400.00	1843.68	13.32	1816.96	1816.96	1.11
	2	*3580.48	86400.00	3580.48	14.69	3580.48	3580.48	0.31
	3	*5477.60	86400.00	**5344.00**	15.52	**5344.00**	**5344.00**	0.01
	4	*7748.80	86400.00	**7160.96**	2449.49	**7134.24**	**7134.24**	8.05
	5	*10688.00	86400.00	**8924.48**	64.40	**8924.48**	**9018.00**	63.95
u-P200D10%$\bar{D}$70	1	*801.60	86400.00	801.60	108.54	801.60	801.60	1.85
	2	*1603.20	86400.00	**1549.76**	1136.86	**1549.76**	**1549.76**	10.43
	3	-	-	2297.92	30.52	2297.92	2299.26	74.77
	4	-	-	3072.80	55.08	3072.80	3074.14	40.66
	5	*4328.64	86400.00	**3847.68**	597.09	**3847.68**	**3847.68**	17.27
	6	*5718.08	86400.00	**4595.84**	184.89	**4649.28**	**4670.66**	51.14
u-P400D10%$\bar{D}$70	1	*1603.20	86400.00	**1549.76**	2183.17	**1549.76**	**1563.12**	64.66
	2	-	-	3072.80	16065.91	3072.80	3072.80	56.34
	3	-	-	4595.84	296.30	4595.84	4602.52	70.87
	4	-	-	6118.88	7915.51	6118.88	6137.58	34.18
	5	-	-	7668.64	21612.06	7641.92	7686.01	62.77
	6	-	-	9191.68	5877.11	9218.40	9352.00	79.74
u-P500D15%$\bar{D}$50	1	1923.84	16185.19	1923.84	329.20	1923.84	1959.91	97.38
	2	-	-	3847.68	14417.75	3847.68	3847.68	13.77
	3	-	-	5744.80	21621.88	5744.80	5747.47	76.45
	4	-	-	7641.92	14412.51	7641.92	7688.68	36.03
	5	-	-	9565.76	14413.03	9565.76	9611.18	27.35
	6	-	-	11489.60	8106.20	11489.60	11696.68	58.85

The table continues on next page

Table 5.7 Makespan($T_{\max}$) in seconds when solving unweighted instances and varying σ (End).

parameters: $\hat{T} = \infty$				Decomposition		VNS		
				time limit = 86400s		stop condition = 300s		
Instance	σ	OPT	sec.	$T_{\max}$	sec.	Best $T_{\max}$	Avg.$T_{\max}$	Avg.(s)
u-P750D15%$\bar{D}$50	1	*3126.24	86400.00	**2885.76**	270.72	**2885.76**	**2933.86**	52.65
	2	-	-	5744.80	17315.08	5744.80	5771.52	78.88
	3	-	-	8657.28	14419.11	8630.56	8649.26	91.33
	4	-	-	11462.88	14419.09	11489.60	11508.30	76.75
	5	-	-	14348.64	14421.91	14428.80	14434.14	35.91
	6	-	-	17234.40	14418.32	17234.40	17485.57	37.64
u-P1000D15%$\bar{D}$50	1	*3847.68	86400.00	3847.68	7424.05	3847.68	3950.55	64.74
	2	-	-	7695.36	14439.34	7641.92	7682.00	36.38
	3	-	-	-	86400.00	11489.60	11605.83	67.56
	4	-	-	-	86400.00	15283.84	15349.30	76.41
	5	-	-	19131.52	14433.69	19104.80	19234.39	66.49
	6	-	-	-	86400.00	22979.20	23410.73	112.49
u-P200D10%$\bar{D}$90	1	*641.28	86400.00	641.28	3201.67	641.28	641.28	17.94
	2	-	-	1202.40	3589.18	1202.40	1203.74	10.09
	3	*2030.72	86400.00	**1790.24**	297.44	**1790.24**	**1790.24**	0.15
	4	-	-	2404.80	10993.74	2378.08	2400.79	1.89
	5	-	-	2992.64	1200.84	2992.64	2992.64	12.98
	6	-	-	3580.48	7224.15	3580.48	3580.48	0.21
	7	-	-	4168.32	2382.04	4168.32	4217.75	43.90
	8	-	-	4782.88	7820.39	4809.60	4817.62	13.23
u-P400D10%$\bar{D}$90	1	1202.40	17470.88	1202.40	3376.48	1202.40	1241.14	131.36
	2	-	-	2378.08	903.16	2404.80	2404.80	8.97
	3	-	-	3580.48	262.56	3580.48	3580.48	0.08
	4	-	-	4756.16	991.95	4756.16	4777.54	58.81
	5	-	-	5958.56	255.96	5958.56	5975.93	59.92
	6	-	-	7134.24	862.01	7134.24	7134.24	1.96
	7	-	-	8336.64	183.60	8336.64	8379.39	69.29
	8	-	-	9512.32	240.45	9619.20	9619.20	31.49

Table 5.8 Makespan($T_{\max}$) in seconds when solving weighted instances and varying σ.

parameters: $\hat{T} = \infty$				Decomposition		VNS		
				time limit = 86400s		stop condition = 300s		
Instance	σ	OPT	sec.	$T_{\max}$	sec.	Best $T_{\max}$	Avg.$T_{\max}$	Avg.(s)
w-P200D5%$\bar{D}$70	1	1886.98	15.84	1886.98	0.39	1886.98	1886.98	0.01
	2	3699.67	172.46	3699.67	0.37	3699.67	3699.67	0.02
w-P200D7%$\bar{D}$50	1	1885.31	22.74	1946.84	0.41	1885.31	1885.31	0.03
	2	3740.19	653.34	3740.19	0.47	3740.19	3740.19	0.04
w-P400D5%$\bar{D}$70	1	3735.32	79.99	3735.32	1.20	3735.32	3735.32	0.01
	2	7457.00	1988.79	7458.55	1.22	7458.55	7458.55	0.19
w-P400D7%$\bar{D}$50	1	3773.12	94.42	3773.12	2.47	3773.12	3773.12	0.03
	2	7436.46	1464.54	7436.46	1.53	7436.46	7436.46	0.26
w-P200D5%$\bar{D}$90	1	1409.48	212.44	1409.48	0.26	1409.48	1409.48	0.08
	2	2735.45	34294.09	2736.79	0.33	2735.45	2736.23	0.90
	3	*4148.52	86400.00	4148.52	0.34	4148.52	4148.52	0.78
w-P200D7%$\bar{D}$70	1	1399.82	453.18	1399.82	0.55	1399.82	1399.82	0.10
	2	2724.20	2647.32	2726.87	0.50	2724.20	2725.04	0.40
	3	*4105.25	86400.00	4105.25	0.59	4105.25	4105.25	3.41
w-P400D5%$\bar{D}$90	1	2812.93	925.85	2812.93	0.56	2812.93	2812.93	0.14
	2	*5555.98	86400.00	5568.43	0.75	**5555.12**	**5555.30**	13.20
	3	*8381.36	86400.00	8381.36	0.74	8381.36	8381.36	4.07
w-P400D7%$\bar{D}$70	1	2791.66	605.48	2791.66	1.74	2791.66	2791.66	0.15
	2	*5545.75	86400.00	5555.17	1.37	**5545.49**	**5545.51**	3.73
	3	*8346.83	86400.00	8346.83	1.43	8346.83	8346.83	13.77
w-P200D10%$\bar{D}$50	1	1121.89	3611.52	1121.89	13.61	1121.89	1121.89	42.04
	2	*2222.97	86400.00	**2216.51**	99.52	**2212.64**	**2213.97**	126.59
	3	*3329.75	86400.00	**3314.91**	69.04	**3314.91**	**3320.12**	54.75
	4	*4509.50	86400.00	**4436.74**	19.78	**4436.74**	**4437.46**	76.78
w-P400D10%$\bar{D}$50	1	2290.48	40443.02	2316.78	34.98	2290.48	2290.48	6.50
	2	*4591.18	86400.00	**4445.09**	16.37	**4434.52**	**4437.33**	78.96
	3	*6765.02	86400.00	**6650.29**	19.40	**6648.11**	**6657.81**	86.10
	4	*9938.25	86400.00	**8901.19**	73.87	**8901.19**	**8902.89**	58.79

The table continues on next page

Table 5.8 Makespan($T_{\max}$) in seconds when solving weighted instances and varying σ (Continued).

parameters: $\hat{T} = \infty$				Decomposition		VNS		
				time limit = 86400s		stop condition = 300s		
Instance	σ	OPT	sec.	$T_{\max}$	sec.	Best $T_{\max}$	Avg.$T_{\max}$	Avg.(s)
w-P200D7%$\bar{D}$90	1	944.06	8669.90	944.06	17.64	944.06	944.06	58.91
	2	*1838.27	86400.00	**1818.19**	315.24	**1816.30**	**1816.84**	94.62
	3	*2737.73	86400.00	**2724.26**	193.49	**2724.19**	**2724.19**	151.52
	4	*3784.47	86400.00	**3633.31**	2826.81	**3632.45**	**3633.19**	149.25
	5	*4952.71	86400.00	**4582.94**	2834.41	**4585.65**	**4600.59**	133.66
w-P400D7%$\bar{D}$90	1	*1909.18	86400.00	**1891.23**	54.26	**1891.23**	**1904.24**	125.80
	2	*3922.21	86400.00	**3701.03**	319.12	**3697.34**	**3697.90**	113.25
	3	*5567.01	86400.00	**5545.72**	49.86	**5545.48**	**5545.48**	78.77
	4	*9094.33	86400.00	**7398.95**	7395.50	**7394.18**	**7396.18**	130.84
	5	-	-	9302.92	15918.34	9300.46	9361.46	113.56
w-P200D10%$\bar{D}$70	1	820.40	62142.01	822.78	1015.08	825.06	825.06	81.79
	2	-	-	1577.97	11209.34	1577.87	1582.31	115.51
	3	*2446.63	86400.00	**2372.45**	14413.80	**2365.78**	**2371.93**	149.33
	4	*4168.27	86400.00	**3159.91**	8257.01	**3157.05**	**3165.93**	107.74
	5	*4198.12	86400.00	**3945.45**	22388.35	**3947.86**	**3958.33**	68.83
	6	*5613.12	86400.00	**4763.35**	5832.36	**4763.35**	**4809.49**	58.26
w-P400D10%$\bar{D}$70	1	*1671.12	86400.00	**1632.77**	7249.23	**1633.77**	**1639.62**	114.80
	2	*3303.16	86400.00	**3166.70**	27629.18	**3164.05**	**3175.29**	134.49
	3	-	-	4748.65	21612.15	4747.60	4756.31	127.08
	4	*7080.11	86400.00	**6335.64**	28821.90	**6334.85**	**6347.79**	101.82
	5	*8989.29	86400.00	**7918.13**	21613.88	**7911.42**	**7943.12**	96.72
	6	-	-	9529.57	21616.77	9534.57	9630.75	73.83
w-P500D15%$\bar{D}$50	1	*2028.11	86400.00	**1979.39**	1826.05	**1988.39**	**2007.95**	55.76
	2	-	-	3905.71	11820.59	3902.44	3913.18	128.80
	3	-	-	5851.96	21620.22	5855.56	5864.11	144.98
	4	-	-	7800.26	14410.70	7804.04	7818.67	135.90
	5	-	-	9753.65	28826.40	9750.66	9790.69	95.56
	6	-	-	11730.98	21619.57	11776.53	11904.97	112.61

The table continues on next page

Table 5.8 Makespan($T_{\max}$) in seconds when solving weighted instances and varying σ (End).

parameters: $\hat{T} = \infty$				Decomposition		VNS		
				time limit = 86400s		stop condition = 300s		
Instance	σ	OPT	sec.	$T_{\max}$	sec.	Best $T_{\max}$	Avg.$T_{\max}$	Avg.(s)
w-P750D15%$\bar{D}$50	1	*3138.73	86400.00	**2984.54**	2474.51	**3008.45**	**3047.97**	105.02
	2	-	-	5849.27	7556.80	5844.06	5858.75	101.48
	3	-	-	8767.62	19072.32	8762.21	8784.29	126.76
	4	-	-	11680.18	21624.97	11679.59	11712.72	144.81
	5	-	-	14602.86	19282.36	14610.90	14665.37	80.64
	6	-	-	-	86400.00	17665.34	17820.95	130.71
w-P1000D15%$\bar{D}$50	1	*4343.26	86400.00	**3917.65**	7657.63	**3962.55**	**4020.12**	86.24
	2	-	-	7740.00	14407.20	7724.25	7752.53	142.33
	3	-	-	-	86400.00	11594.43	11614.93	122.14
	4	-	-	-	86400.00	15467.11	15493.21	112.32
	5	-	-	19306.31	11764.35	19315.50	19383.26	112.98
	6	-	-	23253.94	14438.73	23309.74	23647.27	108.75
w-P200D10%$\bar{D}$90	1	*665.54	86400.00	**646.58**	15579.50	**659.08**	**662.94**	121.04
	2	-	-	1237.80	21616.32	1230.42	1239.56	74.07
	3	-	-	1840.77	28840.56	1838.98	1839.37	104.40
	4	-	-	2460.09	21622.26	2454.44	2459.90	81.69
	5	-	-	3069.42	28825.47	3067.49	3076.62	90.21
	6	-	-	3678.99	36027.84	3678.08	3678.69	92.09
	7	-	-	4305.23	21622.02	4300.25	4330.28	105.68
	8	-	-	4943.87	21621.77	4954.66	4978.82	73.89
w-P400D10%$\bar{D}$90	1	*1302.82	86400.00	**1265.11**	9770.29	**1270.19**	1309.43	105.74
	2	-	-	2468.59	14732.29	2467.42	2475.52	89.36
	3	-	-	3690.58	21646.61	3688.64	3689.12	145.47
	4	-	-	4925.60	21659.91	4924.38	4930.98	105.31
	5	-	-	6151.24	21651.88	6158.54	6169.61	98.90
	6	-	-	7378.95	21647.51	7377.22	7378.39	89.97
	7	-	-	8615.73	19407.21	8611.91	8655.26	101.35
	8	-	-	9893.10	17639.44	9929.49	9981.36	75.65

We observe from the tables that both decomposition and VNS methods performed well for

all instances tested. In fact, both methods find solutions when the exact method fails. In general, it appears that increasing the reliability factor does not affect the performance of the proposed heuristic methods.

Furthermore, the decomposition could reach the optimum or improve (values indicated in bold) the best known makespan (cases with "$*$" in the column "OPT") in several instances, except for the cases "u-P400D7%D90" ($\sigma = 1$), "w-P200D7%D50" ($\sigma = 1$), "w-P400D5%$\bar{D}$70" ($\sigma = 2$), "w-P200D5%$\bar{D}$90" ($\sigma = 2$), "w-P400D5%$\bar{D}$90" ($\sigma = 2$), "w-P400D10%D50" ($\sigma = 1$), "w-P200D7%$\bar{D}$70"($\sigma = 2$), "w-P400D7%$\bar{D}$70"($\sigma = 2$), and "w-P200D10%D70" ($\sigma = 1$) for which more iterations without improving the objective function need to be done by the method in order to be able to obtain best known solutions (as discussed in Section 5.5.2). Otherwise, it kept the percentage deviations w.r.t. the optimum below 1.47% ("u-P400D7%$\bar{D}$90 and $\sigma = 1$") for the unweighted instances and below 3.26% ("w-P200D7%$\bar{D}$50" and $\sigma = 1$) for the weighted cases. Moreover, the decomposition method usually requires considerable smaller execution times than the time spent by CPLEX (first column "sec") to find the optima (or the best known solution for the cases with "$*$") reported in the column "OPT".

Regarding the VNS-based heuristic, it also presented good performance. It is the fastest proposed method, with average makespan deviation w.r.t. the best known solution always inferior to 3.22%("u-P400D10%$\bar{D}$90" and $\sigma = 1$) for unweighted instances, and inferior to 0.57%("w-P200D10%$\bar{D}$70" and $\sigma = 1$) for weighted ones. Besides, VNS solutions are in average better than the best known makespan in 42 instances of the instances tested (values in bold).

Those average makespan values (column "Avg.$T_{\max}$") improvements yield average reduction of 6.14% and 5.17% for the unweighted and weighted cases, respectively. In particular, the VNS heuristic is the unique method reaching feasible solutions for the instances "u-P1000D15%$\bar{D}$50" ($\sigma \in \{3, 4, 6\}$), and "w-P1000D15%$\bar{D}$50" ($\sigma \in \{3, 4\}$). Finally, the VNS heuristic is more reliable in terms of finding near-optimal solutions within a small time horizon when compared to the decomposition method.

The choice of representing a solution through a spatial partition tree as shown in Section 5.4.2 is very important to simplify the exploration of neighboring solutions. However, such representation is not able to represent all the possible spatial-convex covering of the solution space. For instance, Figure 5.10 illustrates an example of covering (dashed lines) which cannot be represented by the adopted spatial-partition tree data structure. Representing such covering would require the use of another data structure. Our VNS explores rather efficiently the solution space, although it is proven not capable of visiting all the feasible solutions for the

CAPsac.

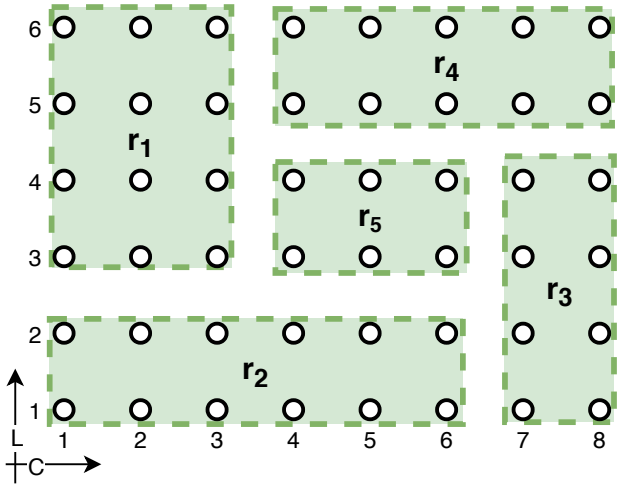


Figure 5.10 Spatial-convex covering which cannot be represented by the adopted spatial-partition tree data structure.

Regarding the sensitivity analysis of the proposed methods to the parameter $\hat{T}$, that is done by progressively decreasing the value of $\hat{T}$ by 1 second until the communication delays turn the problem infeasible. In particular, the experiments are concentrated upon the instances "u-P200D5%$\bar{D}$70" and "w-P200D5%$\bar{D}$70" with $\sigma = 1$. In these instances, the longest communication delays found when executing the decomposition heuristic and the VNS (for 20 runs), when no constraints are imposed to the maximum transmission time (i.e. , $\hat{T} = +\infty$), vary between 33.6s and 48s. For that reason, the first value of $\hat{T}$ used in our experiments was 60 seconds, being decreased by 1 second for each tested $\hat{T}$ down to $\hat{T} = 23s$, when both instances are actually proved infeasible by CPLEX.

Figure 5.11 presents the results of instances "u-P200D5%$\bar{D}$70" and "w-P200D5%$\bar{D}$70" obtained by the decomposition-based and the VNS heuristics for $\hat{T} \in \{24s, 25s, 26s, \ldots, 59s, 60s\}$. The figure shows how decreasing $\hat{T}$ values affect the ability of the proposed methods to achieve optimal or near-optimal solutions. We used distinct shaded regions in the figure to represent the different optimal T^*_{max} values obtained within the interval of tested $\hat{T}$ values. Thus, we can observe for instance "u-P200D5%$\bar{D}$70" two shaded regions before the problem becomes infeasible for $\hat{T} < 24s$, one in which $T^*_{\max} = 2939.20$ for $\hat{T} = [24s, 33s]$, and another in which $T^*_{\max} = 1870.40$ for $\hat{T} = [34s, 60s]$. The vertical axis "Dev.(%)" in the figure reports the percentage deviation w.r.t. the optimum $T^*_{\max}$ in the shaded region associated with each value of $\hat{T}$ tested. Green lines and "•" symbols represent the deviations obtained by the decomposition-based method (named "Decomp.") whereas the average deviation achieved by

20 runs of the VNS method (denoted by "VNS") are represented by the orange lines and empty "○".

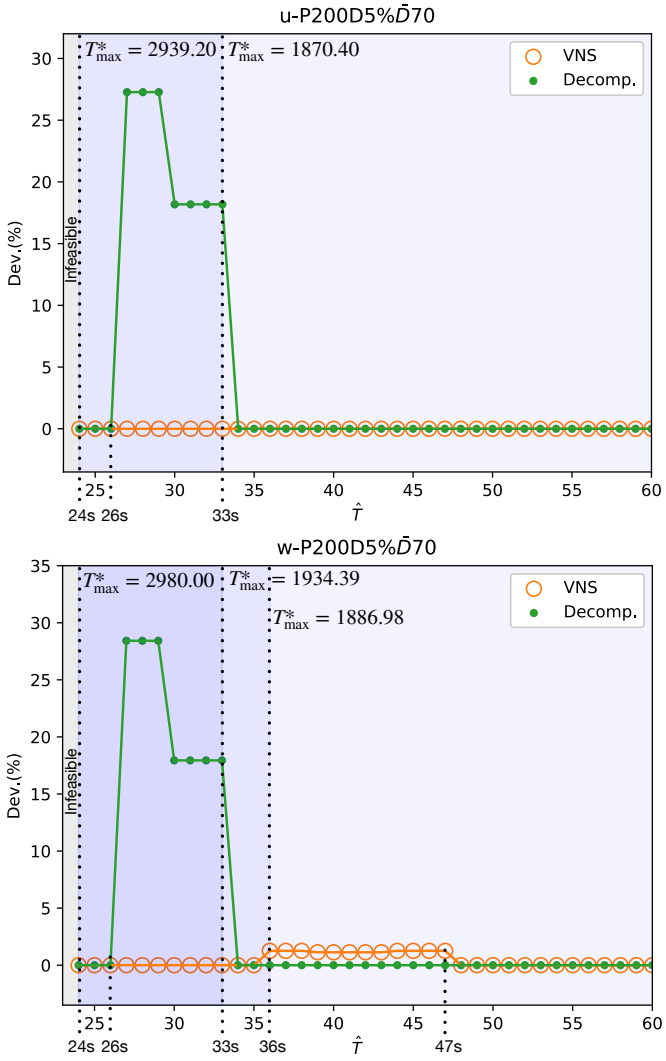


Figure 5.11 Percentage deviation achieved varying $\hat{T}$ for instances "u-P200D5%$\bar{D}$70" (top) and "w-P200D5%$\bar{D}$70" (bottom).

It can be observed in Figure 5.11 from the unweighted instance that the decomposition-based method achieves optimality for most of the tested $\hat{T}$ values except for $27s \leq \hat{T} \leq 33s$ where it reaches percentage deviations up to 27.27%. Those high deviations are mainly due to the $T^*_{\max}$ increase for $\hat{T} \leq 33s$, from 1870.40 to 2939.20. We observed that for $27s \leq \hat{T}$ the initial bounds n_ℓ and n_u of the decomposition method are not good estimates for the cardinalities of the sub-regions that compose the optimal solutions, and thus, the heuristic is likely to finish before reaching a near-optimal solution. For $24s \leq \hat{T} \leq 26s$, the initial limits n_ℓ and n_u lead to infeasible problems until they are large enough to encompass feasible solutions,

which were in this case, close to optima. In contrast, the VNS method achieves optimality for all tested $\hat{T}$ values when solving the unweighted case.

Regarding the weighted case, the decomposition method presents similar behavior as when solving the unweighted case, and its solution deviation goes up to 28.42% for $27s \leq \hat{T} \leq 33s$. The first T^*_{max} increase from 1886.98 to 1934.39 (i.e. +2.5%) was not sufficient to affect the quality of the solution obtained by the decomposition. The VNS only presents small increases in the average deviation for $36s \leq \hat{T} \leq 47s$ but they never surpass 1.25%, otherwise the VNS reaches optimality.

We remark that the VNS method constructs its initial solution in our experiments using $\hat{T}$=$+\infty$. This means that the VNS initial solutions may be infeasible in terms of the transmission delays. Our experiments allowed us to observe that the proposed VNS was able to find feasible solutions even from infeasible starting points.

5.6 Conclusions

We addressed the optimization of the 3D reconstruction step within a swarm-powered 3D mapping mission according to the so-called Covering-Assignment Problem for swarm-powered ad-hoc clouds - CAPsac. It minimizes the completion time of the 3D reconstruction photo-processing phase by exploiting the distributed computational power of the swarm of drones. This was achieved by integrating the photo covering (workload) optimization along with the assignment of the drones to the photo processing subtasks (i.e., sub-regions).

Since time is a very precious resource in emergency field operations supported by UAVs, the limited computation time — in the order of a couple of minutes for the biggest and more complex instances, we proposed two heuristic algorithms to solve this NP-hard problem in a limited amount of time (i) a mathematical programming-based heuristic as well as a (ii) Variable Neighborhood Search method.

Despite the fact that we had focused on a swarm-powered distributed 3D reconstruction for humanitarian emergency response application, the proposed methods can be deployed to any application in the CAPsac's scope. Finally, the methods can be adapted to consider auxiliary computing resources in addition to those offered by the UAV swarm.

Computational experiments were conducted with the unweighted and weighted realistic instances available online at `https://github.com/ds4dm/CAPsac` to assess the performance of the proposed heuristic methods. The experiments exposed that the VNS heuristic either quickly achieves near-optimal solutions or rapidly improves the best known makespan for a vast number of instances. The decomposition-based heuristic is also effective in most of

the tested cases and is an efficient method when compared to solving the $CAPsac$ formulations by commercial solvers. Although the performance of decomposition-based heuristic deteriorates for some values of $\hat{T}$ close to infeasibility, the sensitivity analysis done for the decomposition and VNS methods demonstrated that those methods still perform well when varying σ and $\hat{T}$.

Our current work with Humanitas Solutions (`https://www.humanitas.io/`) is focused on embedding the heuristics proposed in this paper into real deployed UAV swarms.

CHAPTER 6 OptiMaP: SWARM-POWERED OPTIMIZED 3D MAPPING PIPELINE FOR EMERGENCY RESPONSE OPERATIONS

The advancement of mobile devices capabilities has fostered the creation of the Internet of Things (IoT) [1] and smart cities [2]. Given their massive amount of data and processing tasks [3], services providing dynamically scalable and virtualized storage and computing resources over the internet as cloud computing [4] became crucial. However, cloud computing solutions are susceptible to high latency and mobility support issues [5]. Edge Computing (EC) infrastructures are similar to the ones found in cloud computing, but their resources are located at the *edge* of the network, e.g., in computational and network resources on the routing path between the source of the data/processing task and the data centers [6]. EC solutions arise as a popular alternative to cloud computing by addressing high latency issues.

The capabilities of EC are enhanced by the adoption of Unmanned Aerial Vehicles (UAVs), which are also referred as drones, given their aerial capabilities and low cost [7–11]. Such characteristics are further exploited when UAVs cooperate as a fully autonomous *swarm*. In swarm robotics, simple agents are coordinated in a decentralized way to realize complex collective tasks while maximizing the swarm performance and resilience [12]. Naturally, swarms of drones are applied in a vast list of tasks such as target search and tracking, surveillance, and mapping [13]. Concerning EC applications, UAV swarming may implement dedicated wireless networks [14] as well as establish ad-hoc clouds that provides storage and processing resources [15–18].

Likewise EC technology, UAV swarming is often employed in synergy with digital photogrammetry [45, 46] for applications such as 3D mapping, where a set of aerial images are collected for a target region so that its 3D model can be reconstructed. In such context, the set of images required to create the 3D map are usually collected all in parallel by the swarm of drones. However, the 3D reconstruction process is typically carried in a centralized manner by a dedicated computer or data center [48–53], being vulnerable to connectivity and latency issues.

Given the power within the UAV swarm, its distributed resources can be exploited through the creation of an ad-hoc cloud infrastructure [15–18] in which the 3D reconstruction process can be performed. In [124], the authors introduced the Covering-Assignment Problem for swarm-powered ad-hoc clouds (CAPsac) which minimizes the completion time of processing tasks offloaded to a swarm-powered ad-hoc cloud by jointly optimizing the workload generation and workload assignment. In the context of a swarm-powered 3D mapping missions,

the CAPsac aims to optimally create and allocate the multi-node computing workload that constitutes the 3D reconstruction process. Given that each computing node reconstructs a specific sub-region on a specific drone, an optimal solution for the CAPsac defines how to minimize the completion time of the overall 3D reconstruction process by optimally i) dividing the set of images into various sub-regions and then ii) allocating each sub-region to a certain UAV.

Providing the 3D map of the operation area enhances real-life emergence response applications since it improves the decision-making and situation awareness during the mission. In fact, those 3D twin maps enable first responders to expose threats as damage in roads and buildings or dangerous zones [54–57]. In such operations, each minute is critical and it is crucial to build 3D maps as quickly as possible regardless of internet connection. Yet, by efficiently constructing 3D maps of the target region to emergency responders, we also release UAVs to perform other crucial tasks for the first response operation.

In this chapter, we propose and deploy the *swarm-powered Optimized 3D Mapping Pipeline* (OptiMaP) for emergency response operations. The OptiMaP allows the 3D reconstruction process to be distributed across the swarm members instead of relying on an unique central workstation as done in [47–53].

The next section describes the OptiMaP for emergency response missions. In the sequel, each step in the OptiMaP is described: Section 6.2 presents the waypoint generation process; Section 6.3 explains the multi-UAV photo collection step; Section 6.4 describes the workload generation and allocation mechanism as well as the heuristics adopted for solving it; Section 6.5 presents the distributed 3D reconstruction phase. Finally, Section 6.6 shows and analyses the deployment of the 3D mapping missions in a realistic simulator, while Section 6.7 outlines our conclusions.

6.1 Swarm-powered optimized 3D mapping pipeline

OptiMaP is built with the Robotic Operating System (ROS) framework [150], and employs a ground station ROS node in addition to the ROS nodes for the drones. Besides dealing with centralized tasks within the pipeline, the ground station is the interface with the user — it allows the user to setup and supervise the mission, and provides the resulting 3D maps. In order to define the underlying ad-hoc wireless network connecting the drones, OptiMaP uses the Heterogeneous Embedded Ad-hoc Virtual Emergency Network (HEAVEN) middleware [151].

The swarm-powered OptiMaP for emergency response operations can be decomposed into

four main steps (illustrated in Figure 6.1):

1. **Waypoint generation**: Global Positioning System (GPS) coordinates are defined by a ground station to guide the UAVs during the photo collection step. Each waypoint represents a photo shooting location in the target region;
2. **Multi-UAV photo collection**: the UAV swarm cooperates to capture all photos following the generated waypoints.
3. **Optimized workload generation and allocation according to CAPsac**: the set of photos is split into sub-regions, and each sub-region (i.e., subset of photos) is assigned to a UAV according to the CAPsac [124] optimization.
4. **Distributed 3D reconstruction**: the workload created in the previous step is offloaded to the assigned UAVs. Once a UAV lands at its landing position, it performs the 3D reconstruction of its assigned sub-regions.

Besides employing the UAV swarm for cooperative photo collection, our pipeline exploits the swarming ad-hoc cloud infrastructure during the whole 3D reconstruction procedure. Thus, OptiMaP is resilient to latency and connectivity issues and speed-ups the creation of the 3D model of the target region. At the end of the process, the digital replica of the target region is available for the first responders in the ground station. Next, we detail each step of the OptiMaP pipeline.

6.2 Waypoints generation

The image-based 3D reconstruction process requires photos taken from various perspectives of the region of interest in order to create the correspondent 3D model. This step generates the *waypoints* — GPS coordinates which define where those photos must be taken — required to portray the whole target region. Since the resulting 3D model heavily depends on its input photos, generating waypoints that leads to a good 3D model is crucial for the success of the 3D mapping mission. When generating waypoints, parameters as perspective, lens quality, overlap, coverage, and object geometry affect the number of photos required to produce 3D maps with good resolution [87, 96]. The same amount of waypoints may be appropriate to regions of interest with different dimensions when adjusting those parameters [57].

Among those parameters, the overlaps are important for the 3D reconstruction since they define how much the photo footprints (i.e., area captured by a photo) may overlap between each other. Large overlaps improve the resulting 3D model but may require more photos to

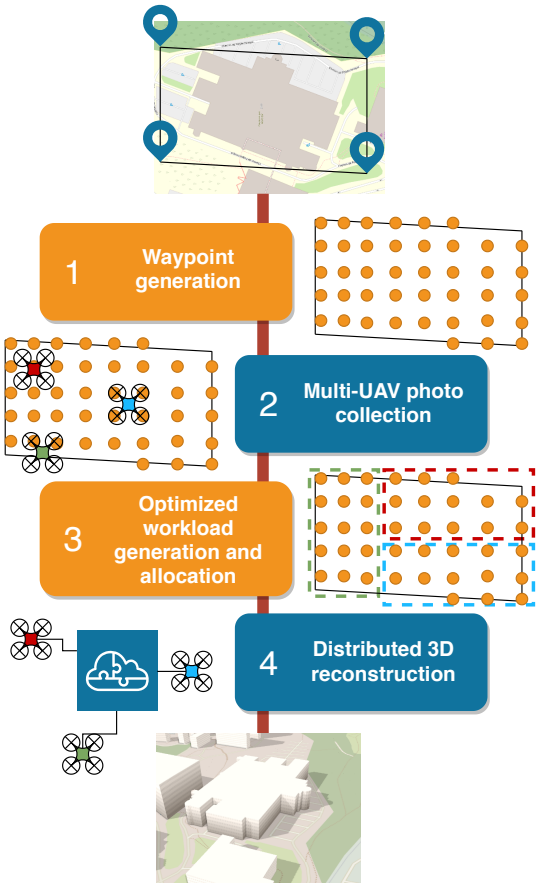


Figure 6.1 OptiMaP pipeline.

capture the whole region of interest. Usually, good values of overlaps range between 60% and 80% [96]. A simple way to fulfill both overlap and region coverage constraints is to generate waypoints on a grid pattern [152].

Given a convex polygon delimiting the region of interest and an overlap θ (in percentage), one can construct a grid of waypoints such that the resulting photo footprints covers the region of interest while respecting θ [152]:

1. construct the rectangular bounding box encompassing the region of interest;

2. place waypoints in a grid pattern along the rectangular bounding box starting from its superior left vertex. The distance Δ_x in meters between two consecutive waypoints in the same row is defined by

$$\Delta_x = w - \frac{w\theta}{100} \tag{6.1}$$

where w is the width in meters of the photo footprint [96]. The spacing Δ_y (in meters)

between two rows is obtained according to

$$\Delta_y = \ell \frac{100 - \theta}{100} \tag{6.2}$$

where ℓ corresponds to the length in meters of the photo footprint [96];

3. remove created waypoints lying outside the convex polygon which delimits the region of interest.

In OptiMaP, we perform the described waypoints generation step at the ground station.

6.3 Multi-UAV photo collection

Once the waypoints were created by the ground station, they can be assigned to drones for the photo collection step. A long list of task assignment methods were proposed in the literature [44]. A swarm of drones benefits from decentralized task allocation methods given their dynamic nature [44]. Since those methods require specific techniques to consider the unpredictable behavior of the UAV swarm (e.g., [153–156]), which is beyond the scope of this chapter, we adopt a simple deterministic method to assign the defined waypoints to the swarm drones.

Given a set of drones D, let us denote by $W = \{w_1, w_2, \cdots, w_n\}$ the set of waypoints created during the waypoints generation step such that waypoints $w \in W$ are ordered according to their creation time. Also, let n_d be the number of waypoints assigned to a drone $d \in D$. The waypoint assignment method assigns $\frac{|W|}{|D|}$ waypoints per drone, i.e., $n_d = \frac{|W|}{|D|}$ for all drones $d \in D$ in such a way that each waypoint is assigned to exactly one drone. Whenever $|W|$ is not divisible by $|D|$, the drones are assigned to either $\left\lceil \frac{|W|}{|D|} \right\rceil$ or $\left\lfloor \frac{|W|}{|D|} \right\rfloor$ waypoints. The first $|W| - |D| \left\lfloor \frac{|W|}{|D|} \right\rfloor$ drones have $n_d = \left\lceil \frac{|W|}{|D|} \right\rceil$ waypoints whereas the remaining drones have $n_d = \left\lfloor \frac{|W|}{|D|} \right\rfloor$. Finally, starting from the first drone in D and the first waypoint in W, the method sequentially assigns n_d waypoints to each drone $d \in D$. This process is repeated until all waypoints are assigned.

Once all waypoints are assigned to a drone by the ground station, the drones take off from their landing position and start the photo collection. Naturally, a drone returns and lands at his landing position when it has collected all of its assigned photos (waypoints).

The deterministic nature of that method allows us to use the exact same scenario when comparing distinct heuristic strategies used in the optimized workload creation and generation step (Section 6.4).

6.4 Optimized workload generation and allocation according to CAPsac

We address the exploitation of the swarm-powered ad-hoc cloud by casting the 3D reconstruction processing task as an instance of the CAPsac. This approach promises to yield large improvements in the 3D reconstruction completion time when compared with centralized 3D reconstruction procedures.

Given the photos P that were captured along waypoints W to portray the target region, the CAPsac is a NP-hard problem that minimizes the completion time of the 3D reconstruction process (presented in the following section) by jointly computing (i) the optimal covering of photos, i.e., the splitting of the target region into multiple convex sub-regions, and (ii) the optimal sub-region-to-drone assignments [124]. A CAPsac solution, thus, renders how to parallelize and distribute a 3D reconstruction task across (*3D-capable*) UAVs by decomposing it into small 3D reconstruction sub-tasks.

CAPsac deals with two main constraints when optimizing workload generation and allocation [124]:

1. the sub-regions must form a *spatial-convex covering*, i.e., the union of all sub-regions (subset of photos) is equal to the target region (P) and each sub-regions is a *spatial-convex set* — all photos lying inside the set's convex hull are allocated to that same sub-region;

2. the transmission of data among the drones must follow the *Max-Min Fairness* (MMF) rate allocation paradigm [79], and the communication delays cannot exceed $\hat{T}$ seconds;

The communication delays must be addressed since the photos are scattered across the drones' hard drives and a 3D reconstruction sub-task cannot start until the responsible drone has all photos that are part of the sub-task (i.e., sub-region reconstruction). As a consequence, drones exchange photos between each other to complete their assigned sub-tasks.

CAPsac accounts for the transmission delays by establishing TCP sessions on top of a single tree topology. Naturally, the MMF paradigm is adopted for the transmission rates allocation since it approximates the TCP protocol behavior [82]. The TCP rate allocation mechanism can diverge from the ideal MMF paradigm when using a multi-hop wireless network [135]. Nevertheless, it is demonstrated by [83] that an approximated form of fairness achieves better routing solutions for elastic traffic demands than adjusting them as inelastic. Other issues as round-trip-time variance and multi-connection schemes [135] can be addressed by adjusting a specific group of constraints [83].

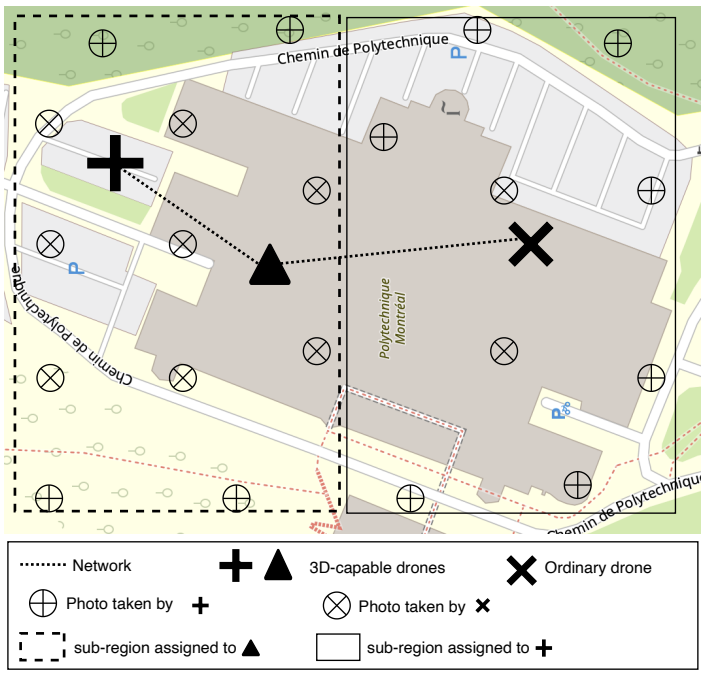


Figure 6.2 Illustrative solution obtained by CAPsac.

Figure 6.2 illustrates a solution for an instance of the CAPsac with two 3D-capable drones ("+" and "▲" symbols) and one ordinary drone ("×" symbol) — the drones are connected by a tree topology network (represented by dotted lines). The waypoints (and the correspondent photos) allocated to the drone "+" are represented by the "⊕" symbols. Similarly, the "⊗" symbols correspond to waypoints allocated to the drone "×". The target region is split into two sub-regions (delimited by rectangles). The sub-region in the left (dashed lines) is assigned to the drone "▲" whereas the sub-region in the right is assigned to the drone "+".

Both exact and heuristic approaches were proposed to solve CAPsac [124, 125]. In [125], heuristic methods for solving the CAPsac were proposed: one decomposition-based method, and a method based on Variable Neighborhood Search (VNS) [157]. The authors demonstrate that their VNS heuristic outperforms the decomposition approach in most of the real scenarios tested.

However, it is not known how much improvement is achieved with workloads generated and allocated by CAPsac heuristics when compared with non-CAPsac approaches. To perform such analysis, we propose a simple constructive greedy heuristic that generates and allocates workloads, but they do not fulfill transmission delay constraints. Thus, the greedy method serves as a baseline for the workloads created and allocated by the CAPsac.

In the next sections, we describe a greedy baseline heuristic and the VNS heuristic of [125]

for the CAPsac.

6.4.1 Greedy heuristic

We propose a simple greedy constructive heuristic for obtaining a workload and its allocation in the ad-hoc cloud computing resources. The constructed solution produces spatial-convex coverings and satisfies the network resilience. However, it does not meet the maximum allowed communication delays, and hence, it does not constitute a CAPsac feasible solution. Nonetheless, that solution is quickly obtained by the greedy heuristic and serves to define a baseline to assess whether ad-hoc clouds actually benefits from CAPsac solutions obtained by other more sophisticated heuristics (as the one in Section 6.4.2).

Our proposed greedy method splits the target region into rectangular sub-regions of equal dimension. Then, it iteratively assigns the unassigned sub-regions among the 3D-capable drones accounting for their current individual completion time and active traffic demands. So, the method (i) greedily selects the 3D-capable drone $\bar{d}$ whose current completion time is minimal and, then (ii) allocates to $\bar{d}$ the unassigned sub-region that yields the smallest increase of incoming data to be transmitted to $\bar{d}$.

6.4.2 Variable neighborhood search-based heuristic

In this section, we summarize the VNS-based method proposed in [125] for solving instances of CAPsac. Variable neighborhood search is a stochastic search framework based on heuristics that has been successfully applied to many NP-hard problems [147] like the CAPsac.

For a given optimization problem $\mathcal{P}$, let us denote $\mathcal{S}$ its solution space, $\mathcal{X}$ its (finite but very large) feasible set of solutions, x a solution of $\mathcal{P}$, and $f(x)$ the objective function $f : x \to \mathbb{R}$ that maps a solution x to its *cost*. We formally define a (minimization) optimization problem $\mathcal{P}$ as

$$\mathcal{P} = \min \{f(x) | x \in \mathcal{X}, \mathcal{X} \subseteq \mathcal{S}\}. \tag{6.3}$$

When the solution space $\mathcal{S} = \mathbb{R}^n$, $\mathcal{P}$ is as a *continuous* optimization problem, whereas $\mathcal{P}$ is classified as *discrete* optimization problem if $\mathcal{S} = \mathbb{Z}^n$. A solution x^* is the *global optimum* of $\mathcal{P}$ if and only if

$$f(x^*) \leq f(x) \quad \forall x \in \mathcal{X}. \tag{6.4}$$

Given a solution x, let us denote by $\mathcal{N}(x) \subset \mathcal{X}$ a neighborhood of x composed by all its neighboring solutions, i.e., all solutions obtained from x by means of a single local transformation, e.g., complementing terms of $x = \{0, 1\}^n$ in a combinatorial optimization problem

or increasing/decreasing terms in $x = \mathbb{R}^n$ up to a certain value in a continuous optimization setting. A *local optimum* $\hat{x}$ associated with a neighborhood is defined as

$$f(\hat{x}) \leq f(x) \quad \forall x \in \mathcal{N}(\hat{x}). \tag{6.5}$$

VNS achieves optimal or near-optimal solutions by keeping a single solution x during its execution while searching for a better solution $x' \in \mathcal{N}_k(x)$ in increasingly larger neighborhoods of x so that $\mathcal{N}_1(x) \leq \mathcal{N}_2(x) \leq \ldots \leq \mathcal{N}_{k_{max}}(x)$. By starting from an initial solution x and $k = 1$, VNS sequentially applies a *diversification step* followed by an *intensification step* until a stopping condition (e.g., execution time) is met. The diversification step draws from $\mathcal{N}_k(x)$ a random neighboring solution x'. Then, the intensification step performs a local descent procedure from x' which results in the local optimum x''. If the x'' is worst or equivalent to x, k is incremented and x is updated. Otherwise, k is reset to 1. The VNS search then resumes from its diversification step, and those steps are repeated until the stopping condition is met. We remark that k is also reset to one every time $k > k_{\max}$ so that the VNS does not degenerate to a random search.

In [125], the authors developed a VNS heuristic for the CAPsac. Their method represents a solution of the problem by means of a *spatial-partition tree* data structure, which is illustrated in Figure 6.3. In the represented tree, the set of distinct photo latitudes L and the set of distinct photo longitudes C are split three times (numbered lines) to form four rectangular sub-regions (dashed lines).

The VNS method of [125] explores four distinct neighborhoods for a solution, namely:

- *sub-tree reconstruction*, which randomly reconstructs a sub-tree rooted at a given depth of the spatial-partition tree;
- *sub-region transfer*, which switches a sub-region assignment from one drone to another;
- *sub-region swap*, which swaps a pair of sub-region assignments between two distinct drones;
- *splitting hyperplane reallocation*, which reallocates splitting hyperplanes in nodes of the spatial partition tree.

The sub-tree reconstruction neighborhood is still applied in the diversification step. In this case, the parameter k of the VNS establishes the depth k from which a sub-tree will be randomly selected for reconstruction. Consequently, $k_{\max}$ is equal to the depth of the whole spatial partition tree. The reader is referred to [125] for more details about the VNS method.

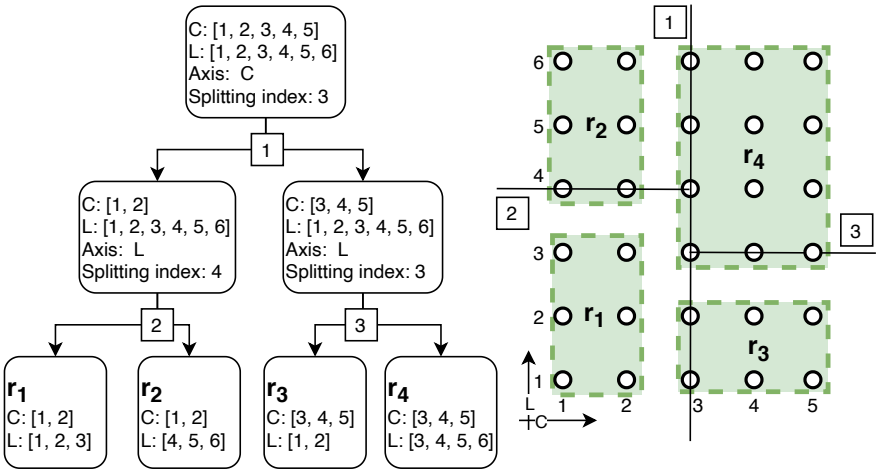


Figure 6.3 Spatial partition tree (left) and the respective spatial-convex covering of waypoints (right).

6.5 Distributed 3D reconstruction

The 3D reconstruction constructs a 3D model of a given scene portrayed by a set of input images taken from different perspectives. It sequentially applies the so-called Structure from Motion (SfM) [84] and Multi-View Stereo (MVS) [85] techniques [86]. The whole process can be decomposed into three steps [87]:

1. *Feature detection*: finds key features as corners, edges, and blobs in all input photos;
2. *Feature matching*: matches the key features found across multiple photos — the matching of key features from distinct photos means that they are the same key feature being portrayed in multiple photos;
3. *Reconstruction*: computes the 3D position of each matched key feature based on their 2D position in the photos and corresponding camera pose.

These steps produce the so-called *dense point cloud* which is the set of 3D positions of all matched key features. At this point, two steps (final phase) still need to be performed to transform the dense point cloud into a complete 3D model:

1. *Meshing*: given a dense point cloud as input, the 3D model surface is estimated (i.e., triangle meshes). Different methods can be applied to this step, e.g., [93, 94].
2. *Texturing*: based on the appearance of pixels in the input photos, characteristics like color, illumination, and texture are added to the 3D model, e.g., [95].

After the meshing and texturing steps, the complete 3D model of the region portrayed by the input photos is ready.

In the context of OptiMaP, each sub-region (subset of photos) created in the previous step is used as the input photos of a 3D reconstruction sub-model (sub-task). Then, the 3D reconstruction process for each sub-region is done according to the workload allocation from the CAPsac solution. As a result, at the end of the reconstruction process, each drone has one dense point cloud for each of its assigned sub-regions. At this point, all dense point clouds are sent to the ground station which is responsible for merging them and performing the meshing and texturing steps. Finally, a digital replica of the whole target area is made available.

All processing steps for the 3D reconstruction are performed through the OpenDroneMap toolkit [158]. Furthermore, each 3D reconstruction sub-task created by CAPsac is implemented as a Docker container [159].

6.6 Deployment of the 3D mapping mission via simulation

Our experimental analysis aimed to evaluate (i) the efficiency of the OptiMaP pipeline for emergency response applications, and (ii) the quality of the workloads created and allocated by the VNS-based method when compared with the workload obtained by the baseline heuristic.

The deployment and anticipated analysis of emergency response situations by simulation are crucial for planning effective real-life missions. Such practice allows examining distinct response protocols without facing an actual crisis. Furthermore, there is a scarcity of realistic data, and the use of multiple real drones may result in high testing costs and risks.

We perform our experiments through a simulator able to provide realistic physics and visuals: the *HyperXSpace* (HXS) simulator [160]. Besides leveraging the realistic drones as proposed in AirSim [161] with detailed realistic 3D scenarios and ad-hoc communication emulation, HXS carries the simulations in a cloud-based infrastructure. HXS emulates the ad-hoc wireless network by changing the package loss probability and reducing the ideal bandwidth between two nodes according to the inter-drone distance. Furthermore, HXS represents each agent (e.g., UAV) as a set of Docker containers in such a way that Software-in-the-loop (SITL) such as flight controllers are supported. Therefore, applications simulated in HXS can be directly deployed on real drones. Such testing environment allows us to deploy and evaluate the OptiMaP in realistic emergency response scenarios. It also provides a controlled experimental setting for the compared algorithms since they can be tested in identical conditions, which

would be very hard to obtain with the use of real drones.

In order to evaluate the OptiMaP in different scenarios, we vary the number of waypoints/photos of the tested instances as described in Table 6.1. Concerning the scenarios with 150, 226, and 363 photos, we considered areas with higher density to represent points of interest that require more photos to produce detailed 3D replicas of those respective areas. All scenarios were constructed to fulfill 90% of overlapping between the photo footprints given a field of view of 90° and a drone altitude of 100m. We tested scenarios with 5 drones and 3 sub-regions (i.e, 3D-capable drones). We kept the number of 3D-capable drones smaller than the total number of drones to force communication among them. Besides, data exchanges allow us to evaluate the impact of the workload generation and allocation in generating communication delays. In all scenarios, the network emulator reduces the bandwidth between two nodes (drones) according to their distance. Given a certain communication link and the respective inter-drone distance, the ideal bandwidth is reduced by 20% each 50 meters starting from a distance of 50 meters (5mbps and 0% package loss) up to 200 meters from which the link is no longer available (i.e., bandwidth equal to zero and 100% of packages loss). Thus, taking into consideration that bandwidth configuration, we placed the drone (landing) positions to generate the (line) topology presented in Figure 6.4. That topology was designed to force multiple traffic demands to share the same links. The values for the maximum allowed communication delay ($\hat{T}$) are presented in Table 6.1 as well. They were set according to each analyzed scenario to represent realistic values.

6.6.1 Simulation results

First, we evaluate the advantage of adopting the OptiMaP when compared to reconstruct the whole target region in a centralized manner. We do not report the time spent by the waypoint generation and the photo collection steps since they are identical for a given number of photos in the HXS. We define completion time as the time required to exchange all photos and perform the 3D reconstruction, i.e., the completion time accounts for communication delays and 3D reconstruction duration only. Figure 6.5 presents the total completion time of the 3D mapping mission when relying on the centralized approach (red ■ symbols) and on the decentralized ones: the greedy heuristic (orange ♦ symbols) and the VNS-based method (green • symbols) – halted after $30s$ of execution time. Similarly, Figure 6.6 shows how the communication delays are affected by adopting the CAPsac workload generation and allocation in comparison with the centralized and non-CAPsac approach (greedy method).

We note from Figure 6.6 that the transmission delays for VNS may also exceed $\hat{T}$, as it is the case for the mapping scenarios 1 (71.16% — 17.79s longer), 2 (17.97% — 7.19s longer), 3

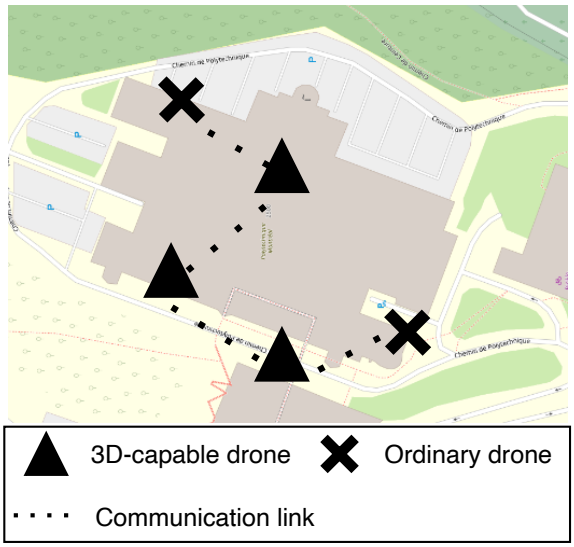


Figure 6.4 Network topology employed in deployed scenarios.

Table 6.1 Deployed 3D mapping scenarios.

Scenario	Photos	Area (km^2)	$\hat{T}$ (sec.)
1	135	0.124	25
2	150	0.124	40
3	211	0.186	35
4	226	0.186	100
5	348	0.293	80
6	363	0.293	120

(73.97% — 25.89s longer), and 5 (24.11% — 19.29s longer). This is due to how the MMF rate allocation paradigm addressed by the CAPsac approximates the TCP protocol behavior. It is known that the ideal MMF paradigm can diverge from actual TCP rate allocation in multi-hop wireless network [135] and that it cannot address issues as the hidden node phenomenon. However, such approximated allocation paradigm leads to better routing solutions for elastic traffic demands than addressing them as inelastic demands [83]. Furthermore, the communication delays could be improved by considering, for each traffic demand, a round-trip-time (RTT) based scalability parameters that would instruct the MMF constraints to assign more bandwidth to the traffic demands served by shorter routing paths [79,124]. For this purpose, before launching the optimization algorithm, we could modify the application to derive the inter-drone RTT with a series of PING commands; the obtained latency values could be then normalized to compute the RTT scalability parameters of the MMF constraints.

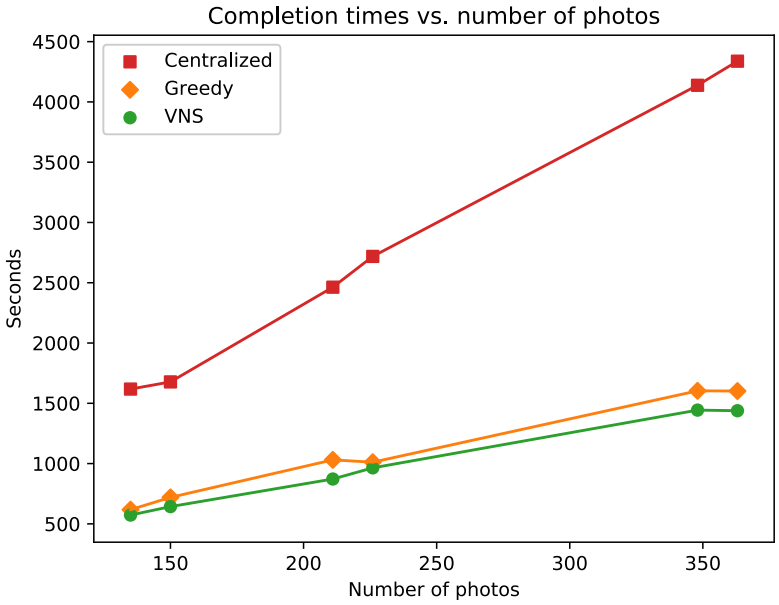


Figure 6.5 Completion times when employing the centralized approach, the greedy heuristic, and the VNS method.

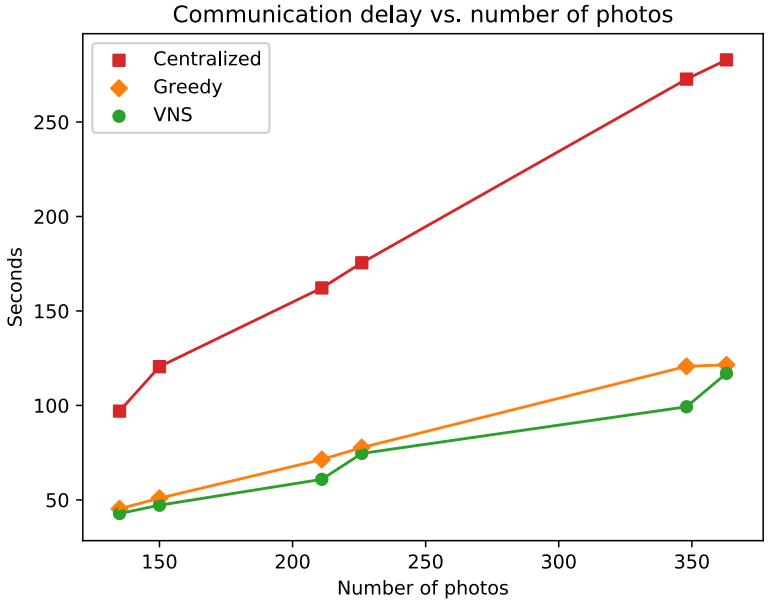


Figure 6.6 Communication delays obtained by adopting the centralized approach, the greedy heuristic, and the VNS method.

We noticed a significant gain of performance when optimizing the workload creation and allocation with the greedy and VNS-based methods. Both approaches presented a dramatic reduction in the completion times and the communication delays when compared to reconstructing the target region with a equivalent centralized workstation. Concerning only the greedy and the VNS heuristics, the VNS presents best performance for both completion and transmission times, and its violations of $\hat{T}$, if any, are always reasonable.

Figure 6.7 reports the completion time (i.e., transmission delay + 3D reconstruction duration) of each drone when employing the greedy (orange ♦ symbols) and VNS (green • symbols) heuristics. As observed in Figure 6.7, the VNS yields most balanced workload generation and allocation (i.e., similar individual drone completion times) across the drones. Hence, we can conclude that the VNS exploits better the computing resources within the UAV-powered ad-hoc cloud. In fact, the completion times of the solutions obtained by the greedy heuristic can be up to 18.22% longer than those of the solutions obtained by the VNS.

Finally, we assess the impact of OptiMaP in the use of the network resources. Figure 6.8 shows the total amount of data exchanged (lines) and the largest transferred amount of data among all performed transmission demands (dashed lines) when adopting the centralized approach (red ■ symbols), the greedy heuristic (orange ♦ symbols), and the VNS-based method (green • symbols). Similarly, Figure 6.9 reports the average throughput achieved by the centralized approach, the greedy heuristic, and the VNS-based method.

We observe in Figure 6.8 that the OptiMaP with the VNS heuristic usually exchanges less amount of data when compared with the centralized and the greedy approaches. Furthermore, as seen in Figure 6.9, the OptiMaP also leads to better flow rate allocations, making a more effective use of the networking resources available in the ad-hoc cloud.

6.7 Conclusions

We proposed and deployed the swarm-powered Optimized 3D Mapping Pipeline (OptiMaP) for emergency response applications. It optimizes the workload generation and allocation by employing a VNS-based heuristic that solves the Covering-Assignment Problem for swarm-powered ad-hoc clouds (CAPsac).

The OptiMaP allows the 3D reconstruction process to be distributed across the swarm-powered ad-hoc cloud instead of relying uniquely on a central workstation. Such approach reduces the creation time of the 3D maps and takes into account transmission delay constraints required by the application. As a result, 3D maps of the target region are rapidly provided to the emergency responders, thus allowing the UAVs to perform other crucial tasks

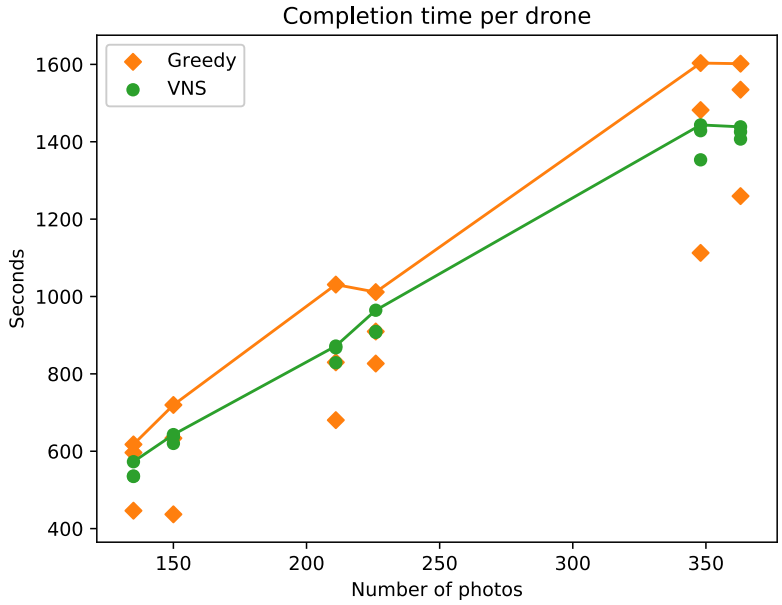


Figure 6.7 Completion time per drone when using the greedy and VNS heuristics used within OptiMaP.

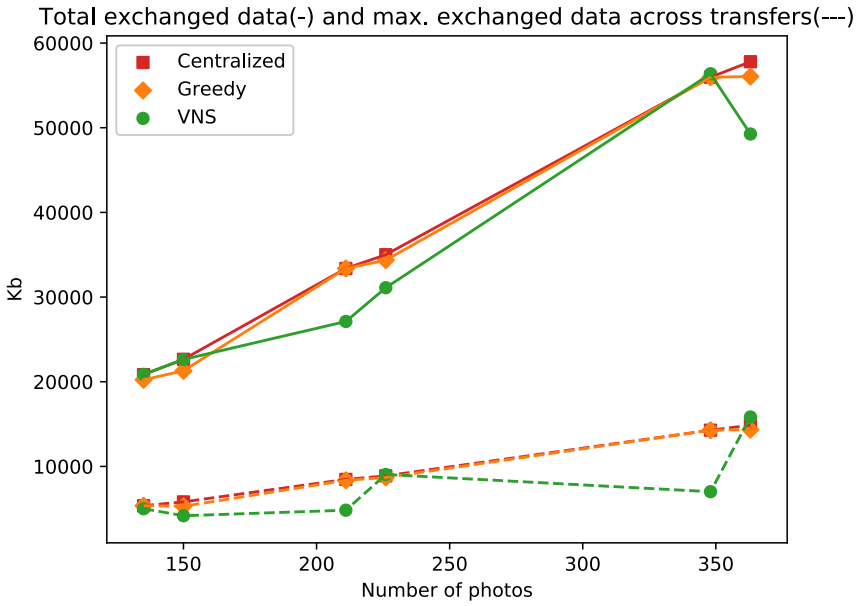


Figure 6.8 Total data exchanged (lines) and the data sent in the largest transmission demand (dashed lines) by the centralized approach, the greedy method and VNS heuristic.

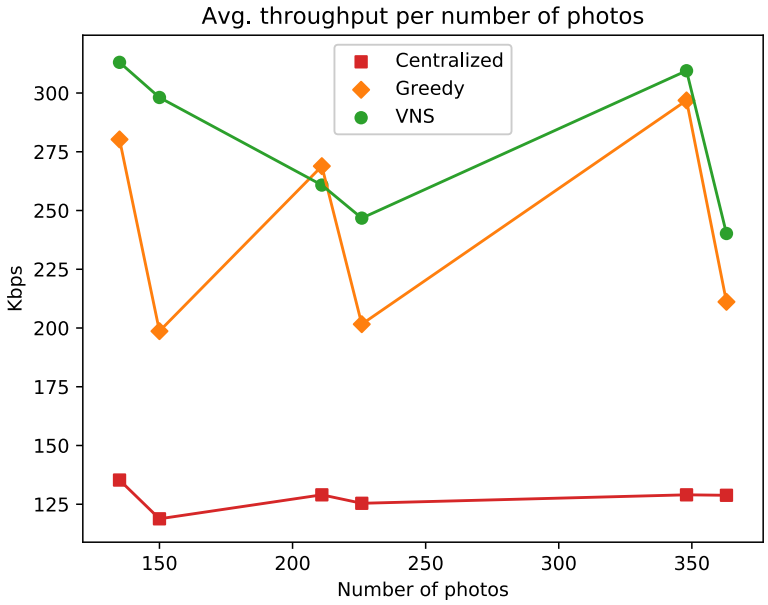


Figure 6.9 Average throughput achieved by the centralized approach, the greedy method and VNS heuristic.

for the first response operation.

The simulation results proved that both communication delays and completion times of those missions are drastically reduced by OptiMaP. When comparing the (CAPsac) VNS-based method against the (non-CAPsac) greedy heuristic, the VNS heuristic presented the best completion times and transmission delays as well as a better utilization of the computing and network resources of the swarm-powered ad-hoc cloud.

CHAPTER 7 GENERAL DISCUSSION

The contributions presented in this thesis focus on optimizing the workload carried by swarm-power ad-hoc clouds. Furthermore, we adopted the swarm-powered 3D mapping mission for emergency response scenarios as use-case. In order to optimize the use of swarm-powered ad-hoc clouds, we proposed a new NP-hard optimization problem, namely CAPsac, that jointly optimizes the workload creation and assignment. We presented different MILP formulations to obtain optimal solutions for the proposed problem. Each minute counts during timely manner applications, and the time required to obtain optimal solutions may be impractical for these situations. Thus, we designed heuristics able to find good CAPsac solutions quickly for emergency response missions. Finally, in order to assess how much improvement can be reached in real-life conditions, we developed an optimized distributed 3D reconstruction pipeline for emergence response missions.

7.1 Summary of Works

We proposed the CAPsac to exploit the network and computing resources embedded within an ad-hoc cloud established on top of a swarm of drones and connected by a dedicated wireless network. A CAPsac solution maps the tasks carried by the ad-hoc cloud to swarm members such that their overall completion time is minimized. Furthermore, the CAPsac jointly groups the processing tasks into batches (i.e., creates a workload) and assigns these batches to swarm members (i.e., assigns the workload). For instance, during a swarm-powered 3D mapping mission, the CAPsac simultaneously optimizes the split of the region to be portrayed in the 3D map (i.e., a collection of photos) into smaller sub-regions (which are subsets of photos), and the assignment of these sub-regions to the drones.

In Chapter 4, we presented the NP-hardness proof of the CAPsac and two different MILP formulations to solve the CAPsac. The photo-based CAPsac (pCAPsac) formulation assigns each photo forming the target region to one sub-region later processed by a swarm member. The region-based CAPsac (rCAPsac) formulation selects the optimal set of sub-regions among all possible feasible rectangular sub-regions of a target region. In order to assess those formulations, we carried a series of computational experiments with a set of unweighted and weighted realistic benchmark instances. These experiments revealed that the "*pCAPsac*" formulation was more efficient by using ordering inequalities that remove from the feasible continuous search space sub-regions whose boundaries are not regular. In fact, the "*pCAPsac*" formulation, within the configured time limit, could solve more than

50% of the considered realistic instances involving up to two hundred images and six drones. However, the different branching priority strategies and row generation methods have not proven to yield a performance gain while solving that formulation. The sensitivity analysis of the formulation "*pCAPsac*" showed that it becomes more difficult to solve as the reliability factor σ increases. Tests with varying values for the maximum allowed transmission time $\hat{T}$ also presented a slight gain of performance up to when $\hat{T}$ approaches a limit for which the problem becomes unfeasible. Finally, column generation was employed in the "*rCAPsac*" formulation, but the presence of highly degenerate optima led to long execution times.

According to the current MILP solvers performance and amount of offloaded computational tasks, it may be impractical for timely manner applications to find the optimal workload creation and workload allocation due to the time required to reach optimality. Therefore, in order to quickly obtain good solutions for critical applications as emergency response missions, we designed heuristics able to find near-optimal CAPsac solutions in a short time horizon.

In Chapter 5, we described a mathematical programming heuristic based on decomposition and a VNS heuristic to minimize the completion time of the data processing step in those timely manner applications according to the CAPsac. The decomposition-based method is inspired by the observation that good solutions have their workload close to the perfect workload division w.r.t. the number of photos per sub-region. Therefore, we limit the solution space of the "*rCAPsac*" formulation to sub-regions for which the number of photos lie inside a predefined interval $[n_\ell, n_u]$. In brief, the decomposition-based method iteratively increases that interval until the value of the objective function doesn't change after two successive interval increments. The other proposed heuristic is based on the largely applied stochastic search metaheuristic named VNS, which aims to reach optimum or near-optimum solutions for global combinatorial optimization problems. VNS employs several different neighborhoods to escape local minima and reach global optimality, and it can be decomposed into the successive use of diversification and intensification steps. Also, a special data structure, named as *spatial partition tree*, was designed to efficiently represent a workload (spatial-convex partition) for our proposed neighborhoods. Accordingly, our VNS heuristic exploits the *sub-tree reconstruction* during the diversification step, whereas the intensification step comprises a Variable Neighborhood Descent (VND) — sequence of local searches over the *sub-region transfer*, the *sub-region swap*, and the *splitting hyperplane reallocation* neighborhoods, respectively. Computational experiments were conducted to assess the performance of those proposed heuristic methods. Our computational results reveal that the proposed heuristics either quickly reach optimality or improve the best known solutions for almost all tested realistic instances comprising up to 1000 images and fifteen drones. Further, they exposed

that the VNS heuristic is the fastest method for finding good solutions to a vast number of instances. The decomposition-based heuristic is also effective in most of the tested cases, and it is an efficient method when compared to solving the *CAPsac* formulations by commercial solvers. Although the performance of the decomposition-based heuristic deteriorates for some values of $\hat{T}$ close to unfeasibility, the sensitivity analysis done for the decomposition and VNS heuristics demonstrated that these methods still perform well for varying σ and $\hat{T}$ values.

Once we developed methods able to find optimal and near-optimal solutions, it is equally important to ensure that the proposed methods have reasonable performance under real-life conditions. Thus, Chapter 6 presented the *swarm-powered Optimized 3D Mapping Pipeline* (OptiMaP) for emergency response operations. We described two distinct ways to generate and allocate workloads within the OptiMaP: (i) (CAPsac) VNS-based method of Chapter 5; (ii) (non-CAPsac) a greedy method that does not ensure transmission delays smaller than $\hat{T}$. We also assessed how effective is the CAPsac when optimizing 3D mapping missions exploiting a swarm-powered ad-hoc cloud in a simulator that provides realistic physics, visuals and wireless multi-hop network. The simulations compared the CAPsac approach against the non-CAPsac and the centralized (i.e., offloading all processing tasks to one swarm member) approaches. They demonstrated that the use of OptiMaP significantly reduces the communication delays and completion times when compared with the centralized strategy. Finally, the OptiMaP with the VNS method displays the best completion times and transmission delays, and it led to the most balanced use of the networking and computing resources within the ad-hoc cloud.

7.2 Limitations

Chapter 4 presented formulations to solving the CAPsac as well as ordering cuts that eliminate solutions with unfeasible borders in the continuous space. However, the "*pCAPsac*" formulation presents a large number of equivalent relaxed optimal solutions, partially due to symmetry. Consequently, the convergence time of the branch-and-cut method solving that MILP formulation is increased. Another challenge appears when solving the continuous relaxation of the "*rCAPsac*" formulation through column generation since the problem presents high degeneracy. As a result, the CG method requires long execution times to finish.

Heuristics for optimizing the CAPsac were described in Chapter 5. We noticed that the decomposition-based method may reach suboptimal solutions whenever the stopping condition (i.e., the number of iterations without improvement on the objective function) is not adequate. Thus, it could be problematic to adjust that parameter in real life missions. Concerning the VNS-based heuristic in Chapter 5, the spatial partition tree plays an important

role when searching for solutions, and it is known that it cannot represent all partitions in the solution space. If that data structure is modified to represent more solution configurations, the solution space quickly increase and make the local searches slow. On the other hand, simpler data structures accelerate the local searches but may lead to poor solutions given its restricted solution representation.

In Chapter 6, we have deployed realistic simulations of 3D mapping missions according to the OptiMaP. We have shown that those missions benefit from the OptiMaP, but simulations with a larger number of (real) drones would be required to scale our proposed pipeline. We remarked that the longest transmission times can exceed $\hat{T}$ seconds during our experiments when employing the OptiMaP. Those limitations were due to the fact that both network emulator and MMF are approximations of the wireless network behavior. Naturally, they may deviate from a real-life network and additional adjustments in the MMF should be performed to address this issue.

CHAPTER 8 CONCLUSION AND RECOMMENDATIONS

In this thesis, we investigated how to optimize the use of a swarm-powered ad-hoc cloud for timely manner IoT applications that require clustering, communication, and reliability constraints. Furthermore, contributions were produced to achieve the main objectives described in Chapter 1.

Chapter 4 proposed a new NP-hard optimization problem, namely the *Covering-Assignment Problem for swarm-powered ad-hoc clouds (CAPsac)*, and two MILP formulations ("$pCAPsac$" and "$rCAPsac$") to find optimal solutions to the problem. Chapter 5 presented two near-optimal algorithms to quickly find good solutions for the CAPsac — a decomposition-based heuristic and a VNS-based method. Finally, in Chapter 6, we described and deployed the *swarm-powered Optimized 3D Mapping Pipeline (OptiMaP)*, which uses the CAPsac to generate and allocate the workload carried by the swarm-powered ad-hoc cloud. Besides being relevant for understanding how to exploit the considered ad-hoc infrastructure efficiently, those contributions are crucial to apply the developed techniques in real life missions as our use-case application, i.e., swarm-powered 3D mapping for emergency response operations. However, those contributions present some limitations, leading to new research opportunities.

- The large number of equivalent optimal solutions in the continuous relaxation of the "$pCAPsac$" formulation deteriorates the branch-and-cut performance since the dual bound increases slowly, and the cutting plane procedure take longer to terminate. We noticed that both boundaries constraints (4.10)-(4.13) and the McCormick inequalities (4.3)-(4.5) contribute to that issue. Naturally, designing new cuts and formulations dealing with those constraints can potentially accelerate the branch-and-cut method.

- The column generation of the "$rCAPsac$" formulation presents high degeneracy. The issue may be addressed by applying column generation over a new MILP formulation that selects tasks-to-drone assignments (i.e., sub-regions-to-drone assignments) instead of sets of tasks (sub-regions). In addition, fast pricing procedures need to be developed to select good columns to be added into the restricted master problem given the large number of possible assignments.

- Concerning the VNS method, we noticed that the proposed spatial partition tree restricts the solution space, which might lead to suboptimal solutions. The adjustment of that data structure is essential to expand the considered solution space without

compromising the local search performance.

- The definition a precise interval $[n_\ell, n_u]$ is critical for the decomposition-based method. The creation of the initial interval $[n_\ell, n_u]$ may benefit from machine learning techniques which could indicate good starting values for n_ℓ and n_u.

- The MMF flow control paradigm is a powerful tool to approximate the transmission behavior in networks, but it can deviate when dealing with wireless networks. For the considered timely manner applications, it is crucial to address issues as route round trip times and hidden nodes in the network. They may be tackled by employing the multiplicative parameters proposed in [83] and stochastic programming techniques.

- An alternative way to solve the CAPsac is to develop a real-time method that could start as soon as the data (images) collection procedure begins. The possible gain of performance lies in the fact that the real-time version would already have a reliable solution and the drones would already have performed part of the necessary transmission demands when the photo acquisition step ends. Decentralized approaches would further improve the system robustness.

- The CAPsac assumes that all the communications are performed over a wireless network, but other communication interfaces may be available for some IoT applications. Furthermore, multi-tree communication topology (i.e., one tree for each communication interface) would reduce the load on the wireless network and decrease communication delays. Such approach could also enhance the system reliability since alternative networks could be used in case of drone malfunction.

- A CAPsac variant allowing changes in topology of the communication network may improve the data exchange across the swarm members. Changing the network topology can be performed by moving drones either closer or far from each other. Since moving drones highly drains their batteries, the energy consumed for maneuvering the drones and changing the network topology need to be considered in this CAPsac variant. Furthermore, drones cannot use the new network topology until all drones are moved to their new positions, thus the time required to move the drones directly affects overall mission completion time and should be taken into account as well.

REFERENCES

[1] H. Hejazi, H. Rajab, T. Cinkler, and L. Lengyel, "Survey of platforms for massive IoT," in *2018 IEEE International Conference on Future IoT Technologies (Future IoT)*, 2018, pp. 1–8.

[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[3] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob, "Big IoT data analytics: Architecture, opportunities, and open research challenges," *IEEE Access*, vol. 5, pp. 5247–5261, 2017.

[4] B. P. Rimal, A. Jukan, D. Katsaros, and Y. Goeleven, "Architectural requirements for cloud computing systems: an enterprise cloud approach," *Journal of Grid Computing*, vol. 9, no. 1, pp. 3–26, 2011.

[5] J. Qadir, B. Sainz-De-Abajo, A. Khan, B. García-Zapirain, I. De La Torre-Díez, and H. Mahmood, "Towards mobile edge computing: Taxonomy, challenges, applications and future realms," *IEEE Access*, vol. 8, pp. 189 129–189 162, 2020.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[7] S. Wan, J. Lu, P. Fan, and K. B. Letaief, "Toward big data processing in iot: Path planning and resource management of UAV base stations in mobile-edge computing system," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5995–6009, 2020.

[8] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2049–2063, 2018.

[9] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint task offloading and resource allocation in UAV-enabled mobile edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3147–3159, 2020.

[10] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, "Joint offloading and trajectory design for UAV-enabled mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1879–1892, 2019.

[11] N. Mohamed, J. Al-Jaroodi, I. Jawhar, H. Noura, and S. Mahmoud, "UAVFog: A UAV-based fog computing for internet of things," in *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2017, pp. 1–8.

[12] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *Swarm Robotics*, E. Şahin and W. M. Spears, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 10–20.

[13] S. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855, 2018.

[14] İlker Bekmezci, O. K. Sahingoz, and Şamil Temel, "Flying ad-hoc networks (FANETs): A survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254 – 1270, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870512002193

[15] A. Malhotra, S. K. Dhurandher, and B. Kumar, "Resource allocation in multi-hop mobile ad hoc cloud," in *2014 Recent Advances in Engineering and Computational Sciences (RAECS)*, 2014, pp. 1–6.

[16] M. Hamdaqa, M. M. Sabri, A. Singh, and L. Tahvildari, "Adoop: MapReduce for Ad-Hoc Cloud Computing," in *Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON '15. USA: IBM Corp., 2015, p. 26–34.

[17] I. Yaqoob, E. Ahmed, A. Gani, S. Mokhtar, M. Imran, and S. Guizani, "Mobile ad hoc cloud: A survey," *Wireless Communications and Mobile Computing*, vol. 16, no. 16, pp. 2572–2589, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/wcm.2709

[18] W. Chen, B. Liu, H. Huang, S. Guo, and Z. Zheng, "When UAV swarm meets edge-cloud computing: The QoS perspective," *IEEE Network*, vol. 33, no. 2, pp. 36–43, 2019.

[19] A. Otto, N. Agatz, J. Campbell, B. Golden, and E. Pesch, "Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey," *Networks*, vol. 0, no. 0, 2018.

[20] W. P. Coutinho, M. Battarra, and J. Fliege, "The unmanned aerial vehicle routing and trajectory optimisation problem, a taxonomic review," *Computers*

& Industrial Engineering, vol. 120, pp. 116 – 128, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360835218301815

[21] P. B. Sujit and D. Ghose, "Search using multiple UAVs with flight time constraints," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 2, pp. 491–509, April 2004.

[22] H. Oh, S. Kim, A. Tsourdos, and B. A. White, "Coordinated road-network search route planning by a team of UAVs," *International Journal of Systems Science*, vol. 45, no. 5, pp. 825–840, 2014.

[23] H. Oh, H.-S. Shin, S. Kim, A. Tsourdos, and B. A. White, *Cooperative Mission and Path Planning for a Team of UAVs*. Dordrecht: Springer Netherlands, 2015, pp. 1509–1545.

[24] P. Lanillos, S. K. Gan, E. Besada-Portas, G. Pajares, and S. Sukkarieh, "Multi-UAV target search using decentralized gradient-based negotiation with expected observation," *Information Sciences*, vol. 282, pp. 92 – 110, 2014.

[25] S. K. Gan and S. Sukkarieh, "Multi-UAV target search using explicit decentralized gradient-based negotiation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 751–756.

[26] X. Ji, X. Wang, Y. Niu, and L. Shen, "Cooperative search by multiple unmanned aerial vehicles in a nonconvex environment," *Mathematical Problems in Engineering*, vol. 2015, pp. 1–19, 2015.

[27] Z. Tang and U. Ozguner, "Motion planning for multitarget surveillance with mobile sensor agents," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 898–908, Oct 2005.

[28] S. Karaman and G. Inalhan, "Large-scale task/target assignment for UAV fleets using a distributed branch and price optimization scheme," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 13 310 – 13 317, 2008, 17th IFAC World Congress.

[29] M. Niccolini, M. Innocenti, and L. Pollini, "Multiple UAV task assignment using descriptor functions," *IFAC Proceedings Volumes*, vol. 43, no. 15, pp. 93–98, 2010.

[30] A. Viguria, I. Maza, and A. Ollero, "Distributed service-based cooperation in aerial/ground robot teams applied to fire detection and extinguishing missions," *Advanced Robotics*, vol. 24, no. 1-2, pp. 1–23, 2010.

[31] H. Choi, Y. Kim, and H. Kim, "Genetic algorithm based decentralized task assignment for multiple unmanned aerial vehicles in dynamic environments," *International Journal Aeronautical and Space Sciences*, vol. 12, no. 2, pp. 163–174, 2011.

[32] A. Barrientos, J. Colorado, J. d. Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente, "Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots," *Journal of Field Robotics*, vol. 28, no. 5, pp. 667–689, 2011.

[33] S. Moon, E. Oh, and D. H. Shim, "An integral framework of task assignment and path planning for multiple unmanned aerial vehicles in dynamic environments," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1, pp. 303–313, Apr 2013.

[34] S. Moon, D. H. Shim, and E. Oh, *Cooperative Task Assignment and Path Planning for Multiple UAVs*. Dordrecht: Springer Netherlands, 2015, pp. 1547–1576.

[35] M. Turpin, N. Michael, and V. Kumar, "Capt: Concurrent assignment and planning of trajectories for multiple robots," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 98–112, 2014.

[36] J. J. Enright, E. Frazzoli, M. Pavone, and K. Savla, *UAV Routing and Coordination in Stochastic, Dynamic Environments*. Dordrecht: Springer Netherlands, 2015, pp. 2079–2109.

[37] A. Sadeghi and S. L. Smith, "Heterogeneous task allocation and sequencing via decentralized large neighborhood search," *Unmanned Systems*, vol. 5, no. 02, pp. 79–95, 2017.

[38] D. Zorbas, L. D. P. Pugliese, T. Razafindralambo, and F. Guerriero, "Optimal drone placement and cost-efficient target coverage," *Journal of Network and Computer Applications*, vol. 75, pp. 16 – 31, 2016.

[39] P. Ladosz, H. Oh, and W.-H. Chen, "Trajectory planning for communication relay unmanned aerial vehicles in urban dynamic environments," *Journal of Intelligent & Robotic Systems*, vol. 89, no. 1, pp. 7–25, Jan 2018.

[40] L. Caraballo, J. Díaz-Báñez, I. Maza, and A. Ollero, "The block-information-sharing strategy for task allocation: A case study for structure assembly with aerial robots," *European Journal of Operational Research*, vol. 260, no. 2, pp. 725 – 738, 2017.

[41] A. Grancharova, E. I. Grøtli, D.-T. Ho, and T. A. Johansen, "UAVs trajectory planning by distributed MPC under radio communication path loss constraints," *Journal of Intelligent & Robotic Systems*, vol. 79, no. 1, pp. 115–134, 2015.

[42] S. Koulali, E. Sabir, T. Taleb, and M. Azizi, "A green strategic activity scheduling for UAV networks: A sub-modular game perspective," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 58–64, May 2016.

[43] S. Xu, K. Doğançay, and H. Hmam, "Distributed pseudolinear estimation and UAV path optimization for 3D AOA target tracking," *Signal Processing*, vol. 133, pp. 64–78, 2017.

[44] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," in *Cooperative Robots and Sensor Networks 2015*. Springer, 2015, pp. 31–51.

[45] M. A. Gomarasca, *Elements of Photogrammetry*. Dordrecht: Springer Netherlands, 2009, ch. 3, pp. 79–121.

[46] W. Linder, *Digital Photogrammetry*. Springer Berlin Heidelberg, 2016.

[47] D. Meyer, E. Fraijo, E. Lo, D. Rissolo, and F. Kuester, "Optimizing UAV systems for rapid survey and reconstruction of large scale cultural heritage sites," in *Digital Heritage, 2015*, vol. 1. IEEE, 2015, pp. 151–154.

[48] P. Doherty, J. Kvarnström, P. Rudol, M. Wzorek, G. Conte, C. Berger, T. Hinzmann, and T. Stastny, "A collaborative framework for 3D mapping using unmanned aerial vehicles," in *International Conference on Principles and Practice of Multi-Agent Systems*. Springer, 2016, pp. 110–130.

[49] G. Loianno, Y. Mulgaonkar, C. Brunner, D. Ahuja, A. Ramanandan, M. Chari, S. Diaz, and V. Kumar, "A swarm of flying smartphones," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 1681–1688.

[50] J. Schmiemann, H. Harms, J. Schattenberg, M. Becker, S. Batzdorfer, and L. Frerichs, "A distributed online 3D-lidar mappin system," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-2/W6, pp. 339–346, 2017.

[51] K. Kobayashi, H. Shishido, Y. Kameda, and I. Kitahara, "Method to generate disaster-damage map using 3D photometry and crowd sourcing," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 4397–4399.

[52] S. Golodetz, T. Cavallari, N. A. Lord, V. A. Prisacariu, D. W. Murray, and P. H. Torr, "Collaborative large-scale dense 3D reconstruction with online inter-agent pose optimisation," *arXiv preprint arXiv:1801.08361*, 2018.

[53] T. Hinzmann, T. Stastny, G. Conte, P. Doherty, P. Rudol, M. Wzorek, E. Galceran, R. Siegwart, and I. Gilitschenski, "Collaborative 3D reconstruction using heterogeneous UAVs: System and experiments," in *International Symposium on Experimental Robotics*. Springer, 2016, pp. 43–56.

[54] N. Kerle, F. Nex, M. Gerke, D. Duarte, and A. Vetrivel, "UAV-based structural damage mapping: A review," *ISPRS International Journal of Geo-Information*, vol. 9, no. 1, 2020. [Online]. Available: https://www.mdpi.com/2220-9964/9/1/14

[55] N. Nikhil, S. M. Shreyas, G. Vyshnavi, and S. Yadav, "Unmanned aerial vehicles (UAV) in disaster management applications," in *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, 2020, pp. 140–148.

[56] C. Liu, H. Sui, and L. Huang, "Identification of building damage from UAV-based photogrammetric point clouds using supervoxel segmentation and latent dirichlet allocation model," *Sensors*, vol. 20, no. 22, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/22/6499

[57] M. Aljehani and M. Inoue, "Performance evaluation of multi-UAV system in post-disaster application: Validated by HITL simulator," *IEEE Access*, vol. 7, pp. 64 386–64 400, 2019.

[58] Y. Tan and Z.-Y. Zheng, "Research advance in swarm robotics," *Defence Technology*, vol. 9, no. 1, pp. 18 – 39, 2013.

[59] L. Bayındır, "A review of swarm robotics tasks," *Neurocomputing*, vol. 172, pp. 292–321, 2016.

[60] P. B. Sujit and D. Ghose, "Two-agent cooperative search using game models with endurance-time constraints," *Engineering Optimization*, vol. 42, no. 7, pp. 617–639, 2010.

[61] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[62] A. Rubinstein, "Perfect equilibrium in a bargaining model," *Econometrica: Journal of the Econometric Society*, pp. 97–109, 1982.

[63] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[64] S. Ghandi and E. Masehian, "Review and taxonomies of assembly and disassembly path planning problems and approaches," *Computer-Aided Design*, vol. 67, pp. 58–86, 2015.

[65] A. Fotouhi, M. Ding, and M. Hassan, "Dronecells: Improving 5G spectral efficiency using drone-mounted flying base stations," *arXiv preprint arXiv:1707.02041*, 2017.

[66] B. P. Rimal and I. Lumb, *The Rise of Cloud Computing in the Era of Emerging Networked Society*. Cham: Springer International Publishing, 2017, pp. 3–25. [Online]. Available: https://doi.org/10.1007/978-3-319-54645-2_1

[67] Y. Mansouri and M. A. Babar, "A review of edge computing: Features and resource virtualization," *Journal of Parallel and Distributed Computing*, vol. 150, pp. 155–183, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731520304317

[68] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18319903

[69] Z. Pang, L. Sun, Z. Wang, E. Tian, and S. Yang, "A survey of cloudlet based mobile computing," in *2015 International Conference on Cloud Computing and Big Data (CCBD)*, 2015, pp. 268–275.

[70] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47 980–48 009, 2018.

[71] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[72] D. F. Pigatto, M. Rodrigues, J. V. de Carvalho Fontes, A. S. R. Pinto, J. Smith, and K. R. L. J. C. Branco, *The Internet of Flying Things*. John Wiley & Sons, Ltd, 2018, ch. 19, pp. 529–562. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119456735.ch19

[73] S. Jeong, O. Simeone, and J. Kang, "Mobile cloud computing with a UAV-mounted cloudlet: optimal bit allocation for communication and computation," *IET Communications*, vol. 11, no. 7, pp. 969–974, Mar. 2017. [Online]. Available: https://doi.org/10.1049/iet-com.2016.1114

[74] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1927–1941, 2018.

[75] X. Hu, K.-K. Wong, K. Yang, and Z. Zheng, "UAV-assisted relaying and edge computing: Scheduling and trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4738–4752, 2019.

[76] M.-A. Messous, H. Sedjelmaci, N. Houari, and S.-M. Senouci, "Computation offloading game for an UAV network in mobile edge computing," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[77] A. Asheralieva and D. Niyato, "Hierarchical game-theoretic and reinforcement learning framework for computational offloading in UAV-enabled mobile edge computing networks with multiple service providers," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8753–8769, 2019.

[78] Y. Wang, Z.-Y. Ru, K. Wang, and P.-Q. Huang, "Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3984–3997, 2020.

[79] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data networks*. Prentice-Hall International New Jersey, 1992, vol. 2.

[80] D. Nace and M. Pióro, "Max-min fairness and its applications to routing and load-balancing in communication networks: a tutorial," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, 2008.

[81] E. Amaldi, A. Capone, S. Coniglio, and L. G. Gianoli, "Network optimization problems subject to max-min fair flow allocation," *IEEE Communications Letters*, vol. 17, no. 7, pp. 1463–1466, 2013.

[82] L. Massoulié and J. Roberts, "Bandwidth sharing: Objectives and algorithms," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 1999, pp. 1395–1403.

[83] S. Coniglio, L. G. Gianoli, E. Amaldi, and A. Capone, "Elastic traffic engineering subject to a fair bandwidth allocation via bilevel programming," *IEEE/ACM Transactions on Networking*, 2020.

[84] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4104–4113.

[85] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, June 2006, pp. 519–528.

[86] Y. Furukawa, C. Hernández *et al.*, "Multi-view stereo: A tutorial," *Foundations and Trends in Computer Graphics and Vision*, vol. 9, no. 1-2, pp. 1–148, 2015.

[87] P. Tang, S. Vick, J. Chen, and S. German Paal, "Chapter 2 - surveying, geomatics, and 3D reconstruction," in *Infrastructure Computer Vision*, I. Brilakis and C. Haas, Eds. Butterworth-Heinemann, 2020, pp. 13 – 64. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780128155035000024

[88] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, p. 133, 1981.

[89] O. Özyeşil, V. Voroninski, R. Basri, and A. Singer, "A survey of structure from motion." *Acta Numerica*, vol. 26, p. 305–364, 2017.

[90] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[91] Y. Ke and R. Sukthankar, "PCA-SIFT: a more distinctive representation for local image descriptors," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, June 2004, pp. II–506–II–513 Vol.2.

[92] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.

[93] M. Bolitho, M. Kazhdan, R. Burns, and H. Hoppe, "Parallel poisson surface reconstruction," in *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koracin,

Y. Kuno, J. Wang, J.-X. Wang, J. Wang, R. Pajarola, P. Lindstrom, A. Hinkenjann, M. L. Encarnação, C. T. Silva, and D. Coming, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 678–689.

[94] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 3, p. 29, 2013.

[95] M. Waechter, N. Moehrle, and M. Goesele, "Let there be color! Large-scale texturing of 3D reconstructions," in *ECCV 2014*, ser. Lecture Notes in Computer Science, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, 2014, vol. 8693, pp. 836–850.

[96] M. Pepe, L. Fregonese, and M. Scaioni, "Planning airborne photogrammetry and remote-sensing missions with modern platforms and sensors," *European Journal of Remote Sensing*, vol. 51, no. 1, pp. 412–436, 2018.

[97] F. Nex and F. Remondino, "UAV for 3D mapping applications: a review," *Applied Geomatics*, vol. 6, no. 1, pp. 1–15, 2014.

[98] J. A. J. Berni, P. J. Zarco-Tejada, L. Suarez, and E. Fereres, "Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 3, pp. 722–738, March 2009.

[99] G. Grenzdörffer, A. Engel, and B. Teichert, "The photogrammetric potential of low-cost UAVs in forestry and agriculture," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXVII, no. B1, pp. 1207–1214, 2008.

[100] J. Martínez-de Dios, L. Merino, F. Caballero, A. Ollero, and D. Viegas, "Experimental results of automatic fire detection and monitoring with UAVs," *Forest Ecology and Management*, vol. 234, no. 1, p. S232, 2006.

[101] F. Chiabrando, F. Nex, D. Piatti, and F. Rinaudo, "UAV and RPV systems for photogrammetric surveys in archaelogical areas: two tests in the Piedmont region (Italy)," *Journal of Archaeological Science*, vol. 38, no. 3, pp. 697–710, 2011.

[102] K. Lambers, H. Eisenbeiss, M. Sauerbier, D. Kupferschmidt, T. Gaisecker, S. Sotoodeh, and T. Hanusch, "Combining photogrammetry and laser scanning for the recording and modelling of the late intermediate period site of Pinchango Alto, Palpa, Peru," *Journal of Archaeological Science*, vol. 34, no. 10, pp. 1702–1712, 2007.

[103] M. Oczipka, J. Bemmann, H. Piezonka, J. Munkabayar, B. Ahrens, M. Achtelik, and F. Lehmann, "Small drones for geo-archaeology in the steppes: locating and documenting the archaeological heritage of the Orkhon Valley in Mongolia," in *Remote Sensing for Environmental Monitoring, GIS Applications, and Geology IX*, U. Michel and D. L. Civco, Eds., vol. 7478, International Society for Optics and Photonics. SPIE, 2009, pp. 53 – 63.

[104] F. Rinaudo, F. Chiabrando, A. M. Lingua, and A. Spanò, "Archaeological site monitoring: UAV photogrammetry can be an answer," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXIX-B5, pp. 583–588, 2012.

[105] G. J. J. Verhoeven, "Providing an archaeological bird's-eye view – an overall picture of ground-based means to execute low-altitude aerial photography (LAAP) in archaeology," *Archaeological Prospection*, vol. 16, no. 4, pp. 233–249, 2009.

[106] W. Hartmann, S. Tilch, H. Eisenbeiss, and K. Schindler, "Determination of the UAV position by automatic processing of thermal images," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXIX-B6, pp. 111–116, 2012.

[107] M. Manyoky, P. Theiler, D. Steudler, and H. Eisenbeiss, "Unmanned aerial vehicle in cadastral applications," *ISPRS-international archives of the photogrammetry, remote sensing and spatial information sciences*, vol. XXXVIII-1/C22, pp. 57–62, 2011.

[108] U. Niethammer, S. Rothmund, M. James, J. Travelletti, and M. Joswig, "UAV-based remote sensing of landslides," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, no. Part 5, pp. 496–501, 2010.

[109] J. G. Smith, J. Dehn, R. P. Hoblitt, R. G. LaHusen, J. B. Lowenstern, S. C. Moran, L. McClelland, K. A. McGee, M. Nathenson, P. G. Okubo, J. S. Pallister, M. P. Poland, J. A. Power, D. J. Schneider, and T. W. Sisson, "Volcano monitoring," in *Geological Monitoring*. Geological Society of America, 2009, pp. 273–305.

[110] C. Zhang, "An UAV-based photogrammetric mapping system for road condition assessment," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXVII-B5, pp. 627–632, 2008.

[111] T.-Y. Chou, M.-L. Yeh, Y. C. Chen, and Y. H. Chen, "Disaster monitoring and management by the unmanned aerial vehicle technology," in *International Archives of Pho-*

togrammetry, Remote Sensing and Spatial Information Sciences, vol. 38, no. 7B, 2010, p. 137–142.

[112] R. Haarbrink and E. Koers, "Helicopter UAV for photogrammetry and rapid response," in *2nd Int. Workshop "The Future of Remote Sensing", ISPRS Inter-Commission Working Group I/V Autonomous Navigation*, vol. 1. Citeseer, 2006.

[113] P. Molina, I. Colomina, T. Victoria, J. Skaloud, W. Kornus, R. Prades, and C. Aguilera, "Searching lost people with UAVs: The system and results of the CLOSE-SEARCH project," in *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 39, no. EPFL-CONF-182482, 2012, pp. 441–446.

[114] P. Zarco-Tejada, R. Diaz-Varela, V. Angileri, and P. Loudjani, "Tree height quantification using very high resolution imagery acquired from an unmanned aerial vehicle (UAV) and automatic 3D photo-reconstruction methods," *European Journal of Agronomy*, vol. 55, pp. 89 – 99, 2014.

[115] J. Bendig, A. Bolten, S. Bennertz, J. Broscheit, S. Eichfuss, and G. Bareth, "Estimating biomass of barley using crop surface models (CSMs) derived from UAV-based RGB imaging," *Remote Sensing*, vol. 6, no. 11, pp. 10 395–10 412, 2014.

[116] R. A. Díaz-Varela, R. De la Rosa, L. León, and P. J. Zarco-Tejada, "High-resolution airborne UAV imagery to assess olive tree crown parameters using 3D photo reconstruction: Application in breeding trials," *Remote Sensing*, vol. 7, no. 4, pp. 4213–4232, 2015.

[117] H. Surmann, N. Berninger, and R. Worst, "3D mapping for multi hybrid robot cooperation," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 626–633.

[118] J. Roters, F. Steinicke, and K. H. Hinrichs, "Quasi-real-time 3D reconstruction from low-altitude aerial images," in *Proc. of the 28th Urban Data Management Symposium*, 2011, pp. 231–241.

[119] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, vision-based flight and live dense 3D mapping with a quadrotor micro aerial vehicle," *Journal of Field Robotics*, vol. 33, no. 4, pp. 431–450, 2016.

[120] S. Huh, S. Hong, and J. Lee, "Energy-efficient distributed programming model for swarm robot," in *Control, Automation and Systems (ICCAS), 2013 13th International Conference on*. IEEE, 2013, pp. 300–305.

[121] S. Milani and A. Memo, "Impact of drone swarm formations in 3D scene reconstruction," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, Sep. 2016, pp. 2598–2602.

[122] M. Darrah, M. M. Trujillo, K. Speransky, and M. Wathen, "Optimized 3D mapping of a large area with structures using multiple multirotors," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, June 2017, pp. 716–722.

[123] L. Wei, X. Jin, and Z. Wu, "3d modeling based on multiple unmanned aerial vehicles with the optimal paths," in *2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*. IEEE, Oct 2016, pp. 1–6.

[124] L. R. Costa, D. Aloise, L. G. Gianoli, and A. Lodi, "The Covering-Assignment Problem for Swarm-Powered Ad Hoc Clouds: A Distributed 3-D Mapping Usecase," *IEEE Internet of Things Journal*, vol. 8, no. 9, pp. 7316–7332, 2021.

[125] L. R. Costa, D. Aloise, L. G. Gianoli, and A. Lodi, "Heuristics for optimizing 3D mapping missions over swarm-powered ad hoc clouds," 2021. [Online]. Available: http://arxiv.org/abs/2103.06953

[126] H. Mei, K. Yang, Q. Liu, and K. Wang, "Joint trajectory-resource optimization in UAV-enabled edge-cloud system with virtualized mobile clone," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5906–5921, 2020.

[127] X. Wei, C. Tang, J. Fan, and S. Subramaniam, "Joint optimization of energy consumption and delay in cloud-to-thing continuum," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2325–2337, 2019.

[128] J. Zhang, L. Zhou, Q. Tang, E. C.-H. Ngai, X. Hu, H. Zhao, and J. Wei, "Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3688–3699, 2019.

[129] Z. Tan, H. Qu, J. Zhao, S. Zhou, and W. Wang, "UAV-aided Edge/Fog computing in smart IoT community for social augmented reality," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4872–4884, 2020.

[130] T. Yu, X. Wang, and A. Shami, "UAV-enabled spatial data sampling in large-scale IoT systems using denoising autoencoder neural network," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1856–1865, 2019.

[131] CBC/Radio-Canada. (2020, April) Émission découverte: Drones humanitaires. [Online]. Available: https://ici.radio-canada.ca/tele/decouverte/site/segments/reportage/139538/drones-humanitaires

[132] W. Gay, *Raspberry Pi Hardware Reference*, 1st ed. USA: Apress, 2014.

[133] L. G. Gianoli, April 2020, private communication.

[134] P. Huang, Y. Wang, K. Wang, and K. Yang, "Differential evolution with a variable population size for deployment optimization in a UAV-assisted IoT data collection system," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 3, pp. 324–335, 2020.

[135] A. Raniwala, P. De, S. Sharma, R. Krishnan, and T. Chiueh, "End-to-end flow fairness over IEEE 802.11-based wireless mesh networks," in *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*. IEEE, 2007, pp. 2361–2365.

[136] D. Bertsimas and M. Sim, "Robust discrete optimization and network flows," *Mathematical Programming*, vol. 98, no. 1-3, pp. 49–71, sep 2003. [Online]. Available: http://www.springerlink.com/openurl.asp?genre=article{&}id=doi:10.1007/s10107-003-0396-4

[137] E. Delage, L. G. Gianoli, and B. Sansò, "A practicable robust counterpart formulation for decomposable functions: A network congestion case study," *Operations Research*, vol. 66, no. 2, pp. 535–567, 2018.

[138] G. P. McCormick, "Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems," *Mathematical programming*, vol. 10, no. 1, pp. 147–175, 1976.

[139] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto, "Optimal packing and covering in the plane are NP-complete," *Information Processing Letters*, vol. 12, no. 3, pp. 133 – 137, 1981.

[140] D. S. Johnson, "The NP-completeness column: An ongoing guide," *Journal of Algorithms*, vol. 3, no. 2, pp. 182 – 195, 1982.

[141] G. Desaulniers, J. Desrosiers, and M. M. Solomon, Eds., *Column Generation*. Springer US, 2005. [Online]. Available: https://doi.org/10.1007/b135457

[142] A. Lodi, "Mixed integer programming computation," in *50 Years of Integer Programming 1958-2008*. Springer, Berlin, Heidelberg, 2010, pp. 619–645.

[143] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, Jan. 2002. [Online]. Available: https://doi.org/10.1007/s101070100263

[144] R. Jain, *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. New York: Wiley, 1991.

[145] OpenDroneMap, "CloudODM," 2020. [Online]. Available: https://github.com/OpenDroneMap/CloudODM

[146] P. Hansen and N. Mladenović, "Variable neighborhood search: Principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, pp. 449 – 467, 2001.

[147] P. Hansen, N. Mladenović, J. Brimberg, and J. A. M. Pérez, *Variable Neighborhood Search*. Cham: Springer International Publishing, 2019, pp. 57–97. [Online]. Available: https://doi.org/10.1007/978-3-319-91086-4_3

[148] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82 – 117, 2013, prediction, Control and Diagnosis using Advanced Neural Computations.

[149] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, p. 509–517, Sep. 1975.

[150] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[151] (2021) Heterogeneous Embedded Ad-hoc Virtual Emergency Network (HEAVEN) middleware. Humanitas. [Online]. Available: https://www.humanitas.io/off-grid-networking

[152] V. Boyavalle, March 2021, private communication.

[153] J. Turner, Q. Meng, G. Schaefer, A. Whitbrook, and A. Soltoggio, "Distributed task rescheduling with time constraints for the optimization of total task allocations in a multirobot system," *IEEE Transactions on Cybernetics*, vol. 48, no. 9, pp. 2583–2597, 2018.

[154] I. Jang, H. Shin, and A. Tsourdos, "Anonymous hedonic game for task allocation in a large-scale multiple agent system," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1534–1548, 2018.

[155] S. Mayya, D. S. D'antonio, D. Saldaña, and V. Kumar, "Resilient task allocation in heterogeneous multi-robot systems," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1327–1334, 2021.

[156] M. Alshaboti and U. Baroudi, "Multi-robot task allocation system: Fuzzy auction-based and adaptive multi-threshold approaches," *SN Computer Science*, vol. 2, no. 2, Feb. 2021. [Online]. Available: https://doi.org/10.1007/s42979-021-00479-x

[157] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097 – 1100, 1997.

[158] (2021) ODM - a command line toolkit to generate maps, point clouds, 3D models and DEMs from drone, balloon or kite images. OpenDroneMap/ODM. [Online]. Available: https://github.com/OpenDroneMap/ODM

[159] (2021) Empowering app development for developers. Docker inc. [Online]. Available: https://www.docker.com

[160] (2021) Hyper-realistic simulation-based innovation platform|Hyper-X-Space. Humanitas. [Online]. Available: https://www.hxs.ai/

[161] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017. [Online]. Available: https://arxiv.org/abs/1705.05065