Auction-based task allocation for online pickup and delivery problems with time constraints and uncertainty

Paolo Rizzo

TUDelft

This page was intentionally left blank

Auction-based task allocation for online pickup and delivery problems with time constraints and uncertainty

by

Paolo Rizzo

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday February 19, 2021

Student number:	4466101	
Thesis committee:	Member	Department
Supervisor TU Delft	Dr. O.A. Sharpanskykh	TU Delft - Air Transport & Operations
Chair	Dr. G.C.H.E. de Croon	TU Delft - Control & Simulation
Examiner	Dr. A. Bombelli	TU Delft - Air Transport & Operations

TU Delft

This page was intentionally left blank

Acknowledgements

I would like to thank my supervisor Alexei Sharpanskykh for the continuous guidance and support. Alexei's contagious passion for multi-agent systems, and for bridging the gap between theory and application, has been a source of inspiration since the first time I stepped into one of his lectures. Thank you for always believing in my ideas and giving me so much space to develop them, while pushing me to never forget the bigger picture. It has been a true pleasure to work with you. I was also fortunate enough to count on the expertise of several others, in particular Johan Los, whose feedback in the early stages of the project has been very valuable in shaping my research.

Thanks to the entire TU Delft community. I came to Delft looking for opportunities to grow, and have not been disappointed. I have been deeply challenged, motivated, and inspired by the people I have met and the community I have been a part of over the past few years. Thanks in particular to Jacopo Serpieri, who supervised a research project during my bachelor and encouraged me to go present it and publish it. I'm grateful for the opportunity you gave me so early in my studies, which in many ways inspired me to take on several other challenges over the past few years.

A special thank you to my girlfriend Isabella, who's never stopped rooting for me. And to the friends who have always been by my side and made this all such an enjoyable ride.

My deepest gratitude goes to my family. To my little sisters and to my parents. Despite the difficult past few months, you've never stopped supporting me and pushing me to power through. I know I can always count on you.

Paolo Rizzo
Delft, February 2021

Contents

List of Figures

List of Tables

I

Scientific Paper

Auction-based task allocation for online pickup and delivery problems with time constraints and uncertainty

Paolo Rizzo[*], **Alexei Sharpanskykh**[†]

Section of Air Transport Operations, Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, 2629HS Delft, The Netherlands

Auctions have established themselves as highly efficient mechanisms for the online allocation of time-constrained pickup and delivery tasks, which is an important problem in the domain of distributed autonomous systems. Current methods leverage simple temporal networks (STNs) to allow agents to efficiently process temporal constraints in the bidding phase of the auction. However, they suffer from two major weaknesses which limit their applicability in real-world systems. Firstly, they are relatively ineffective when tasks have non-deterministic durations, as the STN representation enforces a binary notion of plan controllability which is highly restrictive. We remedy this by introducing the probabilistic temporal sequential single item auction (pTeSSI), a novel polynomial-time auction-based task allocation mechanism in which plans are represented as simple temporal networks with uncertainty (STNUs). Using a recently proposed non-binary characterization of controllability in STNUs, agents efficiently determine the risk of unsuccessful dispatch of their temporal plans and incorporate this in their bids. We evaluate our auction mechanism in an online simulation of an on-demand UAV delivery system and demonstrate that it is more effective and efficient than the current state of the art method. In addition, we propose a dynamic re-auctioning routine to address the second main weakness of sequential auctions when applied to online problems, namely that they do not revisit existing partial allocations over time. We demonstrate that dynamic re-auctioning increases the quality of the allocation and improves system performance, but also increases the auction duration. We mitigate this downside by bundling tasks based on their spatio-temporal synergy and auctioning bundles, rather than single tasks, at once.

I. Introduction

Auctions are highly efficient mechanisms for the allocation of tasks to teams of agents. They allow for the vast majority of the computation to be performed in a distributed manner and lead to substantially less communication overhead than centralized approaches. This makes them highly effective for online use in dynamic problems, where agents need to attend to a constant stream of tasks whose details are not known beforehand. An interesting subclass of these problems which has received little attention in the literature concerns the allocation of pickup and delivery tasks that must be executed within certain time windows. The presence of time constraints makes the allocation problem substantially more complex as agents are required to reason about the temporal consistency of their plans, rather than only spatial synergies, while evaluating their bid for a task.

Despite having received little attention in the literature, the dynamic allocation of temporally constrained tasks has become ubiquitous as we attempt to achieve increased autonomy in complex logistics systems. A modern example is UAV delivery, which has the potential to unlock considerable economic benefits by dramatically cutting costs in the last mile of global supply chains, while also enabling the on-demand delivery of wide ranges of products at unmatched speed. In order to economically deploy and manage such systems at scale, highly efficient mechanisms are required to allocate delivery tasks online in dynamic and uncertain environments, while satisfying time constraints specified by customers.

A promising approach is the Temporal Sequential Single Item (TeSSI) auction [21], a variant of the sequential single-item auction in which agents represent their schedule as a simple temporal network (STN). This allows agents to efficiently propagate the temporal constraints associated to incoming tasks and determine their optimal insertion point within their schedule while bidding. Agents bid the makespan of the schedule associated to the optimal insertion position and thereby attempt to minimize the maximum path cost across the entire team [14]. While highly effective at processing tasks' temporal constraints, TeSSI suffers from two major shortcomings that limit its applicability in real-world systems. The first is that the STN representation is too rigid to allow the agents to represent and reason about possible sources of uncertainty in task durations. The second is that there is no efficient re-auctioning routine to allow the agents to swap tasks. Once the choice is made to allocate a task to an agent, it will not be revisited, limiting the synergies that TeSSI can explore and thereby the overall quality of the allocation.

Our work enhances the applicability of auctions for the online allocation of temporally constrained tasks in real-world systems through the following contributions:

- We propose the probabilistic Temporal Sequential Single Item auction (pTeSSI), a variant of TeSSI in which the agents's schedules are represented as

[*]M.Sc. Student, Faculty of Aerospace Engineering
[†]Assistant Professor, Faculty of Aerospace Engineering

STNUs. When inserting a task into their schedule, agents determine the degree of dynamic controllability [1] of the resulting STNU to obtain the risk of unsuccessful dispatch, which is then incorporated in the agent's bid. pTeSSI enables the allocation of temporally constrained tasks with uncertainty by allowing agents to evaluate and bid on non-deterministic plans.
- We devise a periodic re-auctioning routine for pTeSSI, accelerated through a bundling mechanism that groups tasks based on a measure of their spatio-temporal synergy. Bundles, rather than single tasks, are auctioned at once and the temporal relationship between tasks in the bundle is leveraged to further accelerate the bidding phase.
- We provide an analysis and experimental evaluation of the proposed algorithms in a simulation of an on-demand UAV food delivery system, demonstrating that they substantially improve the performance of the TeSSI auction in a complex and dynamic real-world task allocation problem.

II. The pTeSSI auction

We investigate the setting in which delivery tasks are generated dynamically and are to be allocated online to a team of cooperative autonomous agents. Each delivery task T_i includes a pickup location P_i, delivery location D_i and time window $d_i = [ed_i, ld_i]$ in which the item must be delivered. Note that the delivery windows are hard constraints, and that delivery of the item at a time $t \notin d_i$ is regarded as a failure to complete the task. The novelty is that task durations are not required to be deterministic as in previous work on temporal auctions [21]. Rather, durations are random variables with known distributions or bounds on their realization. This allows us to address scenarios in which task durations cannot be known exactly at the time of allocation, which is often the case in real-world dynamic settings.

The pTeSSI auction is a variant of the sequential single item auction [12] and is based on the same driving logic of dividing the task allocation problem into components that are solved locally and in parallel by the agents. A dedicated auctioneer is responsible for announcing tasks and their details to the agents, collecting their bids, choosing and communicating the winner. When a set of tasks $T = \{T_1, ..., T_m\}$ is put on auction to a team of agents $A = \{A_1, ..., A_n\}$, all agents determine their bid for all tasks and submit their best bids to the auctioneer. The task T_* that yielded the lowest bid among the entire set of bids (for all tasks) is allocated to the agent A_* with the lowest bid. The procedure is then repeated for $T \backslash \{T_*\}$. In the next round, agent A_* needs to re-evaluate all tasks since its schedule has changed and therefore its bid for the remaining tasks may be different. For the other agents, this is not necessary as their schedules have not changed. Tasks that cannot be assigned to any agent are simply deleted from the task set, and the auction continues until the auctioneer has attempted to allocate all tasks. Straightforwardly, the auction lasts $|T| = m$ rounds.

A. Bidding rules

Early theoretical work on sequential single item auctions demonstrated that bids generated in a hill-climbing fashion lead to low and tractable values of the team objective [24] while requiring minimal communication. At each round, the agents bid an estimate of the marginal cost the team would incur if they were to add the task to their local plan, without requiring any information about the other agents' plans.

We consider two different team objectives. The first is minimizing the time taken for the team to execute the latest task in the set on auction (this is a common objective in the auction-based routing literature and often referred to as MiniMAX [14]). As demonstrated by Nunes and Gini [21], it is highly effective in an online setting as it maximizes the availability of the team to accommodate future tasks. The second team objective we consider is the sum of the distances travelled by the agents to accommodate all tasks in the set on auction (a variant of MiniSUM [24]). To capture the fact that our agent's plans are probabilistic, we include the risk of unsuccessful dispatch in both bidding rules.

Let $\mathcal{S} = \{S_1, ..., S_m\}$ represent the final allocation of tasks to agents where S_i contains all tasks in agent A_i's schedule. $\mathcal{M}_i$ denotes the minimum time it takes agent A_i to execute all tasks in its schedule S_i, and let $\mathcal{D}_i$ represent the minimum distance agent i must travel to complete its schedule. In addition, let $\mathcal{R}_i(S_i)$ represent the probability that the schedule will not be dispatched successfully and that therefore time $\mathcal{M}_i$ and distance $\mathcal{D}_i$ cannot be achieved. Then we seek to minimize the following team objectives $\mathcal{F}(S)$:

$$\texttt{MAX-T:}\quad \mathcal{F}(S) = \max_i \left[\mathcal{M}_i(S_i) + \rho \cdot \mathcal{R}_i(S_i)\right] \quad (1)$$

$$\texttt{SUM-DIST:}\quad \mathcal{F}(S) = \sum_i \left[\mathcal{D}_i(S_i) + \rho \cdot \mathcal{R}_i(S_i)\right] \quad (2)$$

where ρ is a parameter of the auction that dictates how heavily the risk ought to be penalized. We normalize $\mathcal{M}_i$ and $\mathcal{D}_i$ into the [0,1] range, which is also the range of the scheduling risk $\mathcal{R}_i$, in order to make ρ scale-agnostic and easier to interpret. Suppose that at a given round we have already allocated a subset of tasks and generated a partial allocation $\mathcal{P}$ where $\mathcal{P} = \{P_1, ..., P_n\}$ are the current agent schedules. Let $\mathcal{P}' = \{P'_1, ..., P'_n\}$ be the

allocation resulting from agent A_i being allocated task T, so that:

$$P'_j = \begin{cases} P_j \cup \{T\} & \text{, for } j = i \\ P_j & \text{, for } j \neq i \end{cases} \quad (3)$$

where the subscript j refers to the j-th element of the set. Then agent A_i's bid for T should be equal to the difference $\mathcal{F}(\mathcal{P}') - \mathcal{F}(\mathcal{P})$ in the team objective. For the MAX-T objective, we can rewrite this as:

$$\mathcal{M}_i(P'_i) + \rho \cdot \mathcal{R}_i(P'_i) - \mathcal{F}(\mathcal{P}) \quad (4)$$

since the fact that task T has not been allocated in any previous round of the auction implies $\mathcal{F}(P'_i) = \max_i \mathcal{M}_i(P'_i) + \rho \cdot \mathcal{R}_i(P'_i)$. The system-level factor $\mathcal{F}(\mathcal{P}')$ can be dropped as it is a constant across all agents' bids and does not affect the winner determination. Thus, agent A_i's bid for task T is:

$$\mathcal{M}_i(P'_i) + \rho \cdot \mathcal{R}_i(P'_i) \quad (5)$$

For the SUM-DIST objective, instead, the agent's bid is:

$$\mathcal{D}_i(P'_i) + \rho \cdot \mathcal{R}_i(P'_i) - (\mathcal{D}_i(P_i) + \rho \cdot \mathcal{R}_i(P_i)) \quad (6)$$

Determining $\mathcal{D}_i$ and $\mathcal{M}_i$ involves solving a TSP with time windows and uncertainty for each agent, a highly complex NP-hard optimization problem, which is intractable to solve optimally even for moderate numbers of tasks. Framing the local planning problem as a simple temporal problem allows us to compute both parameters as well as the scheduling risk $\mathcal{R}_i$ in a highly efficient manner which we outline in the rest of this section.

B. Task scheduling and bidding

We start by describing the case in which all task durations are uniformly distributed, and in Section II.D describe how the representation can be extended to any distribution with a known density function.

Agents represent their schedule as a simple temporal network with uncertainty, in which tasks are sequences of nodes (timepoints) and edges. A pickup and delivery task T_i can be divided into four distinct *actions*, namely t_{pi} - traveling to the pickup location; p_i - performing the pickup action; t_{di} - traveling to the delivery location; and d_i - performing the delivery action. We denote the durations of these actions as TT_{pi}, P_i, TT_{di} and D_i respectively. Figure 1 (right) illustrates a two-task example of the general case in which all actions' durations are non-deterministic. Each action consists of a start node which can be scheduled by the agent (denoted by s) and a finish node (denoted by f) which is scheduled by "Nature", an external entity the agent cannot control. Start and finish nodes of an action a are connected by a *contingent* constraint $f_a - s_a = X_a$ where X_a is a uniformly distributed random variable, represented as a curved arrow. We show the interval $[X_a^l, X_a^u]$ along the curved edge, where X_a^l and X_a^u are the lower and upper bounds of the distribution of X_a respectively. We do this to reflect our knowledge that the realization of the contingent edge will be such that $f_a - s_a \in [X_a^l, X_a^u]$. *Requirement* constraints are represented as straight arrows and used to enforce the sequence in which tasks and actions must occur. For instance, that the delivery of an item can only be initiated once the agent has finished traveling to the delivery location: $sd_i - fp_i \in [0, \infty)$. All nodes are scheduled relative to an origin timepoint, which takes a value of zero and is continuously updated at execution to represent the agent's current state and position. Requirement constraints between the origin and the finish node of a task i's delivery action represent the time window within which the full task must be completed (shown as dotted edges in Figure 1): $ft_{di} - O \in [ed_i, ld_i]$.

Alternative schedule representations can be constructed if needed. For instance, Figure 1 (left) shows an example of a two-task STNU where the travel times are deterministic. The travel actions can be collapsed into requirement constraints between the pickup and delivery actions, e.g., $sd_i - sp_i \in [TT_{di}, \infty]$, since they are no longer governed by random variables. Generally, it is always desirable to minimize the number of nodes used to represent a schedule and perform similar compacting operations when the opportunity arises, since the runtime of the algorithms to be run on the STNU increases with the size of the network. In our experiments, we consider the more general (and demanding) case described above in which all actions have non-deterministic duration.

Algorithm 1 shows the procedure an agent executes to determine a bid for an auctioned task T_a. For each valid position in its STNU, the agent adds task T_a by inserting all task nodes and determining the associated contingent and non-contingent constraints. Note that a task can only be inserted directly after the origin node or the last node of another task. The number of possible insertion points is therefore equal to $m+1$ where m is the number of tasks in the STNU. An insertion point is *valid* if the earliest delivery time of the new task is no larger than the latest delivery time of the task that would follow it. We deem that it may be desirable for an agent to bid on and commit to a task even though successful dispatch of the resulting schedule cannot be guaranteed, provided that the probability of successful dispatch is high enough to match the agent's risk profile. The agent therefore computes the *risk* associated to the new schedule, which is defined as one minus the STNU's degree of dynamic controllability (DDC), which is discussed in detail in Section II.C. This risk represents the probability that the realization of the contingent edges will be such that there exists no dynamic strategy the agent can adopt to ensure successful dispatch and execution of the plan. If the risk is below the

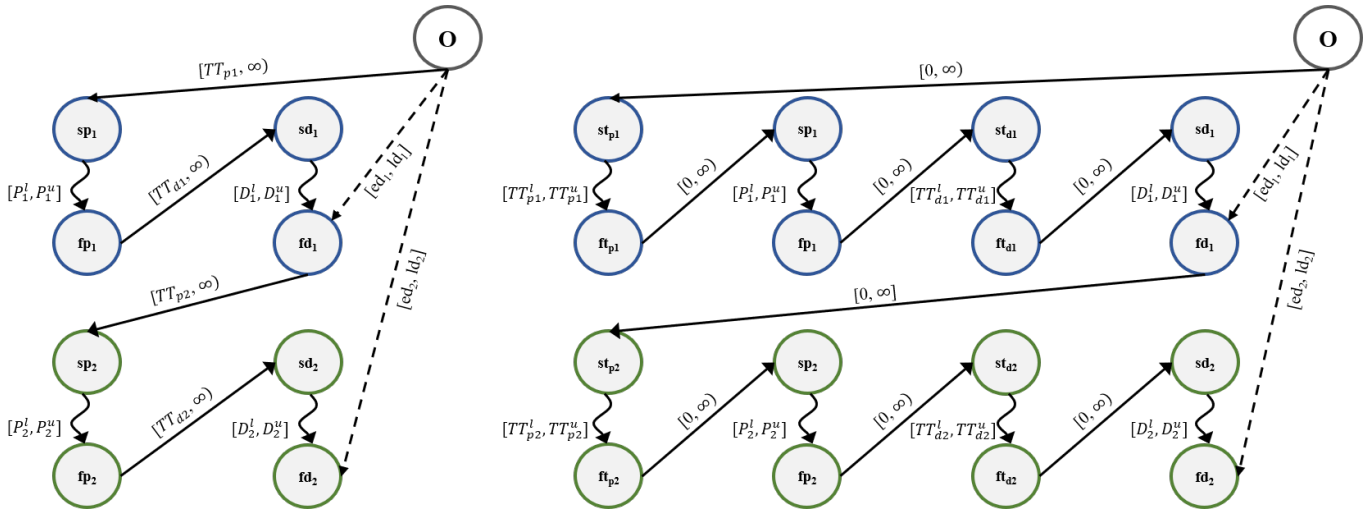


Figure 1. Two-task representation of an STNU containing two pickup and delivery tasks with (right) uncertain pickup and delivery action durations and uncertain travel times, and (left) uncertain pickup and delivery action durations and deterministic travel times. Nodes with blue edges represent the first task in the schedule, whereas nodes with green edges represent the second task in the schedule.

Algorithm 1 pTeSSI Task Scheduling and Bidding

Input
  T_a: Task on auction
  S: STNU form of agent schedule with m tasks
  ρ: Weight of risk in bidding rule
  R: Agent risk threshold
Output
  i^*: Optimal insertion point of T_a in S
  bid*: Agent's best bid for T_a
procedure COMPUTEBID(T_a, S, ρ, R)
  i^* = -1
  for valid i in [1,...,m+1] **do**
    insert T_a in position i of S
    risk = 1 - computeDDC(S)
    if risk $\leq R$ **then**
      bid = computeObjective(S, risk)
      if bid is smallest so far **then**
        $i^* = i$
        bid* = bid
      reset S eliminating T_a
  if i^* = -1 **then**
    return -1, M where M is a large number
  else
    return i^*, bid*

agent's threshold R, then the agent determines its bid for insertion point i. If the bid is the best (smallest) so far, it is saved along with the insertion point. Every time a new insertion point is tried, the STNU is reset to include only the original tasks and not T_a. After having explored all insertion points in its schedule, the agent submits the lowest bid that arises from the best insertion point i^*. If the agent is then awarded the task, it directly inserts it into its schedule in position i^* of its STNU.

C. Degree of dynamic controllability

The notion of degree of dynamic controllability (DDC) is central to the pTeSSI auction, as it allows us to calculate the risk that an agent will not be able to successfully dispatch its STNU. The DDC is a measure of *how* dynamically controllable a network is; that is, the probability that the realization of the contingent edges will be such that the network is dynamically controllable.

To compute the DDC, we first check the dynamic controllability of the network by searching for *conflicts*, constraints that cannot be satisfied simultaneously by any execution strategy. A temporal network is *dynamically controllable* iff it contains no conflicts. Formally, conflicts can be defined as sets of edges which form a semi-reducible negative cycle in the labeled distance graph of an STNU [17], and can be identified in polynomial time [18]. We use the most recent approach to verifying DC by locating conflicts in STNUs, `DC-Check` proposed by Bhargava et al. [3], which runs in O(n^3) time. It attempts to verify dynamic controllability by efficiently walking the network in reverse to demonstrate that all possible walks along a semi-reducible path originating from a negative weight edge will eventually accumulate

a non-negative weight (i.e. there are no conflicts). Otherwise, the network is not dynamically controllable and the algorithm returns the edges along each walk that represented a conflict.

Knowledge of conflicts is often leveraged to determine how to minimally relax the network's constraints in such a way as to make it dynamically controllable [8]. In our case, however, temporal constraints are hard and cannot be relaxed. Rather, we are interested in determining the amount by which the *contingent* intervals should be shrunk in order to eliminate the conflicts and render the network dynamically controllable. The probability that the realization of the original contingent edges will fall within the shrunk intervals is the probability that the network will be dynamically controllable. `ODC-Relax` is a solution to this version of the relaxation problem which is provably optimal only for networks with a single conflict, but demonstrated experimentally to be effective also in the presence of multiple conflicts [1], which we now describe.

Suppose we identify a conflict consisting of $C_1, C_2, ..., C_n$ contingent intervals of lengths $l_1 \leq l_2 \leq ... \leq l_n$, and the total amount by which the intervals must be shrunk to render the network dynamically controllable is equal to k. The objective is to resolve the conflict by shrinking each interval C_j to a new length l'_j while maximizing the realization space of the new network. This corresponds to the following optimization problem:

$$\begin{aligned} \max \quad & \prod_{j=1}^{n} l'_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} l'_j \leq \left(\sum_{j=1}^{n} l_j \right) - k \\ & 0 \leq l'_j \leq l_j \quad \forall j \end{aligned} \tag{7}$$

The above maximization problem can be solved analytically. Let q be the smallest index such that:

$$\sum_{j=1}^{n} l'_j \leq l_1 + l_2 + ... + l_{q-1} + (p - q + 1) l_q \tag{8}$$

Then:

$$l'_j = \begin{cases} l_j & \text{, for } j < q \\ \frac{\left(\sum_{j=1}^{n} l'_j\right) - l'_1 - ... - l'_{q-1}}{p-q+1} & \text{, for } j \geq q \end{cases} \tag{9}$$

Represents a solution to the maximization problem. The intervals of length larger than or equal to l_q are all shrunk to the same value, whereas all smaller intervals are left unvaried. We now need to determine the probability that the realization of the original contingent edges will fall within the shrunk intervals. Let the random variable R_j represent the difference between the realization of contingent edge C_j and its lower bound. Then the conflict will be circumvented, i.e. there will still be a dynamic strategy available to the agent to dispatch the network successfully, if the realizations are such that:

$$\sum_{j=1}^{n} R_j \leq \sum_{j=1}^{n} l'_j \tag{10}$$

The probability that the above holds is determined by applying the Central Limit Theorem (CLT). Since $R_j \sim U(0, l_j)$ and are independent, it follows from the CLT that $\sum_{j=1}^{n} R_j \sim \mathcal{N}(\mu, \sigma^2)$ where $\mu = (l_1 + ... + l_n)/2$ and $\sigma_i^2 = (l_1^2 + ... + l_n^2)/12$. If there are multiple conflicts, the relaxations are performed separately and the probabilities that the conflicts will not occur at execution are then multiplied.

D. Extension to non-uniform edge distributions

Extending the notion of degree of dynamic controllability to PSTNs with arbitrary distributions along contingent edges presents additional challenges. Unlike in the STNU case, density functions may be unbounded, in which case it is inherently impossible to protect a plan's execution against all uncertainty. The traditional notion of dynamic controllability provides little insight in this context, as networks with unbounded edges are *necessarily* uncontrollable. To extend the concept of DDC to such networks, we follow the approach proposed by Akmal et al. [2]. First, we map all unbounded contingent edges of the PSTN into bounded STNU edges. Given a risk budget α and a set of i unbounded edges X_i with density f_i, we determine the intervals Y_i such that $P(X_i \in Y_i) = 1 - \alpha$ by truncating $\frac{\alpha}{2}$ of mass from each tail of f_i. We then check the resulting STNU, which we denote S_α, for dynamic controllability (DC). If it is not DC, all conflicts are extracted and the ODC-Relax algorithm described in Section II.C is applied to determine the probability p_α of successful dynamic execution of S_α. We then need to account for the risk α we ignored on each unbounded edge by truncating the original distributions. The degree of dynamic controllability of our PSTN is therefore $(1 - \alpha)^n \cdot p_\alpha$, where n is the number of edges with an unbounded distribution.

The value of α controls how much realization volume we sacrifice upfront. Lower values of α give `ODC-Relax` more freedom to choose where to sacrifice relaxation volumes, but increase the required number of relaxations of the contingent edges and their magnitudes. While setting α too high may lead us to sacrifice too much realization volume upfront and inadvertently make highly suboptimal relaxations, setting α too low and capturing an unreasonably large amount of the distribution may also hinder performance as the choices made by `ODC-Relax` are agnostic to the underlying distribution. In general, the best value of α may depend on the

underlying network and the distribution its edges follow. Earlier methods to generate an STNU-based approximation of a PSTN - such as DREA [16] - incorporate an outer search over possible values of α, which greatly increases the computational burden. Recent work demonstrates that this can be circumvented by tuning the value of α a priori [2]. In Section V, we zoom in on the effect of α on system performance when the contingent edges are normally distributed.

E. Analysis of the pTeSSI auction

Let us consider an auction with m tasks and n agents. In the first round, all agents need generate bids for all tasks, producing $m \cdot n$ bids. In the remaining rounds only a single agent, namely the winner of the previous round, will need to bid on all unallocated tasks. Since in each round exactly one task is allocated or eliminated from the set on auction, in round $i \neq 0$ one agent will need to generate $m - i$ bids. The total number of bids generated across all rounds of the auction is therefore:

$$mn + \sum_{i=1}^{m} (m - i) = mn + \frac{m^2 - m}{2} = \mathcal{O}(mn + m^2) \tag{11}$$

In order to produce a bid for a task, an agent needs to compute a hypothetical bid for each possible insertion point in its temporal network. The number of possible insertion points is equal to the number of tasks in the agent's schedule. For each insertion point, the agent needs to run `DC-Check` to determine whether the network is DC and identify conflicts, then `ODC-Relax` to determine how to shrink the contingent edges in such a way as to render the network dynamically controllable, and finally determine its bid based on the makespan (`MAXT` bidding rule) or increase in distance travelled (`SUMDIST` bidding rule) in the shrunk network. `DC-Check` runs in $\mathcal{O}(|S|^3)$ time, where $|S|$ is the number of nodes in the STNU [3]. Both the makespan and distance travelled calculations require walking the shrunk network once and are therefore linear in $|S|$.

`ODC-Relax`'s complexity is known to be $\mathcal{O}(c \log c)$, where c is the number of conflicts that need to be resolved [2]. While it is difficult to predict the exact number of conflicts that will be identified at run-time, we can determine an upper bound on c for networks of the structure described in Section II.B. Let us call a network with such structure $\hat{S}$ and its distance graph $\hat{D}$. Since a conflict of $\hat{S}$ is defined as a semi-reducible negative cycle in $\hat{D}$, there can exist no more conflicts in $\hat{S}$ than there are non-trivial cycles (more than 2 nodes) in $\hat{D}$. Recall that the structure of $\hat{S}$ is such that there exists an edge from node n_i to node n_j only if $j = i+1$, or $i = 0$ and $j = Nk$ where N is the number of nodes per task and $k \in \mathcal{Z}^+$. Let us denote all edges stemming from the origin node as *critical* edges, then we can state the following properties about the non-trivial cycles of $\hat{D}$.

Lemma 1: Each non-trivial cycle of $\hat{D}$ contains exactly 2 critical edges.

Proof sketch: Let C be any non-trivial cycle in $\hat{D}$ starting at node n_s. First, we claim that C must contain at least one critical edge. For $s = 0$, this follows trivially from the definition of a critical edge. If $s \neq 0$ and C contains no critical edges, then there must exist some edge from a node $n_{s\pm k}$ to n_s with $k > 2$ and $s \neq 0$, which contradicts the structure of $\hat{S}$. If C contains one critical edge, then it must contain at least two critical edges. C cannot contain more than two critical edges without the origin node being visited at least twice. Hence, C contains exactly two critical edges.

Lemma 2: Any two critical edges of $\hat{D}$ are part of a unique non-trivial cycle.

Proof sketch: Take any two critical edges c_i and c_j leading to nodes n_i and n_j respectively. Since there exists an edge from n_{k-1} to $n_k \forall k \neq 0$, there must be a simple path from n_i to n_j. Hence c_i and c_j are part of a non-trivial cycle C_{ij}. Now take a different pair of critical edges c_p and c_q, which are part of a cycle C_{pq}. It follows from Lemma 1 that $C_{ij} \neq C_{pq}$.

Using Lemmas 1 and 2, the number of cycles in $\hat{D}$ is the number of distinct pairs of critical edges. Since there are $K + 1$ critical edges, where K is the number of tasks in $\hat{S}$:

$$\text{Number of cycles in } \hat{D} = \binom{K+1}{2} = \frac{K^2 + K}{2} \tag{12}$$

$\mathcal{O}(|S|) = \mathcal{O}(K)$, so the worst-case complexity of `ODC-Relax` is $\mathcal{O}(|S|^2 \log(|S|^2))$. Computing a bid for a single insertion point is therefore $\mathcal{O}(|S|^3)$ (due to `DC-Check`) and there are K insertion points. It follows that the entire task scheduling and bidding algorithm is $\mathcal{O}(|S|^4)$. At each round, winner determination is linear in the number of agents since each agent only shares its best bid. If the agents start with empty schedules, we can expect the tasks to be approximately evenly distributed among the agents and the entire auction to have an overall complexity of $\mathcal{O}(\frac{m^6}{n^4})$. This is the same complexity as the regular TeSSI auction [20], despite that the fact that we are solving a substantially more complex scheduling problem in the bidding phase.

pTeSSI always terminates, but does not guarantee optimality as it cannot capture all possible inter-task synergies due to the sequential nature of the auction. For the special case in which there is only one conflict in the agent's STNU, however, `ODC-Relax` is provably optimal and pTeSSI is a maximum factor of 2 and $2n$ away from the optimal solution for the `SUMDIST` and `MAXT` objectives respectively. This follows directly from the general results on (non-temporal) sequential single item

auctions, a formal proof of which can be found in [14], since in their bidding routine agents use a temporal form of the cheapest insertion heuristic to solve their local TSP. In Section V we show that in practice, despite the weak optimality bounds, pTeSSI provides a measurable improvement over the (non-probabilistic) TeSSI auction for both bidding rules.

III. Re-auctioning and bundling

The main weakness of pTeSSI, especially when implemented in an online setting with sets of tasks dynamically entering the problem, is that the amount of inter-task synergies it is able to consider is limited. We are able to capture synergies between the task set on auction and the set of tasks already allocated to a *single* agent, but no combinations of these sets since existing allocations are never revisited. To remedy this, we propose an extension to pTeSSI in which agents dynamically submit subsets of tasks in their schedule to the auctioneer. Every time a new set of tasks enters the system, all agents have the opportunity to temporarily de-commit from tasks in their schedule and send them back to the auctioneer to be re-auctioned together with the newly entered tasks. While this increases the number of inter-task synergies we can consider, it also increases the duration of the auction as more tasks need to be allocated at once. To mitigate this downside and accelerate the re-auctioning phase, we propose a mechanism to generate bundles of tasks and auction them together, exploiting the temporal relationship between tasks in the bundles to further speed up the bidding phase. For the sake of clarity in our explanation and analysis, we denote the version of pTeSSI that runs on bundles rather than single tasks as pTeSSB (Sequential Single Bundle).

A. Task selection

Agents need to determine locally which tasks to submit for re-auctioning. While there are several ways to approach this problem, we propose a mechanism which does not require information about other agents' plans or information exchange with the auctioneer. Given the hill-climbing nature of sequential auctions like pTeSSI, it is important to re-auction only those tasks which have the potential of generating a large enough improvement in the system objective. We want to avoid re-auctioning an excessive amount of tasks and scratching good initial partial allocations.

To decide which tasks to re-auction, the agents execute a procedure which can be interpreted as the reverse of the auction mechanism outlined in Algorithm 1. Agents generate bids proportional to the reduction in the system objective that the removal, rather than addition, of a task would generate. They then re-submit the task to the auctioneer if their bid exceeds a threshold O. Carry over the notation from Section II.A, but this time let $\mathcal{P}'$ be the allocation resulting from agent A_i *removing* task T from its schedule, so:

$$P'_j = \begin{cases} P_j \backslash \{T\} & \text{, for } j = i \\ P_j & \text{, for } j \neq i \end{cases} \tag{13}$$

Then we are again interested in minimizing $\mathcal{F}(P') - \mathcal{F}(P)$, so we can apply the same bidding rules we derived in Section II.A. The only exception is that for the `MAXT` bidding rule we cannot exclude $\mathcal{F}(P)$ from the bid, so the agent bids $\mathcal{M}_i(P'_i) + \rho \cdot \mathcal{R}_i(P'_i) - (\mathcal{M}_i(P_i) + \rho \cdot \mathcal{R}_i(P_i))$.

Algorithm 2 shows the procedure each agent goes through to select tasks for re-auction. For each task in its schedule, the agent computes the bid associated to its removal. The task T_{i^*} with the lowest bid is removed, and we perform the same operation for the remaining tasks in the schedule. We continue greedily until $|bid|^* > O$, at which point all removed tasks are submitted to the auctioneer. Note that we take the absolute value of the bids and use a threshold $O > 0$ to enhance interpretability. All bids are ≤ 0 since the removal of a task cannot increase the system objective.

Algorithm 2 Selection of tasks to be re-auctioned

```
Input
    S:  STNU representation of agent schedule
    ρ:  Weight of risk in bidding rule
    O:  Re-auction threshold
Output
    A:  Tasks to be re-auctioned
procedure SELECTTASKS(S, ρ, O)
    bid* = 0
    while |bid*| ≤ O do
        for i in [1,...,|S|+1] do
            remove T_i from S
            risk = 1 - computeDDC(S)
            bid = computeObjective(S, risk)
            if bid is smallest so far then
                bid* = bid
                i* = i
            reset S by adding back T_i
        remove T_i* from S and append to A
```

B. Bundling

The re-auctioning procedure increases the amount of tasks to be allocated at once, thereby increasing the computational overhead and auction rounds required. Generating bundles of tasks and auctioning these together has been investigated in the auction literature as a way to mitigate this effect while retaining the solution quality improvement generated from broadening the range of

inter-task synergies explored [13]. Heap and Pagnucco propose a two-stage clustering mechanism in which tasks are grouped first based on their pickup and then their delivery location [11]. This approach, however, fails to consider the alignment of task time windows within the bundle and does not account for the sequential nature of pickup and delivery tasks. We expect small travel times between the pickup and delivery locations of sequential tasks, as well as compatibility of their time windows, to be key in the formation of low-cost bundles. Other attempts have been made in the vehicle routing literature [15] [10]. While these approaches are based on general notions of the spatial and temporal compatibility of tasks, none have, to the best of our knowledge, attempted to explicitly consider how these tasks will be scheduled and the likelihood that agents will be able to execute them successfully.

We propose a hierarchical agglomerative mechanism that bundles tasks which can be executed in sequence with controlled risk, by leveraging domain knowledge and the spatio-temporal relationship between tasks. Consider two tasks T_i and T_j with latest delivery time ld_i and ld_j respectively. Let us for now assume that task T_j will be executed after T_i. Since we have no knowledge of the schedule of the agent that will be bidding on the task pair, we determine the probability that T_j can be successfully executed before its deadline given the worst-case scenario that T_i is executed exactly at its latest delivery time ld_i. Let E_{ij} represent the event that the above holds:

$$\begin{aligned}\mathcal{P}(E_{ij}) &= \mathcal{P}(fd_j \leq ld_j | fd_i = ld_i) \\ &= \mathcal{P}(TT_{p,j} + P_j + TT_{d,j} + D_j \leq ld_j - ld_i)\end{aligned} \tag{14}$$

Then we consider the task *ordering* $\langle i, j\rangle$ to be valid if:

$$\mathcal{P}(E_{ij}) \geq 1 - R \tag{15}$$

where R is the agent's risk threshold. The possible orderings to be checked for validity depend on the relationship between the time windows of T_i and T_j. If $[ed_i, ld_i] \cap [ed_j, ld_j] = \emptyset$, then the only possible ordering is $\langle i, j\rangle$ if $ld_j > ld_i$ and $\langle j, i\rangle$ if $ld_i > ld_j$. If instead the intervals $[ed_i, ld_i]$ and $[ed_j, ld_j]$ overlap, both $\langle i, j\rangle$ and $\langle j, i\rangle$ should be checked as outlined above.

Once we know whether two tasks can be executed in sequence with acceptable risk, we can express the dissimilarity between them by means of a distance function, not to be confused with the physical distance between task locations. Let $\Delta T_{ij} = ld_j - ed_i$ and let TT_{ij} represent the travel time from the delivery location of task T_i to the pickup location of task T_j. Then we define the distance between tasks T_i and T_j as follows:

$$\textbf{dist}(T_i, T_j) = \begin{cases} TT_{ij} + \Delta T_{ij} & , \textbf{ord} = [\langle i, j\rangle] \\ TT_{ji} + \Delta T_{ji} & , \textbf{ord} = [\langle j, i\rangle] \\ \min\left(TT_{ij} + \Delta T_{ij}, \right. & \\ \quad \left. TT_{ji} + \Delta T_{ji}\right) & , \textbf{ord} = [\langle i, j\rangle, \langle j, i\rangle] \\ M & , \textbf{ord} = \emptyset \end{cases} \tag{16}$$

where M is a large number and **ord** is a list containing the *valid* orderings of T_i and T_j. The above distance function considers both the proximity of sequential pickup and delivery locations, captured by the travel time component, and the alignment of the time windows, captured by the ΔT factor. Since we guarantee a priori that successful execution of the first task will lead to successful execution of the second with at least $1 - R$ probability, we can directly penalize time window misalignment without having to worry about the time windows being too close. We tighten the bundles as much as possible while keeping the execution risk bounded.

We adopt the complete linkage criterion, only merging bundles if the largest distance across them is below a distance threshold D. This choice of linkage criterion makes the size of the generated bundles tractable and free of large imbalances, which is desirable as agents have limited and equal battery life. Bundles are ordered following the best valid ordering determined while computing the distance via Equation (16).

C. Bidding phase

In pTeSSB, agents bid on bundles rather than individual tasks. Algorithm 3 shows the procedure an agent executes to evaluate its bid on a bundle B_a containing an ordered list of b tasks. The algorithm is similar to the single-task case elaborated in Algorithm 1, but we take advantage of the fact that tasks in the bundle are ordered to further decrease the number of insertion operations required and increase the efficiency of the bidding phase.

For every task in B_a, the agent attempts to insert it into each possible position j of S and compute its bid as described in Section II.A. Once all insertion operations have been performed, task $B_a[i]$ is added in the best position j^* of S. Since the bundles are ordered, we know that $B_a[i+1]$ should be scheduled after $B_a[i]$, so before moving to task $i+1$ we set $j = j^* + 1$. While it does not change the worst-case ($j^* = 1$), this adjustment can substantially reduce the number of insertions if the first task is inserted late in the schedule. The best bid is then the value of the objective when all tasks in B_a have been added to S, normalized by the number of tasks b in the bundle. In order to be allocated B_a, the agent needs to be able to generate a valid bid for *every* task in B_a, so as soon as the agent finds a task without a valid insertion point, it stops evaluating the bundle and bids a large

Algorithm 3 pTeSSB Task Scheduling and Bidding

Input
  B_a: Bundle on auction with b tasks
  S: STNU form of agent schedule with m tasks
  ρ: Weight of risk in bidding rule
  R: Agent risk threshold
Output
  i^*: List of insertion points of tasks in B_a in S
  bid*: Agent's best bid for B_a
procedure COMPUTEBID(B_a, S, ρ, R)
  i^* = [-1]*b
  $j = 1$
  $j^* = -1$
  while i in [1,...,b] **do**
    while j in [1,...,m+1] **do**
      insert $B_a[i]$ in position j of S
      risk = 1 - computeDDC(S)
      if risk $\leq R$ **then**
        bid = computeObjective(S)
        if bid is smallest so far **then**
          $j^* = j$
          bid* = bid
        reset S eliminating $B_a[:i]$
        $j = j + 1$
    if $j^* = -1$ **then**
      reset S eliminating $B_a[i]$
      return [-1]*b, M where M is very large
    else
      insert $B_a[i]$ in position j^* of S
      $i^*[i] = j^*$
      $j = j^* + 1$
    $i = i + 1$
  reset S eliminating B_a
  return i^*, $\frac{\text{bid}^*}{b}$

number. Since the bundles are effectively partial schedules with bounded risk conditional on the successful execution of their *first* task, we expect a larger number of agents to be unable to insert the first task than subsequent tasks in the bundle. This should contribute to making the bidding phase faster than in an equivalent pTeSSI auction where each task in the bundle is auctioned independently.

D. Analysis of the pTeSSB auction

Bundling decreases the number of bids to be generated, as we are effectively running an auction over m/b rather than m items where m is the number of tasks and b is the average bundle size. The number of bids generated is therefore $\mathcal{O}(\frac{mn}{b} + (\frac{m}{b})^2)$. This effect, however, is offset by the increased number of operations to be performed in the bidding phase. Determining a bid for a bundle rather than a task increases the time complexity by a factor b to $\mathcal{O}(b|S|^4)$ in the worst-case where each agent is able to fully insert every bundle on auction at the beginning of its schedule. Auctioning bundles rather than individual tasks therefore does not change the overall worst-case complexity, which is still $\mathcal{O}(\frac{mn}{b} \cdot b|S|^4) = \mathcal{O}(mn|S|^4)$ and $\approx \mathcal{O}(\frac{m^6}{n^4})$ if the agents start with empty schedules. However, the modifications made to the bidding procedure described in Section III.C make pTeSSB faster than pTeSSI in practice, as we demonstrate experimentally in Section V.B.

IV. Experimental Setup

Given our overarching objective of enhancing the applicability of temporal auctions to real-world systems where deterministic task durations cannot be guaranteed, we deem it essential to select a relevant real-world application in our experiments. We evaluate the effectiveness of our auction algorithm in allocating pickup and delivery tasks with uncertainty through several online simulations of an on-demand UAV food delivery operation in the Amsterdam area. While this is still somewhat of a futuristic application at the time of writing, it has received remarkable attention from academia and industry alike and several companies have begun deploying prototype systems in recent years [1]. Decentralized planning approaches are seen as the most suitable for problems involving large teams of UAVs [23] and recently proposed architectures for unmanned aircraft traffic management suggest a high degree of decentralization [9] as a means to handle the large number of actors and high degree of dynamism foreseen in the low-altitude airspace. Delivery UAVs will have to deal with a number of sources of uncertainty, such as airspace congestion, interactions with customers and possible delays, and the time-sensitive nature of food orders makes respecting the customers' time windows of utmost importance. This setting lends itself extremely well to the evaluation of our auction algorithm. Our choice of Amsterdam is due to the high availability of data regarding locations of restaurants and residential buildings, and the large amount of drone as well as food delivery companies active in the area.

A. Operational setting and assumptions

All simulations are set up using the Mesa agent-based modeling and programming framework in Python and include three main types of agents - namely the customers, the UAVs and the operator. Customers submit order requests to the operator including a delivery window, pickup location and delivery location. The operator

[1]The FAA has also recently approved the first fully autonomous commercial drone flight [22]

acts as the auctioneer and allocates the delivery tasks to the UAVs, each of which is responsible for the planning and timely execution of all tasks in its schedule.

We consider a 6 km squared area encompassing the center of Amsterdam - where the majority of restaurants are located - and main residential areas in the Zuid and Oost districts. We use open-source data from OpenStreetMap to obtain the locations of restaurants and residential buildings in the area as shown in Figure 2. Restaurant locations are sampled from the underlying distribution ahead of each simulation. They represent the possible locations from which customers can choose their order to be delivered. Customers spawn at their homes, which correspond to the delivery locations of their order requests, and are locations sampled from the distribution of residential buildings.

Figure 2. Illustration of the 6km x 6km area of Amsterdam considered in the simulations. Restaurant locations (green) and residential buildings (yellow) are taken from OpenStreetMap.

We aim to use the simulation setup as a way to assess the suitability of the proposed auction algorithm as a coordination mechanism for pickup and delivery problems of this nature. Performing detailed modeling of all aspects of the UAV delivery operation would lead us far astray from the goals of this paper [2]. Rather, we make a series of high-level operational assumptions that allow us to direct our attention to factors that are of interest to the task allocation problem.

We represent the environment depicted in Figure 2 as a 60 x 60 grid. The UAVs have limited battery life and must schedule stops to have their batteries swapped before they run out. All battery swaps occur at the operator's hub, which is located in the center grid-cell of the map. We assume there will always be an available battery for a UAV that requires it, which is reasonable given that batteries are relatively inexpensive and can be charged at high rates with modern technology.

We assume UAVs are able to cruise at a speed of 70 km/h, which is well within the capabilities of modern quadcopters, and that their battery life is equal to 45 minutes. While this may seem like a large number compared to commercially available quadcopters which are not optimized for range, it is in line with expectations for future delivery UAVs [4] and not unreasonable based on first principles. Following the sizing approach of D'Andrea [7], we estimate that a small UAV with a structural weight of 2 kg would require a $\approx$ 4 kg modern high-end Li-ion battery to transport a 2kg payload[3] at 70 km/h for 45 minutes in 25 km/h winds. The battery would therefore amount to $\approx$ 50%[4] of the weight of the loaded UAV which is substantial but reasonable, and interestingly enough close to the $\approx$ 45% fuel fraction modern long-haul commercial aircraft are designed for.

UAVs use an A* planner to determine the shortest path between any two locations, and travel at their cruise speed V_c along the path. We capture possible delays that the UAVs may encounter while flying their trajectory through a random variable Δ_{TT}. Let dist(x_1, x_2) be the length of the shortest path between two locations x_1 and x_2 as computed via A*, then the travel time TT is equal to:

$$TT(x_1, x_2) = \frac{\text{dist}(x_1, x_2)}{V_c} + \Delta_{TT} \qquad (17)$$

Similarly, the time taken by a UAV to execute pickup and delivery actions - as well as to complete a battery swap stop - once it has reached the appropriate location are non-deterministic and governed by random variables whose distribution is known to the agent. We make high-level estimates of reasonable bounds on these durations, outlined in Table 1, which we use as nominal in the experiments. We then vary these bounds and assess the effects on system performance in Section V.A.

RV	Description	Lower bound	Upper bound
P	Pickup time	2 min	4 min
D	Delivery time	1.5 min	3 min
B	Battery swap time	1.5 min	3.5 min
Δ_{TT}	Travel time delay	10 %	30 %

Table 1. Nominal bounds of non-deterministic action durations. Δ_{TT} is in percentage of the travel time TT.

[2]For detailed studies on UAV operations and field tests of related technologies and systems, the reader is referred to the work from NASA's UTM project [6] [5], which represents a good overview of the current state of the art.

[3]For reference, this would roughly correspond to an order consisting of 4 regular-sized pizzas and a bottle of wine.

[4]We can expect this figure to drop substantially over the coming years due to fast-paced improvements in battery power and energy densities, driven in particular by competition in the electric car market. Battery energy densities have already tripled since 2010 and prices have fallen by 87% over the same period.

Let us now briefly explain how we arrive at the values outlined in Table 1. While there are a variety of ways to collect and deliver items via UAV, we consider the approach patented by FAA-approved drone delivery scale-up Flirtey to be the most suitable for urban areas as it does not require the UAV to land. To execute a pickup or delivery action, the UAV needs to descend from its cruise altitude to a hover altitude, lower a container via a tether, wait for the items to be collected or loaded, re-collect the container and then climb back to cruise altitude. The FAA's latest concept of operations for UTM [9] suggests that UAVs will be able to cruise at $\approx$ 120 m (400 ft) above ground level (AGL). While there is no regulatory consensus on descent and climb rates that will be regarded as safe in the low altitude airspace, we consider 5 m/s reasonable for a small UAV. We also expect that the tether will be operated at a moderate speed of $\approx$ 1 m/s, to limit second and third party risk and avoid spilling or damaging items in the container. Based on Flirtey's patent, the hover altitude to be decided upon descent should be high enough to clear i.a. power lines and trees. For our urban environment we expect this to fall between 10 and 30 m AGL. The bounds on P in Table 1 account for an additional 1 to 2 minutes for the correct order to be brought to the container and loaded, and those on D for 0.5 to 1 minutes for the order to be collected. For the battery swaps we assume the UAV will need to land at ground level and the swap will take between 0.5 and 2.5 minutes.

B. Online dispatch and execution

Agents dispatch their plans online by following the early-first strategy proposed in [19] on the dynamically controllable STNUs generated by `ODC-Relax`, which is guaranteed to succeed on dynamically controllable plans. The realization of a random variable can only be observed by the agent once it has executed the associated action. For instance, taking over the notation from Section II.B, the UAV will only know the value P_i will take once it has reached the pickup location of task i, executed time point sp_i and Nature has executed time point fp_i. Every time the agent collects information about the realization of an action duration, it updates its schedule and recalculates the DDC. If the DDC drops below $1-R$, the agent performs an online adjustment through which it attempts to increase the DDC back to a suitable level with as few changes to its plan as possible. The agent identifies the task whose contingent edges must be shrunk the most to solve conflicts via `ODC-Relax`, which is the task that will generate the sharpest increase in DDC when removed from the schedule, and sends it back to the operator for re-auctioning. If DDC is still less than $1-R$, then the agent greedily repeats this procedure until DDC $\geq 1-R$.

C. Experiments

We run experiments with a fixed fleet size of 10 UAVs and varying numbers of customer orders, which we refer to as *demand levels* throughout the analysis. All experiments last two hours, reflecting the typical peak hours between 18.00 and 20.00 that food delivery providers tend to experience in Amsterdam[5]. Orders enter the system in a rolling fashion, in equally sized batches every 10 minutes. Customers' delivery time windows are 10 minutes wide and start X minutes from the time at which they place their order, where $X \sim U(20, 40)$. For each combination of parameters presented, the results are averaged over 40 simulation runs, which is consistently found to be enough to stabilize the coefficient of variation. The main system-level parameters we measure and discuss are the total number of deliveries successfully executed by the system and the average auction duration. We also address the distance traveled per successful delivery to aid in drawing a comparison across the two bidding rules. We categorize our experiments into the three distinct sets labeled A, B and C which are explained below. Each set of experiments is aimed at testing different effects and main hypotheses. In Section V.A we discuss the results of set A, in Section V.B those of set B and in Section V.C those of set C.

1. Set A

In set A, tasks are allocated online via pTeSSI and the agents are not allowed to perform dynamic re-auctioning. We run experiments with three different demand levels, namely 60, 120 and 180 orders.

In section Section V.A.1 we focus on the case in which the contingent edges are uniformly distributed. We vary the risk threshold and the weight of the risk in the bidding rule to understand their effect on performance with both the `SUM-DIST` and `MAX-T` bidding rules, and we vary the width of the contingent intervals in Table 1 to examine the impact of different degrees of uncertainty. We also draw a comparison to the results obtained when applying the non-probabilistic TeSSI auction. We test a number of effects with the aim of understanding the differences between pTeSSI and TeSSI as well as those between the `SUM-DIST` and `MAX-T` bidding rules. The main question we seek to answer via this comparison is whether pTeSSI is a more effective task allocation mechanism than TeSSI for the problem at hand without sacrificing efficiency, which we capture with the following hypotheses.

- **H_{A1}**: pTeSSI leads to a larger number of successful deliveries than TeSSI.

[5] see for instance https://www.uber.com/en-NL/blog/ubereats-netherlands-guaranteed-earnings/

- $\mathbf{H_{A2}}$: The average duration of the pTeSSI auction is no larger than that of the TeSSI auction.

We run additional experiments in which the contingent edges are normally distributed and discuss these in Section V.A.2. These aim to evaluate the effectiveness of the extension to unbounded contingent edge distributions described in Section II.D under different levels of the approximation factor α.

2. Set B

In set B, we allow the agents to dynamically re-auction tasks according to the mechanism described in Section III.A. We vary the threshold O in Algorithm 2 to investigate how different levels of re-auctioning affect the number of deliveries that the system is able to execute. Our expectation is that there will be some value of O for which the re-auctioning mechanism consistently improves partial allocations over time and thereby increases the number of deliveries the system can execute. We also expect, however, to observe significant increases in auction durations due to more tasks being auctioned at once. Let the extension "-re" indicate the case in which dynamic re-auctioning occurs, then we hypothesize that:

- $\mathbf{H_{B1}}$: pTeSSI-re leads to a larger number of total deliveries than pTeSSI.
- $\mathbf{H_{B2}}$: pTeSSI-re leads to a higher average auction duration than pTeSSI.

3. Set C

In set C, agents perform dynamic re-auctioning and we attempt to accelerate the auction via bundling in each of the three demand cases. We vary the distance threshold of in the bundling mechanism to examine its effect on the number of successful deliveries and the auction duration. We also aim to establish whether combining re-auctioning and bundling is effective in retaining a performance advantage over the case without re-auctioning while mitigating the associated increase in auction duration. This translates to the following two hypotheses:

- $\mathbf{H_{C1}}$: pTeSSB-re leads to a larger number of total deliveries than pTeSSI.
- $\mathbf{H_{C2}}$: pTeSSB-re leads to a lower average auction duration than pTeSSI-re.

We can view the bundling mechanism as effective if *both* $\mathbf{H_{C1}}$ and $\mathbf{H_{C2}}$ hold.

V. Results and discussion

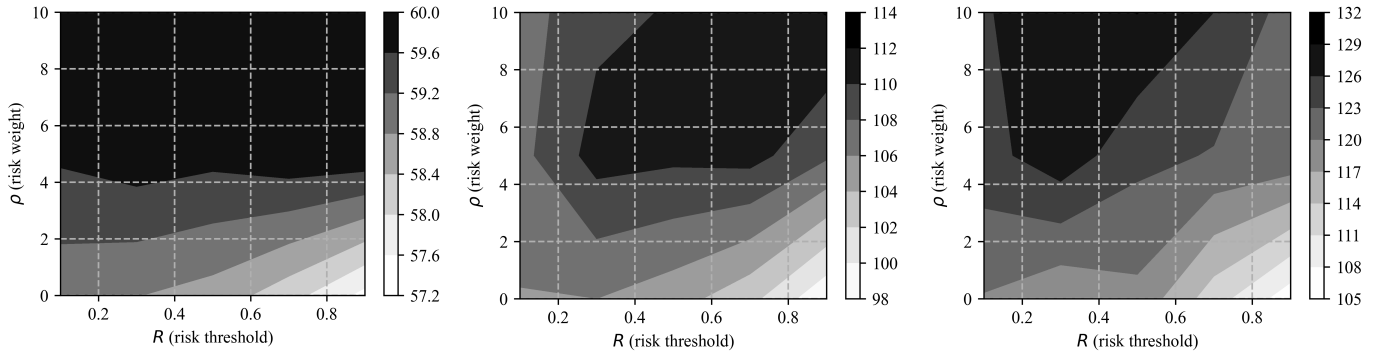
A. Experiment set A - pTeSSI

We begin by presenting the main results from the experiment set A, discussing the case with uniform contingent edges first and then the extension to normally distributed contingents.

1. Uniform contingent edges

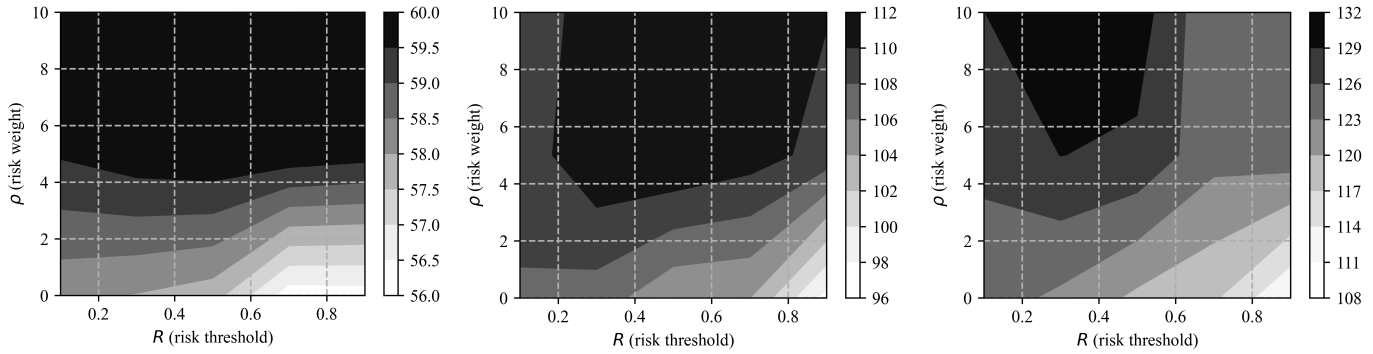
Figure 3 and Figure 4 show the total deliveries generated by the system in each of the three demand cases with varying levels of the risk threshold (R) and weight of the risk in the agent's bids (ρ), for the `MAX-T` and `SUM-DIST` bidding rule respectively. R is varied between 0.1 and 0.9 and ρ between 0 and 10. Both parameters have a pronounced effect on performance which is consistent across both bidding rules. For low values of ρ ($\leq$ 2), taking on more risk has a significant negative impact on performance across all demand cases, decreasing the total amount of deliveries generated by up to $\approx 15\%$ in the highest demand case. To successfully leverage the execution risk, we need to penalize it heavily enough. For values of $\rho \geq 5$, the trend is different and varies depending on the demand case. In the lowest case, increasing risk has no effect on performance as agents are already able to successfully execute virtually all tasks (99.7% on average) with the lowest risk threshold of 0.1. This suggests that there are usually combinations of schedules satisfying all tasks' constraints that can be made dynamically controllable by sacrificing little realization volume, resulting in very low execution risk. The larger values of ρ prioritize these schedules. For the two higher demand cases, we observe an initial increase in performance when increasing the risk from the 0.1 threshold. Satisfying all temporal constraints is "hard" enough that it is beneficial for agents to trade-off some degree of controllability for the ability to commit to more tasks. Accepting large execution risks, however, leads to over-committing and a decline in the total deliveries executed. Many of the conflicts in the agents' temporal plans occur at execution, leading agents to either violate delivery windows or be forced to de-commit tasks in their schedule shortly before dispatch. In the higher demand cases, the strongest system performance is with a risk threshold of 0.5 and high values of ρ. This is the value we would expect given that in a schedule with a DDC < 0.5 it is more likely that the realization of all contingent edges will lead to at least one conflict at execution, than that all conflicts will be avoided and the plan will effectively be dynamically controllable. For the rest of the results presented in this section, we set $R = 0.5$ and $\rho = 10$.

In Table 2, we report the performance of pTeSSI with $R = 0.5$ and $\rho = 10$ and TeSSI for all demand cases and both bidding rules. To adapt TeSSI to work with the non-deterministic tasks present in our experiments, we follow the original proposition of Nunes [21] and formulate each STN constraint as an interval matching the bounds



(a) 60 orders **(b)** 120 orders **(c)** 180 orders

Figure 3. Contour plot of the total deliveries executed by the system with the `MAX-T` bidding rule for the three different demand cases, with varying levels of the risk threshold and the weight of the risk in the bidding rule.



(a) 60 orders **(b)** 120 orders **(c)** 180 orders

Figure 4. Contour plot of the total deliveries executed by the system with the `SUM-DIST` bidding rule for the three different demand cases, with varying levels of the risk threshold and the weight of the risk in the bidding rule.

on the realization of the underlying random variable. We then require the STN to be consistent for all values along the interval. Instead of determining the DDC and verifying whether the risk is below its threshold (lines 13-14 of Algorithm 1), the agent runs the Floyd-Warshall algorithm on the STN associated to the worst-case realization of all contingents and only bids if the latter is consistent. This is equivalent to requiring dynamic controllability on the STNU representation of the temporal plan.

We test for statistical significance using the non-parameteric Wilcoxon signed-rank test as the measurements are paired but not normallly distributed. We test against the null hypothesis that the results of pTeSSI and TeSSI originate from the same distribution for both bidding rules and report the p-value in the appropriate column. We assess the effect size through the Vargha-Delaney A-value, which is the probability that performance using pTeSSI exceeds that achieved using TeSSI based on the observed results. We perform the same analysis to assess the effect of the bidding rule on the performance of both pTeSSI and TeSSI. The A-value reported along the rows is the probability that `MAX-T` performs better than `SUM-DIST` according to the relevant parameter. Note that the effect we test for varies based on the parameter in question. Better performance translates to larger values for the total deliveries parameter, but lower auction durations and distances traveled per delivery. For comparisons leading to statistically significant ($p < 0.05$) effects of large magnitude ($A > 0.71$ or $A < 0.29$) both the p and A values are shown in bold. Note that each effect is tested on a different combination of sets of simulation results, so there is no need to perform a post hoc correction such as the Bonferroni one to control for the family-wise error rate.

The performance difference between pTeSSI and TeSSI is statistically significant and evident across all demand cases. pTeSSI leads to substantially more deliveries than TeSSI, demonstrating the advantage of intelligently leveraging risk rather than enforcing dynamic controllability at all costs, provided it is weighed heavily enough in the agents' bids ($\rho = 10$ for all results in Table 2). The gap grows with the total number of orders, as it becomes increasingly more difficult to accommodate all tasks within their constraints without sacrificing some

		Total deliveries [-]				Auction duration [s]				Distance per delivery [km]			
Demand		TeSSI	pTeSSI	p	A	TeSSI	pTeSSI	p	A	TeSSI	pTeSSI	p	A
60 orders	MAX-T	55.3	59.8	**3.1e-8**	**0.99**	1.61	0.59	**3.6e-8**	**1.0**	6.91	6.94	0.61	0.48
	SUM-DIST	50.1	59.7	**3.3e-8**	**1.0**	2.92	0.94	**3.6e-8**	**1.0**	6.21	6.05	0.10	0.61
	p	**3.3e-8**	0.6	-	-	**3.6e-8**	**3.6e-8**	-	-	**9.8e-6**	**1.1e-7**	-	-
	A	**0.97**	0.52	-	-	**1.0**	**1.0**	-	-	**0.14**	**0.08**	-	-
120 orders	MAX-T	77.8	109.9	**3.5e-8**	**1.0**	11.4	2.08	**3.6e-8**	**1.0**	6.40	6.43	0.39	0.45
	SUM-DIST	80.25	111.8	**3.5e-8**	**1.0**	17.2	3.82	**3.6e-8**	**1.0**	6.03	5.93	0.10	0.60
	p	7.8e-3	0.11	-	-	**3.6e-8**	**1.1e-7**	-	-	**7.76e-7**	**5.73e-7**	-	
	A	0.34	0.40	-	-	**1.0**	**0.94**	-	-	**0.24**	**0.11**	-	-
180 orders	MAX-T	79.3	127.7	**3.5e-8**	**1.0**	32.5	3.5	**3.6e-8**	**1.0**	5.94	5.90	0.35	0.57
	SUM-DIST	89.7	131.0	**3.5e-8**	**0.99**	44.2	5.4	**3.5e-8**	**1.0**	5.49	5.61	0.10	0.39
	p	**7.6e-8**	0.22	-	-	**7.6e-8**	**1.42e-5**	-	-	**3.53e-6**	**1.1e-4**	-	-
	A	**0.07**	0.42	-	-	**0.93**	**0.79**	-	-	**0.15**	**0.25**	-	-

Table 2. Comparison of the results achieved by pTeSSI and TeSSI for the MAX-T and SUM-DIST bidding rules across all three demand cases.

realization volume. Remarkably, pTeSSI also runs substantially faster than TeSSI in practice, despite the fact that the worst-case complexity of both algorithms is the same. This is due to the fact that the conflict-extraction algorithm of Bhargava et al. [3] actually runs faster than Floyd-Warshall in practice, despite both having cubic-time complexity in the worst-case. This confirms both of our main hypotheses $\mathbf{H_{A1}}$ and $\mathbf{H_{A2}}$, supporting the claim that pTeSSI is the more efficient and effective task allocation mechanism in our experiments.

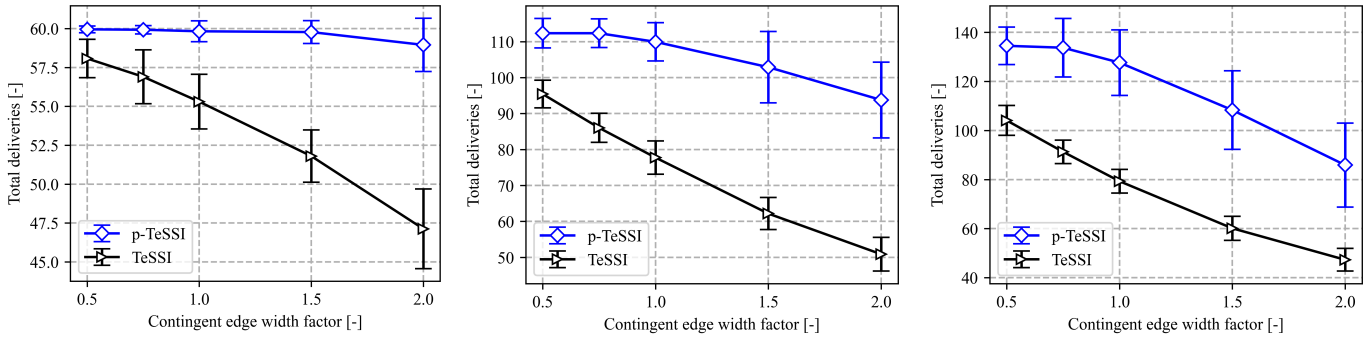
There is no statistically significant difference in the distance traveled per delivery. This suggests that the extra deliveries generated by pTeSSI are compensated by the fact that with TeSSI agents do not travel extra distance due to committing to tasks that will not be executed successfully because of random variables taking on values outside of the controllable realization space. With pTeSSI, the percentage of tasks that are allocated but not executed is at most 1%, 5.3% and 13% for the cases with 60, 120, and 180 orders respectively.

In the TeSSI auction, the SUM-DIST bidding rule tends to lead to more deliveries than MAX-T, with the effect being large for the lowest and highest demand cases. With pTeSSI, on the other hand, we observe no statistically significant difference in the deliveries executed. This may suggest that the risk component of the bidding rule is better suited to the style of optimization performed in MAX-T; that is, minimizing the maximum value rather than the sum across all agents. The auction with the MAX-T runs faster than with SUM-DIST for all demand cases. This is because SUM-DIST needs to compute the objective before and after every task insertion, whereas with MAX-T we only compute the objective after each task insertion. Since computing the objective requires determining the DDC, it is $O(|S|^3)$ where $|S|$ is the number of nodes in the agent's STNU, and the heaviest operation in the bidding procedure. This explains the large effect sizes. We also note that SUM-DIST leads to lower distances traveled per delivery, which matches our expectations given that the aggregate distance travelled by all agents is explicitly part of the objective in this bidding rule. Given that there is no statistically significant difference in the number of deliveries generated by the system and that it is significantly more efficient, we view MAX-T as the better choice for large numbers of tasks. SUM-DIST, instead, is the stronger alternative for smaller problems where runtime is less of a concern or the distance traveled is important. In the rest of our analysis, we focus exclusively on the more efficient MAX-T bidding rule.

Figure 5 shows the number of deliveries executed by the system with both TeSSI and pTeSSI when we vary the width of the ranges associated to the contingent edges. We multiply the width of the nominal ranges in Table 1 by a factor between 0.5 and 2 shown on the x axis. As expected, wider contingent edges make the problem harder, and also tend to increase the performance advantage of pTeSSI over TeSSI. It becomes increasingly more difficult to guarantee dynamic controllability and more rewarding to accept a controlled amount of execution risk. This is most evident in the lowest demand case, where as we discussed in the analysis of Figure 3 and Figure 4 there are combinations of schedules which allow the agents to satisfy nearly all order requests yet retain very high degrees of dynamic controllability. TeSSI rejects these schedules, whereas pTeSSI leverages them.

2. *Normally distributed contingent edges*

We run experiments with normally rather than uniformly distributed contingent edges. We construct normal contingents C_N where the bounds on the uniform edges (Table 1) are two standard deviations away from the mean, i.e. $C_N \sim N(\mu = \frac{U_l+U_b}{2}, \sigma = \frac{U_l+U_b}{4})$ where U_l and U_b are the lower and upper bounds of the uniform contingent edges respectively. Figure 6 shows the performance



(a) 60 orders **(b)** 120 orders **(c)** 180 orders

Figure 5. Total deliveries generated by pTeSSI and TeSSI with varying contingent edge widths, for each of the three demand cases explored. Error bars span one standard deviation from the mean in each direction.

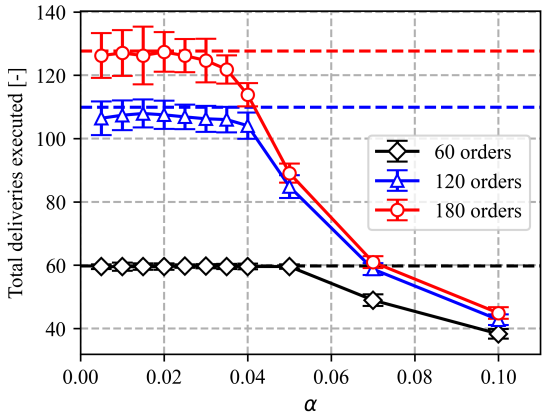


Figure 6. Total number of deliveries executed with pTeSSI in the case with normally distributed contingent edges. Results are plotted with varying α, the factor used in generating the STNU approximation of the underlying PSTN and shown for all three demand cases. Error bars span one standard deviation from the mean in each direction.

of the system with varying levels of α, the approximation factor in the PSTN to STNU mapping. Lower values of α capture more of the original distribution and require more relaxations. Higher values of α truncate larger portions of the underlying distribution and sacrifice more realization volume upfront. The system performs best with the lowest values of α tested, with the total deliveries executed declining rapidly for $\alpha > 0.025$ and $\alpha > 0.5$ for the highest and lowest demand case respectively. This suggests that lowering α increases the success rate achieved when dispatching on the dynamically controllable STNU generated by `ODC-Relax`. This result is consistent with the findings of Akmal et al. [2] and is evidence that `ODC-Relax` is effective in relaxing only those edges that are required to solve conflicts and in doing so minimally. The more room it is granted to choose the relaxations, the better the performance. The dotted lines show the deliveries executed with the uniform contingent edges (from Table 2). With low enough values of α, we achieve nearly as strong a performance as with the uniformly distributed edges. This demonstrates that the relaxations performed on the normally distributed contingent edges are not significantly worse than those on the uniform edges, despite `ODC-Relax`'s choices being agnostic to the underlying distribution. This result confirms that extracting bounds that capture a large portion of the true distribution and subsequently optimizing for realization volume rather than true probability mass is an effective way to deal with non-uniform contingent edges. Note that, while we only experiment with normally distributed edges, we do not expect a significant difference for distributions with non-zero skew. The truncation of the PSTN sacrifices an equal amount of mass from each side of the distribution, thereby naturally accounting for skewed distributions.

B. Experiment set B - dynamic re-auctioning

Here we discuss the results obtained by implementing the periodic re-auctioning mechanism described in Section III. The analysis is conducted only for the `MAX-T` bidding rule, as we have demonstrated earlier that it is more efficient than `SUM-DIST` and we did not observe a statistically significant difference between the number of deliveries executed between the bidding rules.

From our earlier analysis, we know that with pTeSSI the system is able to execute all tasks in the lowest demand level of 60 orders, which would leave no margin for improvement via re-auctioning. We therefore do not consider this case but rather run experiments with demand levels of 120, 180 and 240 orders. Figure 7 shows the total number of deliveries executed by the system for different values of the reauction threshold O. Recall that O represents the minimum drop in system objective required for an agent to de-commit from a task and send it back to the auctioneer for re-auctioning together with the tasks that have newly entered the system. The larger the value of O, the more selective the system. Less tasks will be candidates for re-auctioning and therefore less of

the existing partial allocations will be revisited. Figure 7

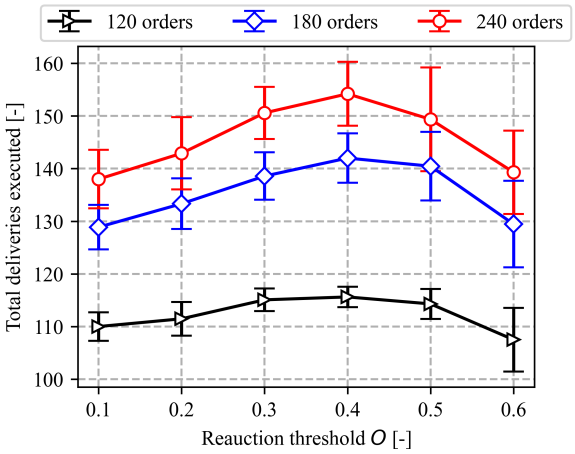


Figure 7. Effect of the re-auction threshold O on the total deliveries generated. Error bars span one standard deviation from the mean in each direction.

shows that the best performance does not correspond to the case in which the most tasks are being re-auctioned (lowest value of O). While this may seem counter intuitive at first glance, it is precisely what can be expected. With the lowest value of $O(0.1)$, the agents consistently de-commit from all unexecuted tasks in their schedule and submit them back to the auctioneer. In each auction, we can look across the combined set of *all* unexecuted tasks in the agents' schedules and the newly incoming tasks. Since we can consider all inter-task synergies available, the *lowest* achievable value of the objective is at its minimum. That is, the best possible allocation at the time of the auction must be one of the choices available. However, sequential auctions like pTeSSI are not guaranteed to find the optimal solution, and the drop in the number of we observe is due to the optimality gap quickly widening when T decreases past 0.4. pTeSSI makes greedier decisions when the agents' schedules are (nearly) empty and more sound ones when it can capture synergies with existing tasks in the agents' schedules. This behavior is common among all sequential auctions and a similar effect was also observed with TeSSI in the work of Nunes [21]. Re-auctioning too many tasks and revisiting good partial allocations at execution, therefore, can negatively impact performance. The fact that the best performance is not with the lowest value of O is an indication that our task selection mechanism is effective in identifying which partial allocations are worth reconsidering, and which not.

We suspect that this might be the reason why the sequential single cluster auctions of Heap and Pagnucco [11], in which *all* unfinished tasks in the agents' schedules are re-auctioned upon completion of a task, at times fail to provide a statistically significant improvement over the case without re-auctioning. We test $\mathbf{H_{B1}}$ and $\mathbf{H_{B2}}$ by comparing the successful deliveries and the average auction duration for the case without re-auctioning to that with a re-auctioning threshold of $O = 0.4$, which is found to lead to the best performance across all demand levels. We test for statistical significance via the Wilcoxon signed rank test and provide the Vargha-Delaney A value as a measure of the effect size, showing the results in Table 3. We confirm both hypotheses, indicating that our re-auctioning mechanism is able to increase the number of deliveries executed by the system, but increases the auction duration in the process. We observe large effect sizes across all demand levels but note that the improvement in the number of successful deliveries increases with the number of orders, suggesting that dynamically revisiting existing partial allocations is more rewarding in harder problems.

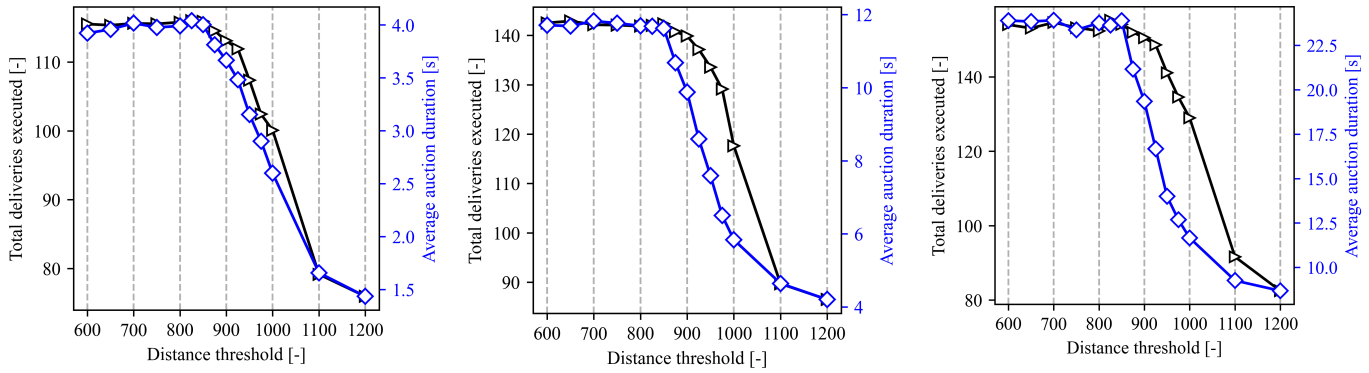
C. Experiment set C - bundling

While our periodic re-auctioning mechanism substantially increases the total deliveries generated, this comes at the cost of longer auction durations since more tasks need to be allocated at once. We attempt to mitigate this effect by allocating bundles rather than individual tasks via pTeSSB as discussed in Section III. Figure 8 shows the average auction duration and total deliveries generated with various levels of the distance threshold in the bundling mechanism. Increasing the distance threshold loosens the requirements for bundle formation, leading to larger and less compact bundles. The auction duration drops significantly with the distance threshold, confirming our expectations and demonstrating that pTeSSB is able to run faster than pTeSSI in practice when bundling occurs. The modifications made to the bidding phase in Algorithm 3 are effective and make the evaluation of a bundle of size B faster than B single-task evaluations (B calls to Algorithm 1). However, as the bundles get larger and less compact, the likelihood that the agents will be able to feasibly insert them in their schedule also decreases, leading to a drop in the total deliveries the system can execute. The distance threshold is effectively a way to regulate the trade-off between auction duration and executed deliveries. It is interesting to note that the trade-off becomes more favorable as number of orders increases. Raising the demand level leads the auction duration to drop more steeply relative to the executed deliveries, as can easily be noted by observing the growth in the area between the two curves.

As discussed in section Section IV.C, we seek to assess whether bundling is effective in significantly decreasing the average auction duration while still preserving a performance improvement over the case without re-auctioning. We test $\mathbf{H_{C1}}$ by comparing the total number of deliveries achieved by pTeSSB-re with a fixed distance threshold to those achieved by pTeSSI, and $\mathbf{H_{C2}}$ by comparing the auction duration of pTeSSB-re to that of

	Total deliveries [-]				Auction duration [s]			
Demand level	pTeSSI	pTeSSI-re	p	A	pTeSSI	pTeSSI-re	p	A
120 orders	109.9	115.6	5.87e-7	0.83	2.08	3.97	3.57e-8	1.0
180 orders	127.7	142.0	2.82e-6	0.86	3.56	11.60	3.57e-8	1.0
240 orders	127.9	153.2	2.19e-7	0.89	6.59	23.25	3.57e-8	1.0

Table 3. Comparison between pTeSSI with and without re-auctioning. Results with re-auctioning are indicated by the extension "-re" and use a reauction threshold of 0.4.



(a) 120 orders **(b)** 180 orders **(c)** 240 orders

Figure 8. Total deliveries generated by pTeSSB-re with varying levels of the distance threshold used in the bundling mechanism. Results are shown for all three demand levels explored.

pTeSSI-re. Based on the analysis in Figure 8, we select a distance threshold of 925 as it seems to significantly decrease the auction duration without largely decreasing the total deliveries in the two higher demand cases. Table 4 shows the outcome of this comparison, including p-values generated from the Wilcoxon signed rank test and the Vargha-Delaney A value as a measure of the effect size. We find support ($p < 0.05$) for both $\mathbf{H_{C1}}$ and $\mathbf{H_{C2}}$ across all demand levels. The effect sizes increase with the demand level, confirming the insight derived from Figure 8 that bundling is more effective for larger numbers of tasks. We are able to generate larger decreases in the auction duration while sacrificing less of the performance increase generated via dynamic re-auctioning. This suggests that when the size of the task set on auction is larger, the bundling mechanism is more likely to find spatio-temporal synergies that will translate to good partial schedules for the agents. In the highest demand case, we are able to retain over 82% of the performance increase while making the auction over 40% faster. In the lowest demand level, on the other hand, we retain less than 40% of the performance advantage while only generating an $\approx 15\%$ improvement in the auction duration.

VI. Conclusion

In this work, we propose solutions to two main limitations of state of the art sequential auction mechanisms for the allocation of temporally constrained tasks that limit their suitability as task allocation mechanisms in real-world distributed systems. Namely, that they are ineffective when tasks have stochastic durations and that they do not re-consider partial allocations over time to uncover new synergies between tasks.

We propose the probabilistic temporal sequential single item (pTeSSI) auction, a novel polynomial-time auction mechanism based on TeSSI [21], designed for the online allocation of tasks with non-deterministic durations and known distributions or bounds on their realization. Agents represent their plans as simple temporal networks with uncertainty (STNUs). Upon attempting to insert a task in their schedule, they leverage the notion of degree of dynamic controllability to efficiently compute the risk of unsuccessful dispatch of the resulting schedule, and include this in their bid. We evaluate the effectiveness of pTeSSI in an online simulation of an on-demand UAV food delivery system under different demand levels. Results show that pTeSSI is effective in allowing agents to leverage controlled amounts of execution risk and substantially outperforms TeSSI for both uniformly and normally distributed task durations of various widths. Even though looking beyond the binary notion of controllabillity enforced by TeSSI requires solving a substantially more complex problem in the bidding phase, we show that pTeSSI retains the same worst-case complexity as TeSSI and runs faster in practice.

We also propose a mechanism whereby agents dynamically select subsets of unexecuted tasks in their

	Total deliveries [-]				Auction duration [s]			
Demand level	pTeSSI	pTeSSB-re	p	A	pTeSSI-re	pTeSSB-re	p	A
120 orders	109.9	111.9	2.42e-2	0.62	3.97	3.48	3.10e-6	0.81
180 orders	127.7	137.2	2.60e-4	0.73	11.60	8.60	4.16e-8	0.97
240 orders	127.9	148.7	5.36e-6	0.81	23.25	16.67	4.48e-8	0.95

Table 4. Comparison between the total deliveries generated with pTeSSI and pTeSSB-re, and between the average auction duration of pTeSSI-re and pTeSSB-re.

schedule and submit them to the auctioneer for re-auctioning. Results demonstrate that dynamically re-auctioning subsets of tasks is effective in improving the quality of the allocation and increases the number of deliveries successfully executed by the agents. We mitigate the additional overhead incurred due to the re-auctioning by constructing bundles of tasks and auctioning them at once, leveraging the temporal relationship of tasks in the bundle to further accelerate the bidding phase. Bundling successfully reduces the run-time of the auction, but also tends to decrease the quality of the allocation. The distance threshold in our bundling mechanism provides a manner to control the trade-off between performance and auction duration, which we find to be more favorable when larger numbers of tasks are on auction. In the higher demand cases explored, combining re-auctioning and bundling is an effective way to limit the increase in auction duration while still retaining a strong performance increase over the case without re-auctioning.

In future work, it would be interesting to explore different bidding rules that address other objectives. A triple objective addressing a combination of the makespan, distance traveled and the risk of unsuccessful dispatch would be a logical extension to the bidding rules explored in our work. Changing the balance between components of the objective function at execution could also be investigated as a way to further improve performance. pTeSSI leverages the recently proposed notion of dynamic controllability of STNUs [2]. As the domain of temporal networks is constantly evolving, novel theory or alternative characterizations of controllability could also pose interesting opportunities for further development of pTeSSI.

References

[1] S. Akmal, S. Ammons, H. Li, and J. C. Boerkoel, "Quantifying Degrees of Controllability in Temporal Networks with Uncertainty," in *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS)*, 2019, pp. 22–30.

[2] S. Akmal, S. Ammons, H. Li, M. Gao, L. Popowski, and J. C. Boerkoel, "Quantifying controllability in temporal networks with uncertainty," *Artificial Intelligence*, vol. 289, 12 2020.

[3] N. Bhargava, T. Vaquero, and B. Williams, "Faster Conflict Generation for Dynamic Controllability," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017, pp. 4280–4286.

[4] B. Butcher, K. W. Lim, and Y. Sheffi, "Assessing Feasibility of the Delivery Drone," Massachussets Institute of Technology (MIT), Tech. Rep., 5 2019.

[5] A. Chakrabarty and C. Ippolito, "Autonomous flight for multi-copters flying in UTM -TCL4+ sharing common airspace," in *AIAA Scitech 2020 Forum*, vol. 1 PartF. American Institute of Aeronautics and Astronautics Inc, AIAA, 2020.

[6] A. Chakrabarty, C. Ippolito, J. Baculi, K. Krishnakumar, and S. Hening, "Vehicle to vehicle (V2V) communication for collision avoidance for multi-copters flying in UTM-TCL4," in *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics Inc, AIAA, 2019.

[7] R. D'Andrea, "Guest Editorial: Can Drones Deliver?" *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 3, pp. 647–648, 2014.

[8] C. Fang, P. Yu, and B. C. Williams, "Chance-constrained Probabilistic Simple Temporal Problems," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14)*, 2014, pp. 2264–2270. [Online]. Available: www.aaai.org

[9] Federal Aviation Administration (FAA), "Unmanned Aircraft System (UAS) Traffic Management (UTM) - Concept of Operations," Washington, DC, Tech. Rep., 2020.

[10] M. Gansterer and R. F. Hartl, "Centralized bundle generation in auction-based collaborative transportation," *OR Spectrum*, vol. 40, no. 3, pp. 613–635, 7 2018.

[11] B. Heap, "Sequential Single-Cluster Auctions for Multi-Robot Task Allocation," Ph.D. dissertation, University of New South Wales (UNSW), 2013.

[12] S. Koenig, P. Keskinocak, and C. Tovey, "Progress on Agent Coordination with Cooperative Auctions," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*. Atlanta, Georgia, USA: AAAI, 2010, pp. 1713–1717.

[13] S. Koenig, C. Tovey, X. Zheng, and I. Sungur, "Sequential Bundle-Bid Single-Sale Auction Algorithms for Decentralized Control," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India, 2007, pp. 1359–1365.

[14] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, and S. Jain, "Auction-Based Multi-Robot Routing," in *Proceedings of the International Conference on Robotics: Science and Systems*, 2005, pp. 343–350.

[15] J. Los, F. Schulte, M. Gansterer, R. F. Hartl, M. T. J. Spaan, and R. R. Negenborn, "Decentralized Combinatorial Auctions for Dynamic and Large-Scale Collaborative Vehicle Routing," in *Preprint - ICCL 2020*, 2020.

[16] K. Lund, S. Dietrich, S. Chow, and J. C. Boerkoel, "Robust Execution of Probabilistic Temporal Plans," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 3597–3604.

[17] P. Morris, "A structural characterization of temporal dynamic controllability," in *Proceedings of Principles and Practice of Constraint Programming (CP-06)*, vol. 4204 LNCS. Springer Verlag, 2006, pp. 375–389.

[18] P. Morris, "Dynamic controllability and dispatchability relationships," in *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems (CPAIOR 2014)*, vol. 8451 LNCS. Springer Verlag, 2014, pp. 464–479.

[19] M. Nilsson, J. Kvarnström, and P. Doherty, "Classical Dynamic Controllability Revisited - A Tighter Bound on the Classical Algorithm," in *Proc. of 6th International Conference on Agents and Artificial Intelligence (ICAART-14)*, Angers, France, 3 2014, pp. 130–141.

[20] E. Nunes, "Decentralized Allocation of Tasks with Temporal and Precedence Constraints to a Team of Robots," Ph.D. dissertation, University of Minnesota, 2017.

[21] E. Nunes and M. Gini, "Multi-Robot Auctions for Allocation of Tasks with Temporal Constraints," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. Austin, Texas, USA: AAAI, 2015, pp. 2110–2116.

[22] A. Pasztor and K. S. Ferek, "FAA Approves First Fully Automated Commercial Drone Flights," 1 2021.

[23] S. Ponda, "Robust Distributed Planning Strategies for Autonomous Multi-Agent Teams," Ph.D. dissertation, Massachussets Institute of Technology (MIT), 2012.

[24] C. Tovey, M. G. Lagoudakis, S. Jain, and S. Koenig, "The Generation of Bidding Rules for Auction-Based Robot Coordination," in *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, L. Parker, F. Schneider, and A. Schultz, Eds. Dordrecht: Springer, 2005, pp. 3–14.

II

Literature Study
previously graded under AE4020

Introduction

The idea of UAV delivery in urban areas has received remarkable attention over the past few years, with large global players such as Amazon, Uber, DHL and several well-funded startups already deploying prototype systems and well poised to begin monetizing on this technology in the near future. Delivery UAVs have the potential to unlock considerable economic benefits by dramatically cutting costs in the last mile of global supply chains, while also enabling the on-demand delivery of wide ranges of goods to customers in urban areas at unmatched speed. Widespread social distancing measures due to the current COVID-19 pandemic have also highlighted the importance of reducing the level of human interaction required in the delivery of essential goods. UAV delivery providers such as Alphabet's Wing have seen a significant spike in demand due to the pandemic and have responded by expanding delivery options to include food and other essential items [122]. Several others such as Matternet, UPS and Zipline have mobilized UAV fleets to deliver medical supplies [119] and test samples [92].

In order to reap the vast economic and social benefits of UAV delivery, however, operators need to deal with regulatory uncertainty and a number of technical challenges, across areas such as design, autonomous navigation and fleet management. Among the driving challenges is the development of online coordination paradigms to allow UAV fleets to autonomously allocate and plan on-demand delivery tasks while avoiding conflicts with other vehicles and obstacles in the environment, which constitutes the focus of this thesis project. Autonomous coordination in the context of on-demand UAV delivery is a highly challenging problem in which the UAVs are required to perform online task allocation and path finding, which are both known to be NP-hard problems[1]. Delivery requests need to be allocated in real-time and given that the urban low-altitude airspace is envisioned to be highly dynamic, efficient re-planning procedures should be possible as the fleet discovers new information about the environment. Capturing the dynamism and planning constraints at hand poses novel challenges in multi-agent coordination and requires substantially extending state of the art techniques from the multi-agent systems literature.

This literature study is divided in four main chapters. Chapter 1 provides an analysis of the setting and environment in which delivery UAVs will need to operate, which is essential to understanding the requirements that will be imposed on the coordination framework. This includes a discussion of the unit economics of UAV delivery, operating constraints of the UAVs and how the low altitude urban airspace will be managed and structured. Chapter 2 provides a thorough analysis of the multi-agent task allocation (MATA) problem, which is the driving challenge in UAV delivery coordination, and state of the art classes of MATA solvers. A tradeoff among the solver classes - spanning a number of different fields such as operations research, game theory and distributed artificial intelligence - is performed, and the most suitable class of approaches is identified and discussed in further detail. Chapter 3 includes a discussion of several multi-agent path finding (MAPF) solvers and a tradeoff based on their driving characteristics. Note that, within this project, we only consider the coordination aspect of path finding and therefore direct our attention to techniques from the multi-agent systems literature which are designed for simple grid-like environments but explicitly consider the interaction between the agents' paths and the conflict resolution aspect of path finding. We do not discuss single-agent approaches specifically tailored to operate in continuous 3d environments or UAV-specific path finding routines which explicitly consider the kinodynamic limitations of the vehicles and the control effort required to execute a given path. In Chapter 4 we present a research proposal for the thesis project based on the findings of the literature study. We define the overarching research objective, translate it into a series of sub-questions and construct work packages to answer the questions in an exhaustive and structured manner.

[1] Aside from special cases, which as we will see in Chapter 2 and Chapter 3, do not apply to the UAV delivery coordination case.

1
Urban UAV delivery

In this chapter, we provide an overview of key operational considerations for UAV delivery. We start by elaborating on different operating models in Section 1.1 and perform a first-principles analysis of the economic viability of a realistic UAV delivery operation in Section 1.2. In Section 1.3 we discuss the key technical challenges to overcome in order to make UAV delivery a reality, and explain why our focus will lie on the UAV coordination problem. We then explore the setting and constraints in which delivery UAVs will have to operate by discussing the characteristics of the latest proposed unmanned aircraft system traffic management (UTM) architectures in Section 1.4. Finally in Section 1.5 we provide a high-level formulation of the UAV delivery coordination problem, which forms the basis of this thesis, and identify relevant research areas to investigate further.

1.1. Operating models

UAV delivery is being investigated by several companies with different operating models in mind. The two most common operating models are *last-mile delivery*, in which large retailers and delivery players intend to use UAVs in the last mile of their global supply chain, and *urban delivery as a service*, in which the goal is to offer cost-effective on-demand delivery of goods directly from local businesses to customers in urban areas.

1.1.1. Last-mile delivery

Current last-mile delivery models are, generally speaking, highly inefficient and difficult for delivery service providers to sustain. According to a recent report by Capgemini, approximately 41% of the overall supply chain costs are incurred in the last-mile, which is more than double the costs incurred in any other link of the supply chain [73]. A highly competitive delivery market pushes many retailers to absorb part of the costs associated to last-mile delivery, charging customers on average 20% less than the cost incurred per customer order to retain or grow their market share. This has led several large e-commerce players and delivery providers such as Amazon, UPS and DHL to investigate the incorporation of UAVs in the last-mile of their supply chain to cut costs and boost their profitability. Amazon's Prime Air project in particular has attracted widespread media attention. The goal is to use UAVs to carry small ($\approx$ 2.2kg) packages from its warehouses to the customer and thereby guarantee delivery in 30 minutes or less of items that are readily available at the warehouse.

While Amazon Prime Air intends to carry packages from the warehouse directly to the customer's doorstep via UAV, alternative concepts have been studied in the literature. A commonly discussed alternative is to deliver packages to set pickup locations in order to enhance safety and reduce planning complexity [95]. This prevents UAVs from having to land or operate close to customers, which is a risky and technically challenging endeavor. Recent work has studied concepts of operations in which trucks and UAVs collaborate to fulfill deliveries. These include models in which trucks act as base stations for UAVs, with the UAVs dispatched from a moving truck which is also delivering to customers [7][83]. Others have simply studied the performance of mixed fleets in which trucks are dispatched for certain deliveries and UAVs for others [176]. Mixed fleets perform well because trucks and UAVs are highly synergistic delivery modes. The advantage of trucks is that they can service several deliveries at once and have high ranges, but they are limited by traffic and slow in urban areas. Trucks are therefore best suited for bulk deliveries with loose time constraints. UAVs, on the other hand, are most advantageous for time-sensitive deliveries of small items to more isolated customers [176]. This explains why the industry is not looking to replace other delivery modes entirely, but rather to enhance their existing supply chain by using UAVs for faster and more cost effective *on-demand* deliveries.

1.1.2. Urban delivery as a service

Another set of companies intends to use UAVs not as the last mile of a global supply chain, but rather as means to offer on-demand urban deliveries as a service. The objective is to transport goods from local businesses to customers via a fleet of delivery UAVs, at speeds and costs that other delivery modes cannot reach. Among

many examples of startups in this space is Flirtey, which is part of NASA's UTM program and has received FAA approval to conduct beyond visual line of sight (BVLOS) UAV deliveries. Flirtey intends to operate a general-purpose delivery service in urban areas, delivering all sorts of items including food, medical devices, retail items and more. Other players focus on a specific class of delivery goods, with food delivery being the most popular. Food is a natural choice for UAV delivery because orders are typically physically light and highly time-sensitive since the food should not get cold before it reaches the customer. Traditional food delivery modes such as human-operated e-bikes are highly inefficient because there are limited opportunities to deliver multiple orders at once (which is the advantage of delivery trucks) since timely delivery to the customer is paramount and items need to be picked up in different locations. The food delivery market is crowded and highly price-sensitive since there is little room for differentiation, meaning that several providers need to subsidize their delivery costs in order to stay competitive. Most companies in the space are therefore growing in a highly unprofitable manner, with Uber's Eats business as a prime example. Several of these companies have looked to UAVs to cut delivery costs, undercut their competition and improve their margins. Uber Eats has recently unveiled a new design for its delivery drone and is expecting to start drone deliveries in San Diego, where it has already conducted a few tests, this summer [24]. In Section 1.2.3, we take a deeper look at the unit economics of Uber Eats and how UAV delivery could affect its business.

1.2. Economic viability

An important question to answer is whether UAVs actually constitute an economically viable solution to the last-mile delivery problem, and if so to what extent and under which limitations. While the vast amount of companies that are active in this space can be seen as empirical evidence that the solution is indeed an economical one, it should not be deemed as conclusive evidence. This is because none of the companies actively working to bring a UAV delivery network to market have released their unit economics to the public, nor are they inclined to do so. It is therefore important to reason independently and via *first principles* about the economic viability of UAV delivery. We shall attempt to derive a high-level estimate of the cost per km of operating a delivery UAV and draw a parallel to existing delivery systems.

1.2.1. Battery and energy cost

We begin by following the approach of D'Andrea [36][1] to estimate the required battery and energy cost per km flown by the delivery UAV. In order to do this, we need to make informed assumptions about relevant performance metrics e.g. cruise speed, range, payload capability, maximum headwind and derive the corresponding battery and energy requirements. Equation (1.1) and Equation (1.2) show approximations of the power requirement and the worst-case energy requirement respectively, the latter equation assuming that the UAV needs to fly its maximum range while flying in the maximum headwind w it can sustain for the entirety of the journey. Equation (1.3) and Equation (1.4) estimate the energy and battery costs per km flown. All relevant variables and assumptions are summarized in Table 1.1. The most important assumptions are discussed further.

It is assumed that the allowed payload mass is of 2.2 kg, which is in line with Amazon's plan and approximately 86 percent of all e-commerce orders reportedly fit this requirement [44]. The cruise speed v is taken to be 50 km/h, which is well within the capabilities of small modern UAVs and quadcopters, and corresponds to the street speed limit within Dutch cities. Operating at such a cruise speed would allow UAVs to deliver faster than traditional last-mile delivery methods such as vans, scooters or e-bikes which must comply with the speed limit and other traffic regulations, and lose further time due to congestion. A study by TNO notes that the average speed of vehicles in urban Dutch areas is ≈ 44 km/h under free-flow (no traffic) conditions, ≈ 25 km/h with normal traffic conditions and ≈ 12 km/h under heavy traffic conditions [93]. While there is still no regulatory consensus on the speeds at which UAVs will be allowed to fly in the low-altitude urban airspace, studies suggest that a small UAV[2] colliding with a human at a speed of 50 km/h would cause injuries well below the critical threshold [131] and would therefore be relatively safe from a third-party risk perspective. The maximum headwind w the UAV should be able to sustain is taken to be equal to 35 km/h. Based on weather data [3], in Amsterdam this would allow the UAV to operate on average for 328 days a year (≈ 90% of days). It is also in line with the common rule of thumb for small commercial quadcopters that the strongest wind in which it is safe to fly is approximately two thirds of the drone's maximum speed, since the UAV's maximum speed should be

[1] Raffaello D'Andrea is a distinguished robotics researcher and professor at ETH Zurich. He also co-founded Kiva Systems (now Amazon Robotics), which was acquired by Amazon as part of its effort to automate warehouse operations and build core capabilities for UAV delivery. He has advised extensively on the topic of UAV deliveries and performed a feasibility study for Matternet, part of which was made public in [36].

[2] More specifically, the study simulates the impact of a DJI Phantom 3 (weight of ≈ 1.3 kg) with the human head under a range of impact speeds and angles[131].

slightly above its cruise speed of 50 km/h in order to cruise safely and perform corrections or sense and avoid maneuvers if necessary.

The required range for the UAV is an important design parameter and difficult to generalize since it will be highly dependent on the specific application, environment and in particular the distance between the charging stations of the operator. Figure 1.1 shows a plot of the required battery mass for different required UAV ranges, with all other parameters fixed according to Table 1.1. The required battery mass m_b is the maximum smallest mass that satisfies both Equation (1.1) (power requirement) and Equation (1.2) (energy requirement). The required battery mass does not vary with the range when the power requirement is dominant (which is a straightforward observation since power is an instantaneous requirement and therefore the range does not appear in Equation (1.1)), but varies steeply and superlinearly in the range once the energy requirement kicks in. For the purpose of this cost estimation, we select d = 16km as the required UAV range. This would allow an operator of a UAV delivery system to serve customers in the vast majority of the greater Amsterdam area via a return trip from a single strategically located charging hub, while keeping the battery mass reasonable. For example, Figure 1.2 considers the fictitious case in which Uber adopts this model for its Eats business, with a single charging hub located at its Amsterdam headquarters. As can be seen, the vast majority of the Amsterdam area lies within 8km of the station and could therefore theoretically be served. The chosen range is also in line with Amazon's vision for last mile delivery, since the company is reportedly designing for a similar range. Naturally, all operational requirements in Table 1.1 and in particular the design range, should be studied in more detail by the operator and tailored to the specific operating model at hand. In several cases it would be more efficient, for instance, to design for more charging stations clustered around high-demand areas, cutting delivery times and allowing for a more relaxed range requirement which brings battery and energy costs down. The intent here is simply to make sound assumptions that are generalizable to a wide range of operating models in order to arrive at a reasonable high-level cost estimate.

With the assumptions as stated in Table 1.1 we arrive at a required battery mass of **2.3 kg** assuming energy and power densities of 0.25 $\frac{kW \cdot h}{kg}$ and 0.35 $\frac{kW}{kg}$, which are attainable with modern lithium-ion batteries. While the battery mass may seem high compared to similarly sized commercial quadcopters which are not designed for range, it is reasonable and in line with similar estimates in the literature [36]. The battery amounts to ≈ 35% of the weight of the entire vehicle including the payload, which interestingly is close to the 38% - 45% fuel fraction range that modern long-haul commercial aircraft are designed for. Given current energy and battery costs, this would result in a cost of **0.35 EUR cent per km** for the battery and **0.27 EUR cent per km** for the energy. These costs are likely to decrease significantly over the coming years due to competitive pressure and economies of scale. The average global cost of lithium-ion batteries has already dropped with a CAGR of over 25% from 2014 to 2018 [61]. With the current momentum in the electric car domain such as Tesla ramping up battery production efforts via its gigafactories and several new electric car manufacturers emerging particularly within the Chinese market, the price of batteries can be expected to continue decreasing substantially. Competition among electric car manufacturers will also drive developments in battery technology, so it is reasonable to expect an accelerated improvement in battery power and energy density. The unit economics will soon look even more favorable than currently estimated.

Power consumption:

$$P_{\text{req}} = \frac{\left(m_p + m_v + m_b\right)}{370 \cdot \eta \cdot \frac{C_L}{C_D}} + p \tag{1.1}$$

Worst-case energy requirement:

$$E_{\text{req}} = \frac{d}{1 - \frac{w}{v}} \left(\frac{m_p + m_v + m_b}{370 \cdot \eta \cdot \frac{C_L}{C_D}} + \frac{p}{v} \right) \tag{1.2}$$

Energy cost per km:

$$C_{\text{energy}} = \frac{c}{e} \cdot \left(\frac{m_p + m_v + m_b}{370 \cdot \eta \cdot \frac{C_L}{C_D}} + \frac{p}{v} \right) \tag{1.3}$$

Battery cost per km:

$$C_{\text{battery}} = \frac{k}{l} \cdot \left(\frac{m_p + m_v + m_b}{370 \cdot \eta \cdot \frac{C_L}{C_D}} + \frac{p}{v} \right) \tag{1.4}$$

Yearly km travelled by UAV:

$$d_{\text{year}} = 365 \cdot (1 - l_w - l_e) \cdot h \cdot (1 - l_d) \cdot v \tag{1.5}$$

Total operating cost per km:

$$\begin{aligned} C_{\text{tot}} = & 2 \cdot C_{\text{battery}} + C_{\text{energy}} + C_{\text{insurance}} \\ & + C_{\text{maintenance}} + C_{\text{labor}} + C_{\text{airspace}} \end{aligned} \tag{1.6}$$

1.2.2. Other operating costs

In order to estimate the cost per km of the other operating cost components, we need to reason about operational characteristics of the UAV delivery system. Firstly, while the end goal is to achieve full autonomy, there will be human operators required for supervisory purposes and especially so in the first stages of deployment. We follow the most conservative estimate in [44] and assume that there will be one operator required for 410

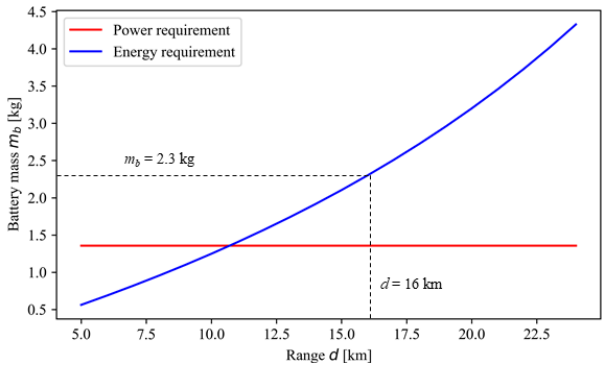


Figure 1.1: Required battery mass with varying design range, based on Equation (1.1) (power requirement) and Equation (1.2) (energy requirement).

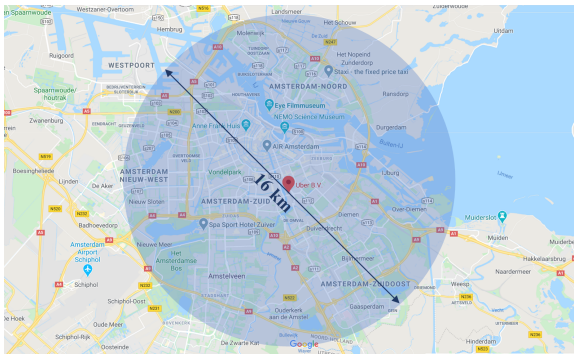


Figure 1.2: Illustration of area that could be covered by a UAV with 16km range given a single charging hub coinciding with Uber's EMEA headquarters in Amsterdam.

UAVs and that their labor will cost 30 EUR per hour. We also refer to estimates made in [44] and assume that the yearly maintenance cost per UAV will be EUR 428 and that the yearly liability insurance cost will be of EUR 500 per year for each vehicle. The latter estimate is in line with currently available insurance packages [111] and can be expected to decrease further as UAV use becomes more widespread and more insurance providers enter this market. While there is no consensus yet on how the low-altitude airspace will be commercially managed for unmanned traffic, there will most likely be costs associated to operating a vehicle. These costs are required to make unmanned traffic management a profitable business and may be fixed or vary dynamically to balance demand and capacity in the airspace. For the purpose of this cost estimation, we refer to the analysis made in [44] and assume that it will cost an average of EUR 50 cents per hour to operate a small UAV in the urban low-altitude airspace.

We assume that the delivery system will be operating for 6 hours a day, with 20% downtime to account for operating inefficiencies and time lost at pick-up and delivery locations or charging stations. Every UAV will therefore be operating at its cruise speed v for 4.8 hours a day[3]. In Section 1.2.1 we note that the headwind limitations of the UAV would only allow it to operate in the worst-case scenario on 90% of days in the Amsterdam area. We therefore assume that the delivery system will be down for 10% of the year due to adverse weather conditions. This is a conservative assumption, since only in extreme circumstances will intense winds persist for the entire day and in principle the 6 hour window could be shifted and centered around less windy hours if weather conditions require it. We assume the system will be down on another 5% of days due to unforeseen circumstances, such as technical issues, security breaches, or third-party safety concerns like large gatherings or events within the operating area. With the above assumptions we conclude that a single UAV will travel 74,460 km per year (see Equation (1.5)).

We then arrive to the total operating cost per km via Equation (1.6). Note that all yearly costs in Table 1.1 are normalized by the distance travelled per year in km, while the hourly costs are normalized by the average velocity of the drone in the 6 hour window in km per hour. This means we assume that the hourly costs (airspace usage and operator salaries) also apply when the UAV is idle. The battery cost is multiplied by 2 because we assume that the operator will order twice the required batteries per UAV, in order to ensure a newly charged battery can always be swapped in at the charging station. In reality the extra batteries required to ensure a smooth operation will likely be significantly less for a large fleet of UAVs in which the spare batteries can be shared among the fleet, since modern charging technology allows Li-ion batteries to be charged very quickly[4].

Given the above assumptions, the operating costs per vehicle amount to **EUR 3.3 cents per km**, which is in line with other estimates in the literature. Matternet estimated it would cost EUR 24 cents to deliver a 2 kg package over a 10 km distance [129] implying a cost per km of 2.4 cents, including all related infrastructure and investment costs. Via its Prime Air service, Amazon estimates it can deliver similarly heavy payloads over 16km for a total cost of EUR 71 cents [179], implying a cost per km of EUR 4.4 cents.

[3] An alternative way to look at it is that the UAV will be operating at an average speed of $0.8 \cdot v = 40$ km/h throughout the daily 6 hour operating window.

[4] For comparison, a Tesla Model 3 battery takes approximately one hour to charge fully on a Tesla V3 Supercharger and weighs 480 kg.

Parameter	Definition	Value
UAV requirements and characteristics		
m_p	Payload mass	2.2 kg
m_v	Vehicle mass	2 kg
$\frac{C_L}{C_D}$	Lift to drag ratio	3
η	Power transfer efficiency	0.5
p	Power consumed by electronics	0.1 kW
v	Cruise speed	50 km/h
w	Maximum headwind	38 km/h
d	Range	16 km
Battery, energy and power delivery		
P_s	Specific power of battery	0.35 $\frac{kW}{kg}$
E_s	Specific energy of battery	0.25 $\frac{kW \cdot h}{kg}$
c	Cost of electricity	0.2 EUR/kW·h
e	Charging efficiency	0.8
k	Battery cost	163 EUR/kW·h
l	Life of battery	500 cycles
m_b	Battery mass	2.3 kg
Operating assumptions		
	UAVs per operator	410
	Operator hourly salary	30 EUR
	Yearly insurance cost per UAV	500 EUR
	Yearly maintenance cost per UAV	428 EUR
	Operating hours per day	6 hours
	Percent days inoperable due to weather	10%
	Percent days inoperable due to other factors	5%
	Daily downtime percentage	15%
Operating costs per km		
$C_{battery}$	Battery cost per km	0.35 EUR cent
C_{power}	Electricity cost per km	0.27 EUR cent
$C_{insurance}$	Insurance cost per km	0.67 EUR cent
$C_{maintenance}$	Maintenance cost per km	0.57 EUR cent
C_{labor}	Labor cost per km	0.15 EUR cent
$C_{airspace}$	Airspace usage cost per km	1.00 EUR cent
T_cost	**Total cost per km**	**3.3 EUR cent**

Table 1.1: Summary of assumptions and results in the economic viability analysis of Section 1.2.

1.2.3. Comparison with existing food delivery system

We use the estimated delivery cost per km to draw a high-level economical comparison between a current food delivery system and a possible UAV delivery system. We select food delivery as the use case for this analysis because while there is already plenty of consensus in the literature regarding the economic benefits of UAV last-mile delivery [73], there are few studies about the economics of food delivery via UAVs as compared to existing systems. We base the analysis on the food delivery provider Uber Eats because it has transparent requirements for rider fees and its parent company Uber is publicly traded so its financials are available.

It is an official policy of Uber Eats in Amsterdam that riders are guaranteed betweeen EUR 10.0 and 11.3 per hour for deliveries between 17.00 and 23.00 provided they perform at least one delivery per hour[5] [2]. In the optimistic case that a rider completes 4 orders per hour on average, this brings Eats' labor costs per delivery to **EUR 2.5 per order**. In reality, Eats incurs several additional costs per order, but labor is considered by far the largest so we use it as a conservative estimate for Eats' total operating costs. In contrast, assuming the average distance to be travelled to execute a full order (including traveling to the restaurant and then to the customer) is of 4 km, the total operating costs associated to delivery via UAV would be a mere **EUR 13.2 cents per order**, reducing the cost by approximately 19 times.

We refer to Uber's annualized latest quarter financial results[6] to understand what the impact of implementing a UAV delivery system could look like for Eats. The analysis is summarized in Table 1.2. An important note is that for every order Eats only recognizes as revenue the total paid by the customer minus the rider fee and the share taken by the restaurant. The Eats business is growing rapidly (68% yearly revenue growth) but in a highly unprofitable manner with an EBITDA margin of -62%. A major issue is the low 17% take rate, due to the high share going to the restaurant and the high delivery costs to Eats. Assuming Uber Eats continues to serve the same demand (total bookings and average order value stay constant), operating a UAV delivery system would allow the company to increase its take rate to 26% and achieve a 54% increase in revenue. With such a drastic decrease in delivery costs, Uber can also undercut its competition by offering lower delivery fees to the customer and more favorable unit economics for its restaurant partners. Since in the food delivery space both customers and restaurants are very price sensitive, this would allow Eats to substantially increase its market share and grow the top line while still improving its take rate, making the economic benefits far greater than the estimated revenue growth.

Uber Eats
Q4 2019 annualized financials [5]

	Current system	**UAV delivery**
in EUR million		
Total bookings	15,921.4	15,921.4
GAAP Revenue	2,671.8	4,116.8
Take rate	17%	26%
in EUR		
Average order value[7]	26.1	26.1
Delivery cost	**2.50**	**0.13**
Other costs	19.2	19.2
Take on order	4.4	6.7

Table 1.2: Impact of operating a UAV delivery network for Uber Eats, based on the company's latest quarter annualized financials.

1.3. Technical challenges

While the economic and social benefits of UAV delivery are evident, there are several technical challenges to be overcome in order to make it a reality. Based on the existing literature, we distinguish among three main challenges: UAV design, localization and navigation, and UAV coordination. The challenges are summarized in Figure 1.3.

[5] The exact guaranteed rates vary based on the day of the week and the time window.
[6] Q4 2019 is the latest quarter for which financials are available at the time of writing.

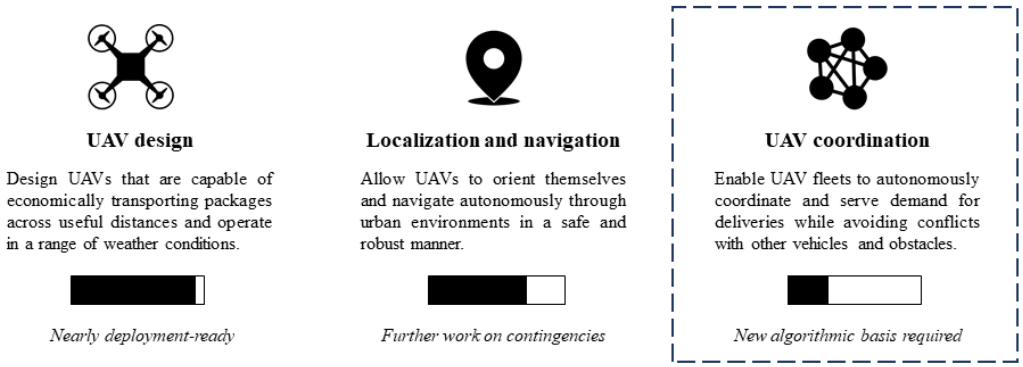


Figure 1.3: Main technical challenges for UAV delivery.

UAV design
A successful UAV delivery service hinges on the design of UAVs that are capable of economically transporting packages across useful distances and can operate in a range of weather conditions. UAV technology is already very advanced, due to the extensive military applications of UAVs and the recent boom in popularity of other use cases such as photography, surveillance and even drone racing. Industry is therefore already far ahead in this domain, with several companies such as Amazon [90], Wing [67] (an Alphabet company), DHL [133] and Uber Eats [66] unveiling and beginning to operate prototypes of their delivery UAVs. The performance characteristics of these UAVs (e.g. reported ranges and top speeds) are already strong enough to enable highly profitable UAV delivery operations, as discussed in Section 1.2.

Localization and navigation
Enabling autonomous UAVs to orient themselves and safely navigate in urban environments is a challenge on many fronts. Relevant technology is improving rapidly, with research efforts driven by tech companies (e.g. Google and Amazon) and accelerated by several startups pioneering autonomous drone technology (e.g. Skydio or Exyn). The boom of self-driving cars is also pushing the development of localization and navigation technologies, such as improvements in computer vision algorithms and SLAM. Computer vision algorithms are crucial in the package pick-up and delivery phases, where UAVs may need to operate in close proximity of humans. The same goes for sense and avoid maneuvers, for which UAVs need to be able to reliably detect unexpected obstacles in real-time. Simultaneous localization and mapping (SLAM) is a technology that enables autonomous agents to perform vision-based mapping and navigation using only on-board sensors. This allows UAVs to operate in previously unknown environments with limited GPS coverage. While several of the underlying technologies already exist, further work is required on contingencies and experimental campaigns should be conducted to assess the safety and robustness of autonomous navigation systems for UAVs in urban environments.

UAV coordination
In order to effectively serve the demand for deliveries, operators require suitable fleet coordination paradigms. These should be mostly autonomous and require minimal human supervision in order to bring down the labor cost and be economically advantageous as discussed in Section 1.2. A suitable paradigm should balance several interconnected problems while considering the performance limitations of the UAVs. The fleet should be able to perform *task allocation*, that is assign UAVs to customer orders while minimizing some combination of system cost parameters. The UAVs need to be able to perform *path finding* in order to generate feasible and low-cost trajectories in the environment that allow them to execute delivery tasks while obeying airspace rules, if any. Multi-agent systems provide a natural way to reason about the UAV coordination problem, because they provide a framework to model their behavior and interaction given different goals and communication paradigms. Several of the above subproblems have in fact been studied in isolation in the multi-agent systems literature. However, there is a lack of integrated approaches that provide a framework applicable to the UAV coordination problem in the context of deliveries in urban environment. There is a strong need to extend relevant classes of methods and explore algorithmic paradigms which are relevant to this problem. We therefore consider the UAV coordination problem as the greatest technical inhibitor to the implementation of an economically viable UAV delivery network, and select it as the focus of this thesis. In Section 1.4, we will discuss the latest thinking regarding unmanned aircraft system traffic management (UTM) in order to better understand the setting, regulations and requirements under which UAVs must coordinate. In Section 1.5 we define the UAV coordination problem which will form the basis of this thesis and identify relevant research areas to be

explored.

1.4. Managing UAV traffic

In order to enable UAV deliveries in an urban environment, it is important to consider their safe integration into the airspace. This falls in the domain of *unmanned aircraft system traffic management* (UTM), which is an important topic of discussion within the aviation community. UTM is concerned with developing concepts to ensure UAVs can effectively share the low-altitude airspace to perform different missions. While discussions on regulatory and technical aspects of future UTM systems are still underway, NASA is already prototyping its UTM system in collaboration with the FAA and several industry partners. NASA employs a crawl-walk-run strategy to the technical development of the UTM system, in which the required capabilities are built up in four tiers targeting different applications, geographical areas and risk levels. Figure 1.4 shows an overview of the technical capability levels (TCL) considered by NASA in the rollout of their prototype UTM system. Most functionalities within TCL 4, which relates to beyond visual line of sight (BVLOS) UAS operations in densely populated urban areas, have already been tested and validated in flight experiments.

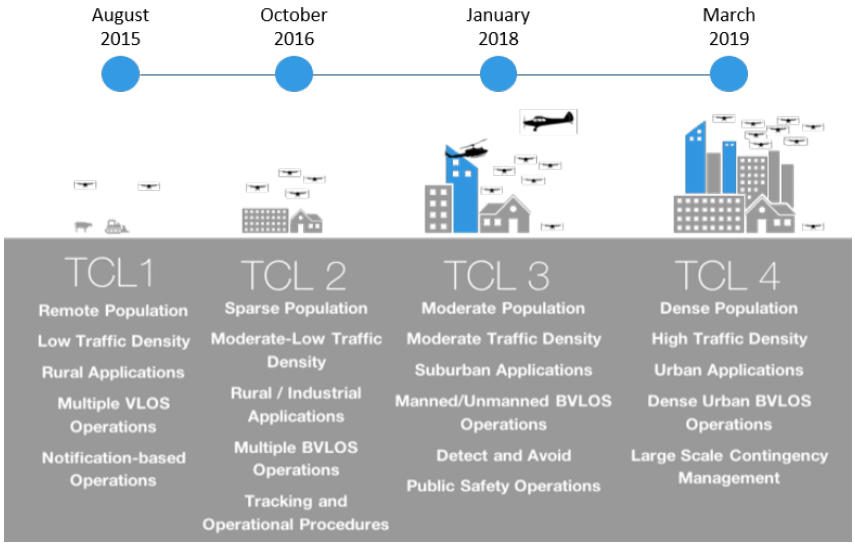


Figure 1.4: NASA technical capability levels (TCL). Figure adapted from [79] and dates taken from talk given by the principal investigator of NASA's UTM project [87].

1.4.1. Principles of UTM

NASA is deemed one of the thought leaders in UTM, and their vision is shaped by collaborations with government, industry and academia alike. It revolves around the fulfillment of three fundamental principles:

1. **National and regional security**: it is essential to ensure that UAS operations do not threaten national and regional security. This involves the protection of key ground assets (e.g. the White House in the US or key landmarks) as well as people, against three types of threats: (1) rogue or unauthenticated systems that purposely aim to cause damage to key assets, (2) authenticated systems that enter geo-fenced areas without prior approval or (3) authenticated systems that have been hacked and are being leveraged to cause harm [124].
2. **Safe airspace integration**: a fundamental requirement in UTM is that UAS should be able to safely share the airspace with other UAS and traditional aviation. This is a challenging principle to fulfill because UAS operators will include commercial entities with often conflicting goals and interests. Smooth, continuous cooperation between these operators, at much higher frequencies than in commercial aviation, is required to ensure safe operations in urban areas.
3. **Scalable operations for economic growth**: the economic value of enabling urban UAS operations is significant, in particular for deliveries as made clear by the analysis provided in Section 1.2. According to a report by McKinsey [33], over USD 3 billion in capital has already been allocated to urban UAS startups. It is imperative to safeguard this value and ensure that the UTM system is able to fulfill the first two principles while still allowing operators to reap significant economic benefits. It is therefore important to collaborate with prospective operators in the development of the UTM system, and to develop tools for operators that incentivize them to use the airspace in a profitable and scalable manner.

The three principles represent somewhat conflicting goals that the UTM system needs to balance. More stringent safety requirements and security measures may limit the amount of throughput in the system and raise costs for operators. NASA's approach is to hold the first principle of ensuring national and regional security above the other two which motivates, among other things, the fact that public safety and security vehicles are expected to have priority at all times in the urban airspace. The second and third principles, instead, will be balanced based on two propositions. The first is that the airspace should be structured in such a way as to allow flexibility where possible and structure where necessary. The main advantage of using the urban airspace is that it provides much more flexibility than ground transportation networks, so in the interest of the third principle traffic rules should be set only in the case where there is an imbalance between demand and capacity. Section 1.4.4 will elaborate on different airspace structures that have been investigated in the literature. The second proposition is that performance requirements should be tailored to the specific application and areas in which the operation will be conducted. Surveillance missions in rural areas, for instance, will not have the same performance requirements as delivery operations in densely populated urban areas, although no consensus exists yet as to precisely which requirements these will be.

1.4.2. UTM architecture

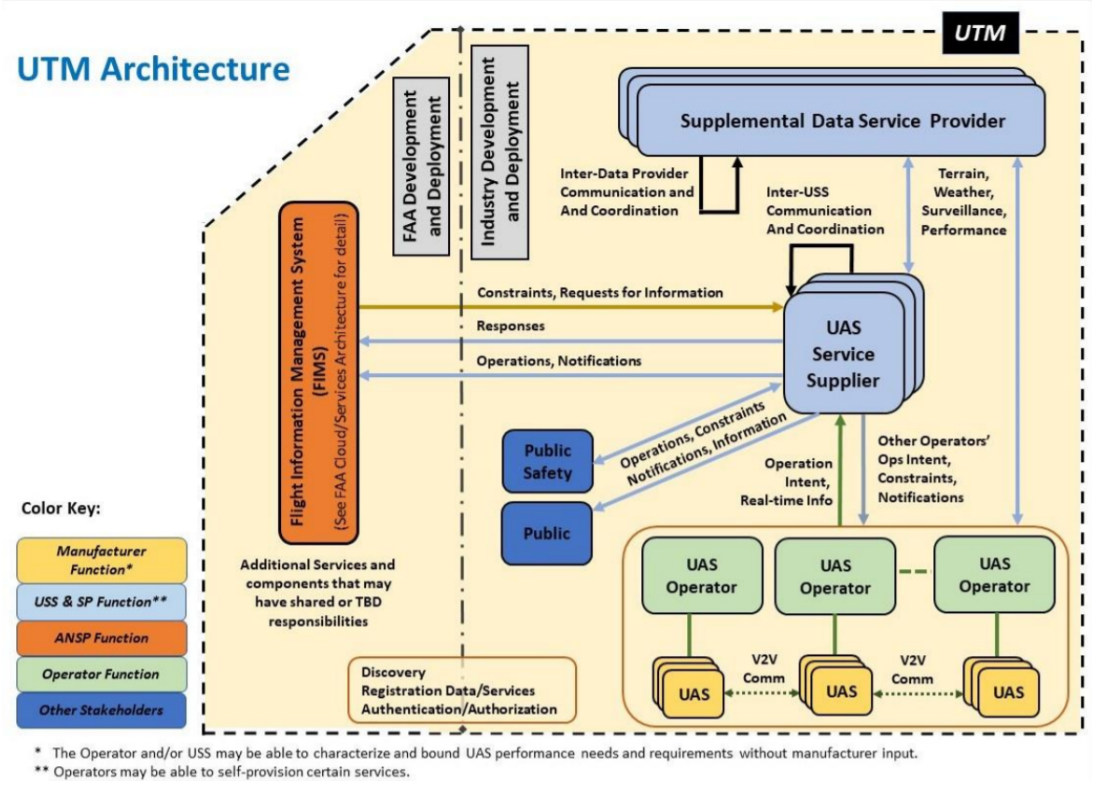


Figure 1.5: Latest UTM architecture proposed by NASA and the FAA. Taken from [52].

Figure 1.5 shows the latest proposed UTM architecture by NASA and the FAA [52]. The operations are not managed by a centralized entity as in ATC, but rather coordinated in a distributed manner by a set of actors with specific roles and responsibilities. Responsibilities are clearly split between the *air navigation service provider (ANSP)* and industry parties, the most important of which are the *operator* and the *UAS service provider (USS)*.

The ANSP's role is limited to the provision of real-time airspace constraints to the operator, who is then fully responsible of conducting its own operations and ensuring compliance with the airspace constraints. Operators are responsible for sharing flight plans among themselves and ensuring deconfliction through a distributed information network. Operators may choose to use a third-party USS to support their operations, or may choose to provide these services themselves (be their own USS). The USS acts as a link between the UTM information network and the operator, facilitating the flow of information. Information from the ANSP is aggregated with other information sources, such as other operators' flight plans, and submitted to the operator in the form of constraints and notifications. The USS also takes care of broadcasting operational information to relevant stakeholders, such as the ANSP and other USSs.

It is important to note that, within UTM, deconfliction is expected to happen in a distributed manner and

there is no provision of separation services such as in ATC. This requires operators and USSs to develop tools to facilitate coordination among UAVs and enable them to plan conflict-free operations. In order to reap the most economic benefit, these tools should be automated and require limited human supervision. For on-demand and high-density applications such as UAV delivery, this is a highly complex problem already for a single operator as discussed in Section 1.3. The proposed UTM architecture further motivates the importance of addressing the UAV delivery coordination problem, which is the focus of this thesis.

An added layer of complexity comes from the fact that operators or USSs are not only required to deconflict their own UAV fleets, but also ensure that there are no conflicts with the plans of other operators. Since operators are self-interested and may even be competing commercial entities, it is challenging to ensure coordination between them. Within NASA's UTM architecture, operators are given an order of priority (e.g. public health or security vehicles have priority over commercial vehicles), and vehicles in the same priority class are expected to negotiate among each other according to a fixed protocol to resolve conflicts. While the protocol remains to be designed, NASA has drafted several requirements and desiderata in collaboration with industry and regulators. The most important are that the protocol should be finite, accepted by all UTM actors and transparent to operators [135]. A few attempts have been made to develop suitable negotiation protocols, but work in this area remains very limited. An example is the recent work from researchers at the National Institute of Informatics (NII) [4], which proposes a sequential bilateral finite-horizon alternating offers protocol that allows operators to share costs of replanning. However, their work lacks formalism and the protocol does not guarantee pareto optimality given that it is sequential bilateral and does not capture all possible deals available to the operators. In addition, all operations are assumed to be de-conflicted pre-flight, whereas to enable on-demand applications such as deliveries the protocol should allow operators to change their plans in-flight.

1.4.3. UAV communication

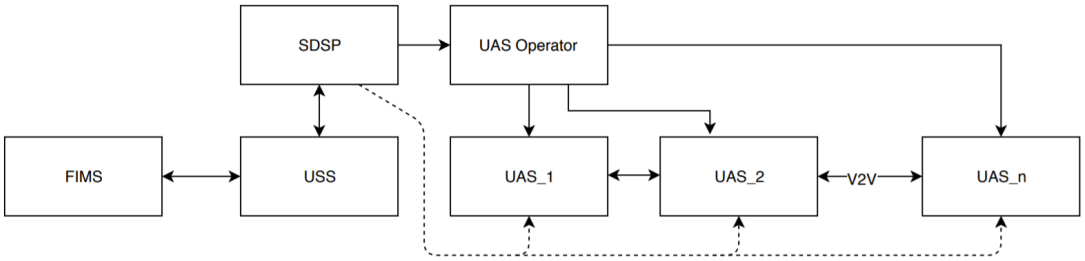


Figure 1.6: Communication protocol in NASA TCL 4 UTM architecture. Taken from [28].

An aspect of the UTM architecture that is critical to the UAV coordination problem is the communication paradigm that will exists between operators and their UAV fleet, and that UAVs will use to communicate with each other. In its TCL 4 architecture (described in section Section 1.4.2), NASA proposes the communication structure shown in Figure 1.6. Operators maintain a bidirectional communication link with all of the UAVs, allowing the operator to broadcast important information to the entire fleet. This includes airspace constraints broadcasted by the ANSP through FIMS and communicated to the operator via its USS, or weather information communicated via a supplemental data service provider. The communication link with all UAVs is also necessary for the operator to obtain telemetry data from its fleet and ensure that none of the airspace constraints mandated by the ANSP are violated. Vehicle to vehicle (V2V) communication links will also allow the UAVs to communicate between each other in order to share useful information about the environment and possibly resolve conflicts among their plans. However, which technology will be used for communication is an open question, both for operator-UAV and V2V links. Several options are being investigated in industry and academia, including direct radio links, satellite communication and cellular networks [28] [188]. Table 1.3 shows an overview of the advantages and disadvantages of the most commonly discussed approaches. The fifth-generation (5G) cellular networks are particularly promising, since they can support a peak data rate of 10 GB/s with no more than 1-ms round trip latency, which makes them suitable for delay-sensitive and high-rate UAV communication [28]. However, these services will have limited coverage in remote areas and aerial-ground interference is considered a major challenge, although promising mitigation approaches have been proposed [103]. Technology companies such as Verizon [118], Ericsson [126] and Qualcomm [1] consider UAV communication to be among the most promising use cases of their 5G technology.

Technology	Characteristics	Advantages	Disadvantages
Direct link	Direct point-to-point communication via licensed or unlicensed bands	Simple, low cost	Limited range, low data rate, vulnerable to interference, non-scalable
Satellite	Communication and internet access via satellite	Global coverage	Costly, heavy and energy-consuming equipment, high latency, large signal attenuation
Cellular network	Enabling UAV communications through cellular infrastructure and technology	Almost ubiquitous accessibility, cost- effective, superior performance and scalability	Unavailable in remote areas, and potential interference with ground communications

Table 1.3: Comparison of candidate UAV communication technologies. Taken from [28].

1.4.4. The airspace for UTM

The UTM architecture proposed by NASA and the FAA focuses on UAS operations **below 400 feet** (122m) above ground level (AGL), across both controlled and uncontrolled airspace. The majority of the operations are expected to be conducted in uncontrolled (class G) airspace, in which ATC has no responsibility to provide separation services at any altitude. In controlled airspace (class A,B,C,D,E), ATC normally provides separation services to both manned and unmanned aircraft, however this does not hold for unmanned aircraft operating below 400 ft AGL. The airspace for UTM (<400 ft AGL) is therefore fully managed according to the distributed network described in Section 1.4.2, with limited interaction with ATC in controlled airspace (e.g. in contingency scenarios). Operators are expected to be fully cooperative and to meet regulatory and performance standards at all times.

Since operators in UTM are given extensive freedom with respect to the manner in which they manage their operations, several studies have considered applying different airspace structures and traffic rules to limit the number of possible conflicts and facilitate the deconfliction process. NASA's philosophy is that these structures should only be employed if demand is high enough to make it impractical to safely achieve deconfliction in the absence of airspace structure (see Section 1.4.1). Metropolis, a research initiative funded by the European commission, has investigated the effects of several traffic concepts on UTM safety and capacity [163] in urban settings, namely:

- **Full mix**: UAVs are allowed to fly any trajectory as long as it is conflict-free and within the constraints provided by the ANSP and USS.
- **Layers**: the airspace is split into vertically stacked altitude layers. In each layer, UAVs are only allowed to fly within a specified heading range. The goal is to limit the relative velocity between UAVs and thereby reduce the probability of conflicts.
- **Zones**: the airspace is split horizontally in a manner that captures the layout of the city. Figure 1.7 shows the Zones concept as described in [163]. The authors use clockwise and counterclockwise concentric zones that mimic the function of ring roads in cities. Radial zones connect the concentric zones and facilitate traffic either towards or away from the center. Vehicles are allowed to travel at any altitude, provided that their heading follows the rules of the zone. Other concepts in the literature include denser zones for the city center that follow the layout of the streets [146], as shown in Figure 1.8.
- **Tubes**: provide fixed (bidirectional) routes in the air that are structured to fit only one UAV both vertically and horizontally. This is the most structured concept in which 4D separation of vehicles is fully enforced by the traffic rules and no further de-confliction efforts are required on part of the operator. The tubes concept can be seen as a graph structure, in which the edges represent the tubes and the nodes the intersections between them. No more than one vehicle is allowed to be on an edge (tube) at the same time. Tubes are envisioned to be combined with layers in order to form a three-dimensional lattice-like structure as shown in Figure 1.9. Greater speed is allowed at higher layers and therefore longer tubes are required.

The above concepts were tested in simulation in a fictitious urban environment of comparable size to Paris, for different demand levels [163]. Figure 1.10 shows the number of flights simulated for each demand case and the associated results in terms of number of conflicts and route efficiency. Route efficiency is defined, per flight segment, as the ratio of the shortest origin-destination distance to the actual distance travelled. The demand conditions were the same for all concepts, but for the tubes scenario less flights were simulated for all demand conditions. This is because the tubes concept did not provide enough capacity to handle all flight requests in any of the demand cases, and therefore a number of operations were rejected. Remarkably, when conflict resolution between vehicles is allowed, the zones and tubes concepts do not bring down but rather increase the number of conflicts in the airspace. It appears that the higher traffic densities that these structures generate outweigh the effect of enforcing reduced relative velocities between vehicles. In addition, these concepts

substantially reduce route efficiency, in particular the tubes concept for which, in the highest demand case, route efficiency is 1.5 times worse than for full mix traffic. The layers concept also does not provide measurable improvement in the number of conflicts compared to the full mix case, but also allows for comparable levels of route efficiency. This is because UAVs are still allowed to fly their optimal headings and inefficiencies only arise due to required altitude changes.

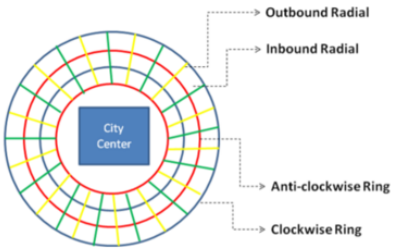


Figure 1.7: Zones concept with radials. Taken from [163].

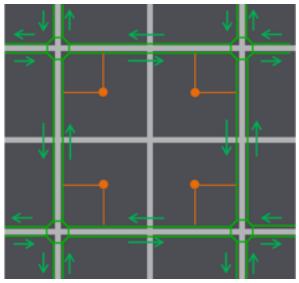


Figure 1.8: Zones concept for dense city center. Taken from [146].

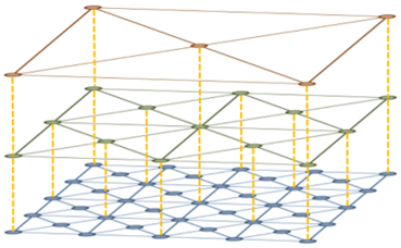


Figure 1.9: Layered tubes concept. Taken from [163].

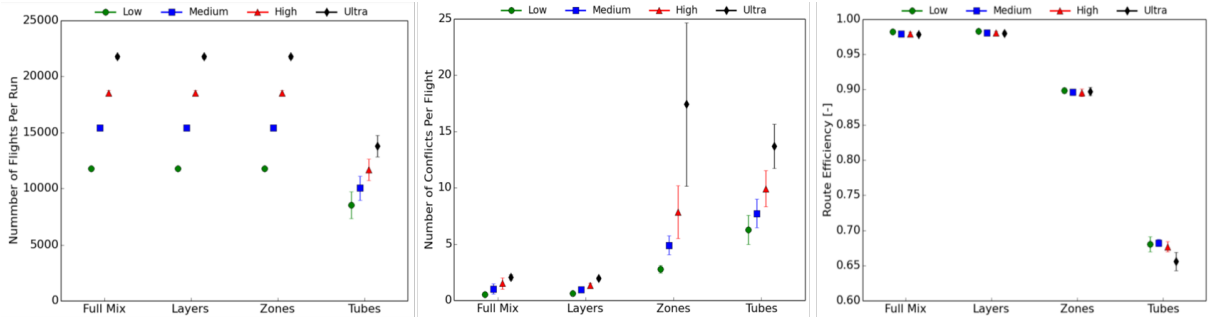


Figure 1.10: Results from the Metropolis study showing the number of flights per run (left), number of conflicts per flight (center) and route efficiency (right) for the full mix, layers, zones and tubes airspace configuration concepts. Figure taken from [163].

Overall, the results show that the studied airspace configurations do not provide improvements in the number of conflicts in the airspace and can severely compromise route efficiency, negatively impacting the value operators (and their customers) can generate from urban UAS operations. It should be noted, however, that there are other arguments for employing degrees of airspace structure, such as controlling third-party risk and ensuring public safety. It is therefore to be expected that there will be some limitations as to where UAVs are allowed to fly. From a capacity and conflict avoidance perspective, though, there seems to be no measurable advantage to employing any of the tested airspace configurations. In line with the principles of UTM (Section 1.4.1), the full mix concept should be preferred where possible and plan de-confliction should occur via the operators rather than be enforced by the airspace structure in order to maximize the value generated from urban UAS operations. This underlines the importance of developing coordination mechanisms that allow operators' fleets to plan conflict-free operations in an autonomous and decentralized manner.

1.5. The UAV delivery coordination problem

Now that we have explored the topic of UAV delivery and the broader notion of unmanned traffic management, we can elaborate on the general problem formulation that will be the basis of this thesis project, namely the UAV coordination problem for urban deliveries. As discussed in Section 1.3, multi-agent systems represent a natural manner to address this problem, so we define the problem within this framework.

We will focus on the pick-up and delivery scenario that appears in the urban delivery as a service operating model (Section 1.1.2), since from a UAV coordination perspective this is substantially more complex than the last-mile delivery problem in which there are only a few warehouses (and therefore start locations of delivery tasks) and these are fixed throughout the problem. Within the urban delivery as a service formulation, demand for deliveries originates from customers which request goods to be collected from a pickup location (which we can think of as a store or a restaurant, for instance) and delivered to their home. This constitutes a more

challenging planning problem because both the number of possible pick-up locations and customer locations can be very large. While there are several ways to attempt to define this problem, we propose a general high-level representation in which there are three types of agents [8]:

- **Operator**: The operator is a commercial entity which must use a fleet of UAVs to fulfill a certain demand for deliveries. Within the shared low-altitude airspace, there may be multiple operators each attempting to conduct their own operations. The operator is a self-interested agent with the goal of maximizing its own profit irrespective of the profit of other operators. From any given delivery executed by its UAVs, the operator incurs a cost proportional to the distance travelled (in Section 1.2 we estimate this to be $\approx$ 0.33 EUR cents per km) and generates revenue which could be a flat fee or a variable fee proportional to the value of the order, the distance travelled or a combination of these. The exact revenue and cost rules should be decided based on the planning methods used, but it is important that there exists some notion of profit and that the operator exhibits the goal of attempting to maximize this profit. Coordination between the operators, for instance for conflict resolution, should be decentralized. As discussed in Section 1.4.2, given the amount of operators and levels of density expected for commercial UAV traffic, it is highly impractical to have a single controlling entity responsible for deconflicting all operations.
- **UAV**: the UAV performs the actual task execution. It must travel to a pickup location in the environment, collect a package, and deliver it to a target location. UAVs belonging to the same operator form a *fleet*. For realistic UAV properties such as battery life, payload capacity, range and cruise speed, we can refer to the analysis made in Section 1.2. We determined that a feasible limit on the payload capability of the delivery UAV is 2.2 kg, so it is reasonable to assume that the UAVs will only be able to carry one package at a time and will therefore not be able to visit several pickup locations in sequence before visiting a drop-off location, which is important to note from a planning perspective.
- **Customer**: The customer agent is the source of demand for deliveries. It should have the capability to request the delivery of some item with a specific value from a pickup location to its own location via an operator. In a setting where there are multiple operators and these are connected to the same pickup location and could therefore both deliver the item requested by the customer, the customer should have some preference mechanism that allows it to select an operator instead of another. This could be as simple as selecting the operator that is able to deliver the item with the lowest cost, or include more complex factors such as expectations regarding the operator's reliability or timeliness.

There are several factors that make this problem more challenging than many similar planning problems explored in the multi-agent systems literature. Firstly the solution should be able to operate in a *lifelong* manner. That is, it needs to continuously accept demand for deliveries in a rolling fashion and therefore both the environment and the agents' goals change dynamically throughout the problem. Unlike traditional warehouse-type robot coordination problems, we do not envision there being fixed rest locations for each UAV agent and therefore the agents are in principle not required to follow simple rule-based approaches after the execution of a task. The problem is by nature much less structured since the environments are larger and there is more room to strategize. One could imagine, for instance, that hovering above a high-demand area would be a better strategy than heading to a charging station for a delivery UAV, given the right battery life and demand conditions. There may also be uncertainty regarding the timeliness of items at pickup locations, for instance a restaurant may take longer than declared to prepare a food order. This setting is therefore a challenging and worthwhile one to investigate, because of its importance in enabling UAV delivery and because it poses novel challenges in multi-agent coordination.

It is important to note that there are effectively two layers to this coordination problem, namely *single-operator* coordination and *multi-operator* coordination. Single-operator coordination concerns how a single operator's fleet should behave in order to most effectively serve the demand for deliveries and maximize profit, and is therefore a *cooperative* problem. Multi-operator coordination includes multiple self-interested operators who need to conduct their operations in a conflict-free manner, and therefore includes a *competitive* layer since the fleets are competing with each other. The challenge in the multi-operator case is to develop efficient and fair negotiation protocols that allow self-interested operators to deconflict their operations in real time, for use within the UTM architecture as discussed in Section 1.4.2. The two layers are therefore complementary but different in nature. In this project, we will focus on the **single-operator (cooperative) coordination** case, since it is deemed more urgent to enable UAV delivery. It is of paramount importance to provide coordination tools for operators to incentivize them to make economical use of UTM services. The availability of such tools will also allow more accurate modeling of delivery operations and render the evaluation of negotiation protocols

[8] Note that this is a high-level formulation only, intended to understand which literature to investigate. A more detailed representation of the problem could involve more types of agents or different characteristics. For instance, stores or restaurants could be represented as agents if we want to incorporate characteristics such as preparation times, delays, etc.

for deconfliction between multiple operators easier. The latter is a natural problem to look at once coordination tools for the operators have been developed, and is therefore left as a possible extension to the project or direction for future work.

1.5.1. Relevant research areas

Based on the above problem formulation, we identify two highly relevant areas of the multi-agent systems literature.

- **Chapter 2 - Multi-agent task allocation:** An important requirement of the UAV fleet is that it should be able to perform task allocation, allocating different UAVs to pickup and delivery tasks in an effective manner and in real-time as the tasks enter the problem, which is highly challenging. The task allocation routine should also be able to operate in dynamic environments, allow for flexibility in the team objective and efficient replanning as new information regarding the environment or the tasks is discovered by the fleet. We investigate several techniques applicable to this problem, spanning a number of different fields such as operations research, game theory and distributed artificial intelligence.
- **Chapter 3 - Multi-agent path finding:** In order to effectively carry out pickup and delivery tasks, the UAVs need to efficiently plan low-cost paths to the pickup and delivery locations while avoiding conflicts with other agents and obstacles in the environment. It is therefore essential to review methods tailored to this task. Several useful techniques for multi-agent path finding, which is a well-known NP-hard problem in distributed artificial intelligence, exist in the multi-agent systems literature. These methods are concerned with finding conflict-free paths for a set of agents which typically attempt to minimize a system-level cost function. The majority of these techniques are designed for small, static and dense environments such as warehouses, but several have been successfully extended to larger and more challenging environments and applied to problems involving UAVs.

2
Multi-agent task allocation

Task allocation is a well-studied problem which consists in feasibly assigning a set of tasks to a team of agents while attempting to maximize a global objective function that is representative of system performance [43]. In this chapter, we review a number of techniques applicable to the task allocation problem and discuss the most applicable ones for the UAV delivery coordination case in more detail. We begin by providing a taxonomical overview of multi-agent task allocation (MATA) problems in Section 2.1 and in Section 2.2 we provide an overview of relevant classes of MATA solvers. In Section 2.3 and Section 2.4 we discuss prominent approaches in each class. We then perform a comparative evaluation of the solvers in Section 2.5 and identify the most suitable class of approaches for the UAV delivery coordination problem, which is discussed in further detail in the rest of this chapter.

2.1. Taxonomy of task allocation problems

The task allocation problem is ubiquitous in a number of real-life applications and a large amount of variants to the problem have therefore been studied. Gerkey [58] proposes the following three axes to classify multi-agent task allocation problems:

- **single-task agents (ST)** vs **multi-task agents (MT)**: in ST problems, each agent is only able to execute a single task at a time, whereas in MT problems, agents are able to execute more tasks at once.
- **single-agent tasks (SA)** vs **multi-agent tasks (MA)**: in SA problems, each task is to be executed by a single agent whereas in MA tasks can require multiple agents in order to be executed.
- **instantaneous assignment (IA)** vs **time-extended assignment (TA)**: the IA problem formulation only allows agents to reason about a single task at a time and the assignment is therefore instantaneous. In TA problems, agents are allowed to plan ahead and consider future allocations, either because information about future tasks is available or because they have the capability to reason probabilistically about which tasks will enter the system in the future.

Given the payload considerations discussed in Section 1.2, we assume that a UAV can only carry one package at a time, and the problem is therefore characterized by single-task agents (ST). It also safe to assume that, in flight, a single package should be carried by no more than one UAV. However, recent concepts of operations have introduced the idea of multi-modal deliveries, examples of which include the use of multiple UAVs with different safety specifications to sequentially carry the same package across different areas until its destination, or the combined use of UAVs and delivery trucks [176]. In these cases, our SA/MA classification of the problems hinges on the manner in which a task is defined. While multi-modal problems can be formulated in a MA manner, they can also be formulated in a SA manner by using a more granular definition of task which simply involves the act of carrying the package across a set path, without necessarily delivering it to the end customer. The delivery of an item to the customer would then involve the successful completion of multiple of these more granularly defined tasks. Along Gerkey's temporal axis, both the IA and TA formulations could be applicable to the UAV delivery problem depending on the objective we are considering. Suppose we are concerned with allocating delivery tasks in such a way as to minimize the time the customer needs to wait for his order and that customers are prioritized based on the time at which they submit their order. That is, customers who order first are to be served first. In this case, we could simply queue the tasks based on the order in which they entered the system and allocate a single task at a time to the most suitable UAV at the time of allocation. Instantaneous assignment (IA) would be a suitable framework for this objective. If, however, we are interested in addressing more complex and perhaps more realistic objectives, such as minimizing the total distance travelled under delivery time constraints for specific customers, then UAVs need the ability to plan across sequences of tasks and maintain a schedule. For this case, a time-extended assignment formulation is most suitable (TA). We therefore focus on solution techniques that apply to the ST-SA/MA-IA/TA variants of the task allocation problem, and use these notions to steer our analysis of the solvers.

2.2. Overview of solution techniques

Since the multi-agent task allocation problem is present in a variety of settings, the available solution techniques find their roots in a number of different disciplines, such as operations research, artificial intelligence, multi-agent systems and game theory. In Figure 2.1 we map the most prominent classes of solution techniques for MATA problems including examples of algorithms from each class, and categorize them in terms of the degree of centralization in the planning and control dimensions. Planning refers to the process of *computing* a solution to the task allocation problem. In centralized planning approaches the entire computation is performed by a single entity and therefore runs on a single machine, whereas in distributed approaches each agent solves a part of the planning problem locally. The control axis here refers to the degree of centralization in *decision-making* for task allocation. In techniques featuring centralized control, a single entity decides on the allocation for the entire system and broadcasts it to all agents. Under a distributed control architecture, each agent makes a part of the decision regarding the system-wide allocation, for instance which tasks to assign to itself. The global allocation therefore emerges from a series of local decisions. We elaborate further on the characteristics of these approaches and provide an overview of the state of the art in all relevant classes of solvers, discussing *centralized* planning techniques in Section 2.3 and *distributed* planning techniques in Section 2.4.

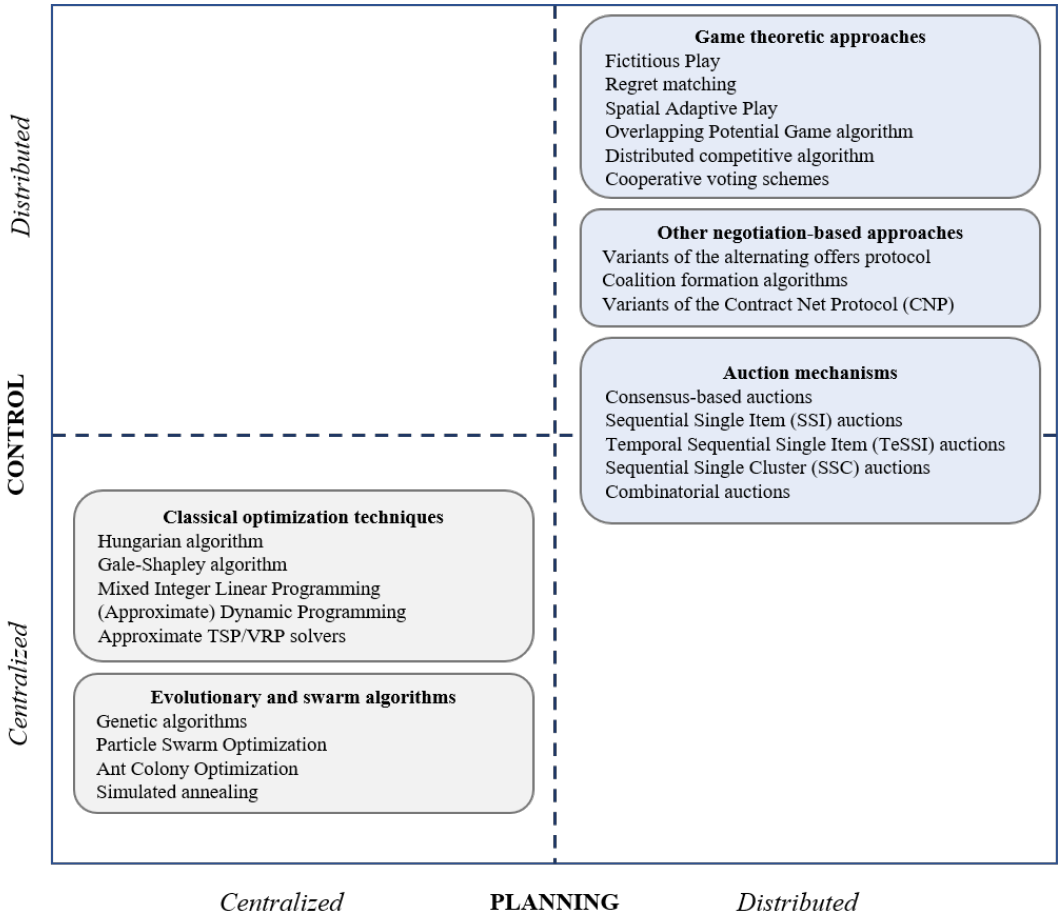


Figure 2.1: Overview of main classes of MATA solution techniques.

2.3. Centralized planning techniques

These techniques require a centralized planner to compute an allocation that maximizes the system objective, and broadcast such allocation to all agents to allow them to adjust their local plans accordingly. They therefore impose a centralized control paradigm on the system, in which every agent needs to be able to communicate with the central planner. While centralizing the planning procedure leads to optimality guarantees for most problems, it also leads to low robustness since the centralized controller represents a single point of failure in the mission. Scalability is also a point of concern, since these methods are computationally intensive and optimal solutions may be intractable for large problems, which motivates the use of approximate solution methods instead. Among centralized planning techniques, we distinguish between *classical optimization*

approaches from the operations research literature and *evolutionary and swarm algorithms* which have more recently emerged as alternatives to solve discrete optimization problems approximately.

2.3.1. Classical optimization approaches

In the one-shot ST-SA-IA case, multi-agent task allocation can be posed as an *Optimal Assignment Problem* (OAP), which is a well studied problem in operations research. Suppose there are r robots and t tasks, and that the centralized planner can produce system-level utility estimates for each of the rt possible task-robot assignments. Then the planner can determine the optimal outcome in $O(r\,t^2)$ time through an LP-based matching algorithm such as the Kuhn-Munkres Hungarian algorithm [88]. Liu et al. provide an improved version of the Hungarian algorithm that accounts for uncertainties in the assignments' utility estimates and use it to solve ST-SA-IA multi-agent routing problems with localization error [94]. Several other variants of the Hungarian algorithm aimed at solving more realistic instances of assignment problems have been proposed, such as a dynamic variant which can efficiently recalculate optimal assignments when a subset of robot-task utilities changes [107]. Roldan et al. deviate from the pure system-level view and formulate a task allocation problem for competing teams of UAVs as a centralized stable marriage problem and use the Gale-Shapley algorithm to solve it [138]. While approaches based on matching algorithms have been shown to scale well to large amounts of robots and task, they only solve one-shot IA cases and are not easily generalizable to the lifelong [1] or TA settings. They also rely on accurate estimates of system utility for every robot-task combination, which may be unavailable or computationally expensive to produce.

Examples of problems in the operations research literature that tackle more general and relevant variations of the task allocation problem include dynamic formulations of the NP-hard Travelling Salesman Problem (TSP) and its generalization, the Vehicle Routing Problem (VRP). Several variants of these combinatorial optimization problems have been addressed in the operations research literature, using exact methods such as Branch-and-Bound, Constraint Satisfaction and Dynamic Programming [120]. However, solutions to large problems are intractable and only optimal in the objective function, which is often based on approximations of the underlying model. A common example in routing problems is the use of heuristics to estimate the length of the optimal path to a given target. This realization has led to extensive work on solving the VRP using approximate instead of exact methods. In particular, a recent branch of the literature has introduced variants of the VRP aimed specifically at addressing challenges in UAV delivery, such as the TSP with sidekick and the VRP with Drone (VRPD), in which a fleet of trucks equipped with UAVs needs to deliver packages to customers [83] [7]. Schermer et al. solve the VRPD with a two-stage-heuristic [145], which at first ignores the UAVs and creates efficient VRP tours for the trucks, and in a second stage inserts the UAVs into the existing tours. A similar heuristic approach is proposed by de Freitas et al. [37] who first generate solutions for the fleet of trucks, and then apply a general variable neighborhood search (GVNS) procedure to incorporate the UAVs into the VRP. Ulmer et al. instead address the situation in which *either* a UAV or a truck need to execute a delivery, and propose a policy function approximation based on the geographical clustering of tasks to decide wheter to dispatch a UAV or a truck [176]. Collectively, centralized solvers for routing problems involving UAVs show that these can provide measurable benefits to traditional delivery vehicles such as trucks. They are also useful for feasibility studies and fleet sizing efforts since they can help understand the operational conditions in which UAVs provide the most benefit [174]. However, they are hardly suitable as online UAV coordination paradigms. They suffer from poor scalability, are computationally intractable for large problems and require the central controller to constantly communicate plans with the entire fleet.

2.3.2. Evolutionary and swarm algorithms

Efforts to solve TSP and VRP problems approximately have also relied on evolutionary and swarm algorithms to reduce the computational time. Ha et al. [62] introduce a novel Hybrid Genetic Algorithm (HGA) to solve the VRPD problem under both the total operational cost and total completion time objectives. The HGA incorporates crossover, local search operators and an adaptive penalization mechanism tailored specifically to the VRPD problem that lead to substantially higher solution quality than other variants of the GA. A series of other studies develop tailored GA variants to account for factors such as routing under varying wind conditions [98]. Jiang et al. solve an offline VRP with time windows (VRPTW) in which a swarm of UAVs needs to serve customers within pre-defined intervals using a Particle Swarm Optimization approach, which is shown to converge faster and lead to lower cost solutions than a GA for the same problem [76]. The Ant Task Allocation (ATA) algorithm proposed by Du et al. represents one of the few attempts to develop a swarm optimization approach to solve more general MATA instances, which is a variant of the Ant Colony Optimization algorithm

[1] One exception to this is CENTRAL [99], a lifelong task allocation and path finding solver that performs optimal task assignment via the Hungarian algorithm every time a task enters the system.

that uses the task selection model of honeybees [46]. However, despite the fact that testing is limited to simple problems in very small grid-like environments, the algorithm still requires several thousands of iterations for the allocation to stabilize. While some techniques within this class provide improvements in computational time compared to exact solvers, most of them are still deemed computationally intractable for real-time use in dynamic problems such as UAV delivery coordination. Moreover, they inherit the aforementioned communication and robustness issues that are common to all centralized planning approaches.

2.4. Distributed planning techniques

With a distributed planning architecture, the computational burden of solving the planning problem is shared among the agents. Each agent uses its own computational resources and memory to solve a part of the planning problem, and agents may have access to communication channels to share relevant information throughout the process. Agents' plans are typically coupled, and decisions are based on local information, so among the main challenges of distributed planning is to design protocols that capture this coupling and possible inconsistencies in situational awareness in such a way as to lead to good system performance. Classes of distributed planning techniques differ in the protocols they impose on the agents and in the extent to which the planning workload is distributed. Compared to centralized approaches, these classes of techniques are much more robust and scalable since they can operate under different network structures and can avoid the presence of a single point of failure in the planning process [2]. However, it is clearly more difficult to guarantee optimality, particularly when limited information can be shared and real-time performance is important. Among distributed planning techniques suitable for the MATA problem, we distinguish between *auction mechanisms*, *game theoretic approaches* and *other negotiation-based approaches*.

2.4.1. Auction mechanisms

Auctions are promising mechanisms to solve the task allocation problem while conducting the vast majority of the computation in a distributed manner [84]. They have been studied extensively in the artificial intelligence and multi-agent systems literature since the early work on distributed coordination via contract nets [155], and have been applied to several allocation problems. Examples include the allocation of tasks to fire brigades and ambulance teams in RoboCup Rescue simulations [108], computational resources and workflows to CPUs in grid computing [132], search locations to exploration rovers [173], and targets to UAVs in a variety of routing problems [30]. Auction-based task allocation mechanisms are computationally efficient and suitable for online use in dynamic problems, as demonstrated by several hardware tests [78]. They also typically scale well to problems with large robot teams and are flexible to a variety of team objectives.

An important note is that auction mechanisms used for task allocation are usually *cooperative*. The most common formulation is that agents bid an estimate of their cost of executing a task and the agent with the lowest bid is assigned the task. The problem is in establishing bidding rules and task scheduling paradigms local to the agents that lead to the optimization of the desired system objective. Several of the issues and challenges that arise in competitive auction settings studied in the economics literature, such as collusion and other forms of strategic behavior, do not appear in the multi-agent coordination case. However, several notions from the economic theory of auctions can still prove useful in a cooperative setting. The concept of individual rationality is a good example. In a commercial setting such as a UAV delivery network where the operation needs to be profitable, it may be necessary for UAVs to reason about the profitability of tasks. Individual rationality can then be exploited to ensure that UAVs only execute tasks which are profitable for them and therefore for the network.

In most auction variants, the agents eligible for the tasks act as bidders and there is an auctioneer which is responsible for determining a winner and allocating the tasks. The role of the auctioneer may be a central agent or may be varied throughout the problem, for instance through the passing of a token. A popular class of auctions in this category are *Sequential Single Item (SSI)* auctions, in which tasks are allocated in multiple rounds [84]. A vast amount of SSI variants with different properties and performance guarantees exist in the literature, accounting for factors such as task swaps [190], rollouts [191] and regret clearing [86]. A recent mechanism called TeSSI [112] extends the SSI auction to efficiently handle temporally constrained tasks, allowing individual agents to plan across sequences of tasks with time windows by representing their schedule as a simple temporal network (STN). Combinatorial auctions have also been used to assign targets to teams of exploration robots [14]. In contrast to the SSI case, tasks here are auctioned all at once and therefore all synergies between tasks are accounted for, guaranteeing optimality. However, combinatorial auctions are not applicable to large

[2] Auctions that require an auctioneer can be seen as an exception, given that the auctioneer is essential to the success of the planning procedure. However, in most cases the role of the auctioneer can be varied dynamically throughout the problem to avoid the presence of a single point of failure throughout the mission.

online problems because the computational cost and communication overhead is exponential in the number of tasks [84]. Sequential single cluster auctions [68] [69] are a promising middle ground, as they allow agents to bid on subsets of tasks. When the clusters are intelligently determined, most of the synergies among tasks can be captured while reducing the communication overhead by several orders of magnitude.

In the presence of communication constraints, such as limited range or bandwidth, it may be impractical or impossible for all agents to communicate with the auctioneer. A common solution to this is to sacrifice system performance and conduct the auction only within the neighborhood of the auctioneer or among a select group of agents. *Consensus-based auctions*, instead, use ideas from the literature on distributed consensus algorithms [113] to eliminate the need for an auctioneer entirely and allow agents to converge on a conflict-free assignment independently. If the agents' task evaluation function is defined appropriately, consensus-based auctions provide convergence guarantees and provably good performance bounds under realistic network structures. The Consensus Based Auction Algorithm (CBAA) and the Consensus Based Bundle Algorithm (CBBA), its multi-assignment extension, were first introduced by Brunet et al. [20] to solve static allocation problems and represent the algorithmic basis for this class of auctions. Several extensions to the CBBA have been developed to make it applicable to a wider range of allocation problems. Most notable extensions include improved agent evaluation functions to handle time-sensitive tasks [120], the inclusion of relay tasks to prevent network disconnects [120] and the notion of partial replanning to improve online performance [22].

2.4.2. Game theoretic approaches

An alternative framework to address the task allocation problem is by formulating it as a game between agents and their neighbors and using methods from game theory to solve it. Agents are treated as independent decision-makers that attempt to maximize their own utility given their local knowledge and expectations of other agents and the environment. While game theory has been applied extensively to non-cooperative settings such as multi-agent patrolling [70], in recent work it has also been applied to cooperative MATA settings such as surveillance [38] and mapping [137] missions. The rationale is that by carefully designing protocols and agent-specific utility functions that align with the global system objective, cooperation can be enforced in a distributed manner and with relatively little communication overhead.

A number of learning techniques from the literature on multi-player games have been adopted as negotiation mechanisms for task allocation problems. These include variants of fictitious play and regret matching [10] in which agents do not have knowledge about their peers' utility functions, in order to capture the lack of global situational awareness that characterizes MATA problems where agents do not necessarily have the same information about the environment or even the tasks to be allocated. The main issue with these approaches is that, while they guarantee convergence to a stable and conflict-free allocation (an allocation which in game theoretical terms constitutes a Pure Strategy Nash Equilibrium (PNSE)), they provide no performance guarantees and are empirically shown to produce allocations which are far from optimal. An algorithm that maintains the same convergence properties, but also provides probabilistic performance guarantees is Selective Spatial Adaptive Play (SSAP) [10], which can be tuned to generate near-optimal allocations at the expense of long conversion times. However, the SSAP approach is only suitable for one-shot SA problems, and was shown to require over 1,500 negotiation steps to converge to a provably good allocation in a large-scale assignment problem, which constitutes an impractically large communication overhead to deploy it in an online setting. Chapman et al. address the dynamic TA problem by formulating it as a Markov game, approximating it as a series of static potential games to ensure the tractability of equilibria, and then solving these games using the Distributed Stochastic Algorithm [29]. Empirical results for a search and rescue scenario are promising, but there are no performance guarantees and highly suboptimal allocations are achieved in cases with low communication ranges.

The recent work of Roldan et al. proposes a distributed competitive and a hybrid cooperative approach to solve one-shot allocation problems in swarm exploration and mapping [137]. In the competitive approach, the problem is formulated as a set of games between agents and their neighbors in which the agents' strategies are their task selections and their payoffs consist of the negative of their cost of performing the task. Agents search the best Nash Equilibrium (NE) and select that as the allocation for the neighborhood, with the global allocation simply consisting of the union over all neighborhoods. The major drawback of this approach is that it requires agents to have knowledge about each others' utility functions in order to ensure convergence to the same NE and therefore a conflict-free assignment, which can be highly impractical in an online setting since agents would need to know their peers' schedules with no uncertainty. In the cooperative algorithm, each neighborhood consists of a leader and several citizen agents, which express their preferred allocation for *all* tasks in the neighborhood (also specifying which task each of the neighbors should be allocated) and cast them as votes to the leader who determines the allocation via a Borda counting rule. This approach suffers

the same limitation as the competitive algorithm, in that every agent needs to be aware of its neighbors' utility functions, while also requiring that all agents be connected to the leader.

2.4.3. Other negotiation-based approaches

Other techniques from the multi-agent negotiation literature have also been applied to the task allocation problem. Constrained coalition formation algorithms, first introduced by Shehory and Kraus [151], have been applied extensively to problems in which different capabilities and therefore multiple agents are required to complete a task (problems with MA tasks according to the taxonomy in Section 2.1) [127] [128]. In recent work, Capezzuto et al. propose an anytime coalition formation framework for temporally constrained MA tasks which, despite its lack of performance guarantees, is empirically shown to be superior to previous approaches in this class in terms of both solution quality and efficiency in a range of one-shot search and rescue simulations. Variants of the Contract Net Protocol (CNP) [155] have also been applied to relatively unstructured MATA problems. The CNP is a particularly useful framework for situations in which task discovery occurs in a distributed manner and in which tasks are complex and may require specific capabilities or multiple, differentiated roles [162] [189]. Sujit et al. propose an alternative negotiation-based approach based on a variant of Rubinstein's alternating offers protocol [140] to perform distributed target allocation in a cooperative UAV search and attack mission with connectivity restrictions. Within their neighborhood, agents make proposals regarding the tasks they would like allocated to themselves and proceed to execute the tasks if they receive consensus from their neighbors, who will reject proposals if they would generate higher utility from performing the task themselves. Note that this process is similar to the consensus phase in CBBA, with the exception that consensus is only sought within the agents' neighborhoods and not across the entire fleet, which reduces the communication overhead but leads to higher cost allocations. Other variants of the alternating offers protocol have been adopted to solve task allocation problems involving heterogeneous UAV fleets in domains such as aerial mapping [139] and surveillance [177].

2.5. Comparison of solver classes

We now make a general comparison between the relevant solver classes, based on the strengths and weaknesses of state of the art approaches already discussed in Section 2.3 and Section 2.4. In order to identify the class of techniques that is most suitable for the UAV delivery coordination problem, each class is scored on the following equally weighted criteria, which are given a score from 1 to 3 (roughly equivalent to "Low", "Medium" and "High" in comparative terms):

- **Robustness:** relates to the ability of the planning paradigm to cope with uncertainty, constraints and possible sources of failure that characterize real-life missions. Examples include faulty communication channels or failure of a UAV.
- **Scalability:** measures the extent to which the approach is able to retain its computational efficiency as the number of agents, tasks, and the hardness of the constraints increases.
- **Solution quality:** reflects the overall cost of the allocation typically obtained by algorithms in the class.
- **Computational efficiency:** reflects the running time required for the algorithms to return a conflict-free solution, which is related to the time complexity of the approach and the degree of parallelization that it allows.
- **Flexibility:** is a measure of how simple it is to modify the planning framework to address varying team objectives, agent characteristics or task constraints.
- **Suitability for online use:** measures the ability of the approach to cope in real-time with the discovery of new tasks and to handle changes in the environment or in the characteristics of already allocated tasks, such as changing start times or pickup locations.
- **Maturity:** reflects the extent to which the techniques have been tested in the context of UAV coordination. For this criterion, the scores have the following meaning: (1) simulation testing limited to small (no more than 30 agents or tasks) or offline problems and few application areas, (2) extensive simulation testing including large-scale online problems across several application areas, (3) meets the requirements of 2 and has additionally been implemented and tested on real hardware.

Based on the analysis, auction mechanisms are deemed the most suitable task allocation framework for the UAV delivery coordination problem. They exhibit strong robustness since they can be designed to operate under different network structures, and are highly flexible in the team objective since the agents' bidding rule can be varied without changing the structure of the allocation mechanism. Auctions also offer a tractable manner to control the tradeoff between solution quality and efficiency. Enhancements such as task swaps or global replanning procedures can be leveraged to increase the allocation quality at the expense of higher communication overhead, and planning problems on the agent level can be solved to different degrees of optimality. In the

	Centralized planning		Distributed planning		
	Classical optimization approaches	Evolutionary and swarm algorithms	Auction mechanisms	Game theoretic approaches	Other negotiation -based approaches
Robustness	1	1	3	3	3
Scalability	1	2	3	3	3
Solution quality	3	2	2	2	2
Computational efficiency	1	1	3	2	2
Flexibility	2	2	3	2	2
Suitability for online use	1	1	3	2	3
Maturity	3	2	3	1	3
Total	**12**	**11**	**20**	**15**	**18**

Table 2.1: Comparative evaluation of relevant MATA solver classes. For a description of state of the art solvers in each class see Section 2.3 and Section 2.4.

UAV delivery case, for instance, optimizing a single agent's schedule involves solving a TSP, which can be done optimally or in a heuristic manner. They are also suitable for online use and have been applied extensively to target assignment in dynamic multi-agent routing problems. Several state of the art auction approaches have been used in problems involving UAV fleets and tested extensively both in simulation and on hardware in indoor environments with promising results. Negotiation-based approaches are a close second to auction mechanisms, since they exhibit similarly favorable robustness and scalability properties and are also highly suitable for online use. In particular, we recognize the value of the alternating offers protocol as a tool to align the agents' incentives given limited information about each others' plans. We deem it a valuable paradigm to possibly extend an auction mechanism to allow for peer-to-peer task exchanges, and therefore discuss its theoretical foundations in Appendix A.

The focus of the rest of this chapter is on investigating further auction mechanisms for cooperative task allocation problems. We elaborate on state of the art variants of the SSI auction (Section 2.6), and its temporal extension TeSSI (Section 2.7). We do not discuss pure combinatorial auctions due to the aforementioned scalability limitations, but rather focus on sequential single cluster auctions (Section 2.8) which are vastly more efficient and can guarantee similar performance with appropriate clustering procedures. Finally, we discuss consensus-based auctions (Section 2.9) which leverage a distributed consensus phase to remove the need for an auctioneer entirely. This is a powerful representation because it is suitable for problems with communication range constraints, which can arise if global (e.g. cellular) communication is lost or unavailable and UAVs need to communicate via direct links.

2.6. Sequential single-item auctions (SSI)

In Sequential Single Item (SSI) auctions, targets are allocated independently in multiple rounds. In each round, the agents place bids on the targets that are yet to be allocated, and the auctioneer then allocates a single target to a single winning agent [84]. SSI auctions have been applied extensively to the multi-agent task allocation problem, and their efficiency has been demonstrated experimentally in early studies [42]. Lagoudakis et al. [89], however, were the first to conduct a theoretical analysis of the performance of SSI auctions for routing problems with varying team objectives. More specifically, they define *multi-agent routing* as an allocation problem that consists of a set of agents $A = \{a_1, ..., a_n\}$, a set of target locations $T = \{t_1, ..., t_m\}$ and a strictly positive and symmetric[3] cost function $c(i, j)$ that specifies the cost of travelling between any two locations i and j. The goal is to find an assignment of agents to target locations that minimizes some team objective. Let $P = \{P_1, .., P_n\}$ be a partition of T that specifies the allocation at the end of the auction, in which P_i is the set of targets allocated to agent a_i. Lagoudakis et al. study the following simple team objectives [89]:

- `MINISUM`: $\min_P \sum_j \text{APC}(a_j, P_j)$. Where $\text{APC}(r_j, P_j)$ is the minimum *agent path cost* incurred by agent a_i by traveling to all target locations in P_j, starting from its current location. The objective is therefore to minimize the sum of path costs across all agents[4].
- `MINIMAX`: $\min_P \max_j \text{APC}(a_j, P_j)$. The objective here is to minimize the largest among all the agents' path costs[5].

[3] The symmetry assumption is common in routing problems, and implies that the cost of travelling among any two locations in the environment does not depend on the direction of travel. In real-life environments for UAV routing, however, we may encounter factors that invalidate this assumption, such as wind.

[4] In the path finding literature, this is commonly known as the sum-of-costs objective.

[5] In the path finding literature, this is commonly known as the makespan objective.

Bidding rule	**Team objective**					
	MINISUM		MINIMAX		MINIAVE	
	lower	upper	lower	upper	lower	upper
BID-MINISUM	1.5	2	n	$2n$	$\frac{m+1}{2}$	$2m$
BID-MINIMAX	n	$2n$	$\frac{n+1}{2}$	$2n$	$\Omega(m^{1/3})$	$2m$
BID-MINIAVE	m	$2m^2$	$\frac{n+1}{2}$	$2m^2 n$	$\Omega(m^{1/3})$	$2m^2$

Table 2.2: Bounds on the performance ratio obtained when using an SSI auction to solve the allocation problem in multi-agent routing with n agents and m targets, with different team objectives and bidding rules. Results assume the optimal bid for each rule is approximated using the cheapest insertion heuristic. Table adapted from [89].

- MINIAVE: $\min_P \frac{1}{m}\sum_j \text{CTPC}(r_j, P_j)$. Where $\text{CTPC}(r_j, P_j)$ is the minimum *cumulative target path cost* of all locations in P_j, assuming agent a_j starts from its current location and visits all targets in P_j. The objective is to minimize the average target path cost across all the agents' targets.

The following generic bidding rule can be applied to translate the above team objectives into suitable agent bids [89]. Let $S = \{S_1, ..., S_n\}$ be the allocation at the beginning of some round r in the auction, where S_i holds the subset of tasks allocated to agent a_i. Let us now begin the round and put task t on auction. Each agent a_i should bid the marginal increase in team objective that would arise if he was allocated task t. We are therefore proceeding in a hill-climbing fashion, in which we attempt to allocate each task to the agent that generates the lowest disruption to the team objective. This will result in a high quality, but not necessarily optimal allocation given that we cannot take into account all synergies between tasks due to the sequential nature of the auction. Applying this bidding logic results in the following bidding rules for the above three objectives:

- BID-MINISUM: $\text{APC}(a_i, S_i \bigcup t) - \text{APC}(a_i, S_i)$. Agent a_i should bid the difference in the path cost that he incurs in adding task t to its plan.
- BID-MINIMAX: $\text{APC}(a_i, S_i \bigcup t)$. Agent a_i should simply bid his new path cost. It is not necessary to subtract the maximum agent path cost before the allocation of t. Since it is a system-level variable and is the same for all agents, it would be a constant subtracted from all bids, which does not affect the winner determination[6].
- BID-MINIAVE: $\text{CTPC}(a_i, S_i \bigcup t) - \text{CTPC}(a_i, S_i)$. Agent a_i should bid the increase in the cumulative target path cost that he incurs if task t is allocated to him. The $1/m$ factor can be dropped since it multiplies all bids and therefore does not affect the winner determination[7].

Every agent a_i represents its plan as a path through all targets in S_i. Computing APC and CTPC requires determining an optimal path through the targets and therefore solving a TSP, which is NP-hard. Lagoudakis et al. [89] suggest to use a heuristic approach to determine such a path. They propose the cheapest insertion heuristic, whereby agent a_i checks all possible insertions of the new task t into its original path, and then chooses the insertion that leads to the lowest cost increase. Using this heuristic, the authors provide a theoretical assessment of the performance achieved by using the above bidding rules for the three different team objectives. The results are summarized in Table 2.2, which shows the bounds on the ratio between the cost of the allocation obtained via the different bidding rules and the cost of the optimal allocation. For all objectives, there are tractable performance guarantees. The bounds on the MINIMAX and MINIAVE scale with the number of robots and the number of targets respectively, whereas the MINISUM objective addressed via BID-MINISUM leads to performance bounds which do not vary with the size of the allocation problem. In principle, the upper bounds in Table 2.2 still hold for any method of approximating the optimal path through all targets as part of the bid evaluation procedure, as long as the approximation is no worse than that generated via the cheapest insertion heuristic.

Several improvements to the basic SSI auction have been made to generate higher quality solutions while still retaining a degree of efficiency suitable for online task allocation. Among the most important improvements are *rollouts* and the *K-swaps procedure*, the latter being a framework that was first formalized in the context of SSI auctions but is in principle applicable to any kind of sequential or online task allocation mechanism.

[6] This shows a key difference with auctions in the economics literature, in which the price paid is important. Here, we are in a fully cooperative setting and are only concerned with the winner determination. Subtracting a constant from all bids or scaling all bids by a constant factor makes no difference in the allocation.

[7] See footnote 6.

2.6.1. SSI with rollouts

Zheng et al. [191] are the first to introduce the idea of *rollouts*[8] in SSI auctions, which modify the standard SSI auction as follows. Let us suppose a task t is being auctioned and that again the vector $S = \{S_1, ..., S_n\}$ contains the tasks already allocated to the agents and is known to all agents before the round begins. Each agent behaves according to the specified bidding rule, but considering a *complete* rather than a partial target allocation. In computing its bid, agent a_i considers the allocation that would arise if it was allocated t in addition to all targets in S_i, and then completes this allocation by the same hill-climbing logic to arrive at the complete target allocation (of all m targets) that would follow if it was allocated task t. The agent then uses the cost of this resulting complete allocation to compute its bid for task t. We are therefore allowing the agents to reason about the consequences of the allocation of t on the allocations that will occur in future rounds of the SSI auction. This allows us to capture some, albeit not all, of the synergies between tasks and improves the overall quality of the allocation. Rollouts are most beneficial in the first rounds of the SSI auction, since there are a large amount of targets still to be allocated and which will be affected by the winner determination of the current round. The drawback of rollouts is that they increase the number of rounds in the SSI auction, since the agents need to simulate several rounds of the auction in order to compute their bid for a single round. Solutions to this problem include only performing rollouts in the first rounds of the auction, or sampling a subset of all possible rollouts and performing only these at each round.

2.6.2. The K-swaps procedure

In any kind of sequential allocation method such as SSI auctions, swaps are necessary to capture synergies between tasks that have already been allocated and auctions that are allocated in a later round. The same is true for online settings, such as UAV delivery, in which new tasks enter the system continuously and need to be allocated in real time. Zheng et al. [190] develop a general distributed framework to re-allocate tasks to cooperative agents in order to decrease team cost, based on K-swaps, a novel contract type that specifies task exchanges between multiple agents.

In order to understand the swapping procedure and its properties, we must first summarize Zheng et al.'s formalization of the K-swaps contract [190]. Suppose that the vector $P = \{P_1, ..., P_n\}$ contains the task allocation before any swaps among agents occur, and let $P' = \{P'_1, ..., P'_n\}$ denote the allocation after all swaps have occurred. We now define three types of swap operations that agents can conduct. An *out swap* occurs if an agent a_i transfers a task t to a different agent and is represented by the vector $(a_i, -, t, -)$. An *in swap* occurs $(a_i, -, -, t')$ if agent a_i receives a task $t' \notin P_i$ from a different agent. An *exchange swap* (a_i, a_j, t, t') occurs if agent a_i transfers task t to a_j and a_j transfers task t' to a_i. Note that an in-swap and an out-swap regarding the same two agents can be combined into an exchange swap; if this is the case, the swaps are called resolvable.

A *partial* k-swap, denoted as s^k, is a contract that describes all task exchanges of a given set of agents $A(s^k) \subseteq A$. The contract consists of:

- A set of in-swaps that specifies tasks transferred from agents outside $A(s^k)$ to agents in $A(s^k)$.
- A set of out-swaps that contains the tasks transferred from agents in $A(s^k)$ to agents outside $A(s^k)$.
- A set of compact exchange swaps which contains the swaps between agents that are both in $A(s^k)$.

The term k refers to the size of the set of exchange swaps in s^k. Finally, a partial k-swap is complete if and only if it has empty in and out swap sets, and therefore only contains exchange swaps. A complete k-swap describes k task exchanges among multiple agents, and is termed profitable if it decreases the team cost of the allocation. Zheng et al. [190] prove that, given any task allocation problem with n tasks and a suboptimal solution P, there always exists a complete k-swap with $k \subseteq n$ that results in an optimal allocation. Naturally, such a k-swap will be profitable, so it is sufficient to search all possible profitable complete k-swaps in order to find it. The authors propose the following distributed approach that, given an initial suboptimal allocation $P = \{P_1, ..., P_n\}$ [190], and a constant K defined by the user, constructs all profitable complete k-swaps with $k \leq K$:

1. Establish a complete ordering of agents according to an index i, and initialize the set S^{glob} of all profitable complete k-swaps to empty.
2. Every agent a_i initializes three sets to empty: (1) S_i^{local}, the set of swaps that the agent constructs (2) S_i^{send}, the set of all profitable swaps the agent shares with the other agents (3) $S_i^{receive}$, the set of partial swaps that it has received from the other agents.
3. Every agent constructs all feasible partial swaps that contain itself only, appends them to S_i^{local} and if they are found to be profitable, also to S_i^{send}. The agent then sends the swaps in S_i^{send} and resets S_i^{send} to empty.

[8] The term rollout is derived from reinforcement learning, in which it is used to describe the procedure by which policies are evaluated based on true rewards rather than first-round estimates of such rewards [191] [171].

4. For K rounds, each agent a_i:
   - Adds all partial swaps that it has received from others to $S_i^{receive}$.
   - Combines every pair of partial swaps in $S_i^{receive}$ and S_i^{local} that is combinable. Combining a pair of partial swaps s^a and s^b means creating a new partial k-swap s^k by: (1) adding all exchange swaps that were originally in s^a and s^b, (2) adding the new exchange swaps that arise from combining all resolvable pairs of in-swaps and out-swaps ($\langle a_i, -, -, t\rangle + \langle a_j, -, t, -\rangle = \langle a_i, a_j, t, \emptyset\rangle$) and (3) making s^k compact by rewriting all combinable exchange swaps as a single exchange swap ($\langle a_i, a_j, t, \emptyset\rangle + \langle a_i, a_j, \emptyset, t'\rangle = \langle a_i, a_j, t, t'\rangle$). Then:
     - If the combined s^k is a profitable complete k-swap with $k \leq K$ and agent a_i is the core of s^k, then a_i adds s^k to S^{glob}. An agent is said to be the core of s^k if removing him from $A(s^k)$ preserves the profitability of all swaps in s^k, and no agent that comes before it in the ordering established in step 1 has this property.
     - Else if s^k is not complete, but $k \leq K$ and $s^k \notin S_i^{local}$, then agent a_i appends s^k to S_i^{local} and if s^k is profitable also to S_i^{send}.
   - Sends to all the other agents all partial k-swaps in S_i^{send}, and empties the set S_i^{send}.

Following the above procedure, all profitable k-swaps with $k \leq K$ are guaranteed to be constructed within K rounds and are stored in the set S^{glob}. The authors of [190] present two distinct approaches to execute the profitable k-swaps in the context of a sequential auction. The first is the `GREEDY` approach, in which at every round all profitable k-swaps are generated according to the above distributed procedure, and the swap with the highest gain for the system is executed on the current solution. The second is `ROLLOUT`, a more sophisticated but computationally expensive approach based on the idea of rollouts discussed in Section 2.6.1. At each round of the auction, all profitable k-swaps are generated. Each k-swap is hypothetically executed and then the above greedy approach is applied to the resulting post-swap allocation. The team cost incurred after this hypothetical layer is used to value the k-swap in the current round of the auction.

Zheng et al. apply the K-swaps framework to SSI auctions with varying numbers of agents and tasks, using both `GREEDY` with $K = (1,2,3)$ and `ROLLOUT` for $K = (1,2)$ [190]. The results are summarized in Table 2.3. The optimal solution is computed via a MIP solver with a two-hour time limit. Values shown in brackets indicate that the optimal solution was not found within the the limit, and therefore represent upper bounds on the true optimal allocation cost. K-swaps are shown to improve the allocation substantially, with the rollout procedure outperforming the greedy one but as expected being more computationally expensive. For small problems (n=2 and 4), the three-swap greedy implementation is able to find the optimal allocation. For larger problems, none of the tested K-swap variants are able to solve the problem optimally (the required K is evidently larger than 3) but reduce the initial cost substantially, up to ≈ 23.5% for `GREEDY` with $K = 3$ and ≈ 24.2% for `ROLLOUT` with $K = 2$ for the largest problem instance.

Agents	Targets	Minimal cost	Initial cost	GREEDY						ROLLOUT			
				K=1		K=2		K=3		K=1		K=2	
				Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time
2	6	166.2	176.1	166.4	0.00	166.2	0.00	166.2	0.00	166.2	0.00	166.2	0.00
4	12	229.1	265.1	243.4	0.00	233.8	0.00	229.1	0.00	242.2	0.00	232.6	0.02
6	18	265.8	323.1	276.1	0.00	268.2	0.00	266.9	0.04	272.8	0.01	266.3	0.27
8	24	[297.4]	369.8	314.9	0.00	308.4	0.02	299.6	0.20	308.2	0.03	299.6	0.68
10	30	[337.7]	420.5	367.7	0.00	350.4	0.03	340.4	0.67	354.6	0.08	338.7	4.11

Table 2.3: Resulting allocation cost and additional computational time required when performing K-swaps in a SSI auction according to the `GREEDY` and `ROLLOUT` procedures. Results taken from [190].

2.7. Temporal sequential single-item auction (TeSSI)

A recent cooperative auction variant called the *Temporal Sequential Single-Item* auction (TeSSI), introduced by Nunes et al. [112], extends the SSI auction by allowing agents to plan over sequences of tasks with temporal constraints. We consider it a separate variant rather than an improvement to the SSI auction (such as the *rollouts* described in Section 2.6.1), because the planning paradigm used by the agents to schedule tasks and therefore evaluate them within the auction procedure is substantially different. While the allocation of tasks with temporal constraints in auctions had been explored in previous work [105] [109], these attempts had limited flexibility as they were unable to deal with tasks with overlapping time windows or changing start

times. TeSSI is general enough to handle tasks with overlapping constraints and changing start times, making it suitable for online allocation problems in dynamic settings.

Let us return to our standard allocation problem in which there is a set of agents $A = \{a_1, ..., a_n\}$ and a set of tasks $T = \{t_1, ..., t_m\}$. Now, let us consider the added constraint that each task $t_i \in T$ has an earliest start time ES_i and latest start time LS_i with $ES_i \leq LS_i$, as well as a latest finish time LF_i. Every task is also associated with a duration DUR_i such that $LS_i + DUR_i = LF_i$. The task can be started no earlier than ES_i, and can be finished no later than LF_i, and must therefore be executed within the time window $[ES_i, LF_i]$. Each agent captures these properties of the tasks by maintaining its task schedule as a simple temporal network (STN), a three-task example of which is provided in Figure 2.2. The actual task start and end times are represented by the S_i and F_i time points, and the associated constraints as self-loop arrows. An origin time point with a value of 0 is also added to the STN, although not shown in Figure 2.2, to represent the starting point of the agent. There are two main constraints in the STN. The first is that the finish time of a task can be no earlier than the start time, meaning that $DUR_i \geq 0 \ \forall i$. The second is that the agent can only commence the next task once it has completed its current task , meaning that there is no arc in the STN that directly connects two or more task start or task end nodes.

Algorithm 1 TeSSI Task Evaluation and Scheduling Algorithm

Input: task on auction t, agent schedule T_a of length m tasks, agent STN
Output: Optimal insertion position i^* of task t, resulting agent_objective* which is agent's bid for t

```
1:  procedure EvaluateTask(t, T_a, m, STN)
2:      i* = -1
3:      if T_a = ∅ then                          ▷ t is first in STN, no need to find optimal insertion
4:          add time points and constraints of t to STN
5:          agent_objective = compute_objective(STN)
6:          return 0, agent_objective
7:      else
8:          for i in [0,...,m] do
9:              insert t in position i in T_a
10:             add time points and constraints of t to STN
11:             propagate STN via Floyd-Warshall algorithm
12:             if STN consistent then
13:                 agent_objective = compute_objective(STN)
14:                 if agent_objective is smallest so far then
15:                     i* = i
16:                     agent_objective* = agent_objective
17:                 reset T_a and STN eliminating task t
18:         if i* = -1 then                      ▷ No consistent insertion of t into STN possible
19:             return -1, M where M is a very large number
20:         else
21:             return i*, agent_objective*
```

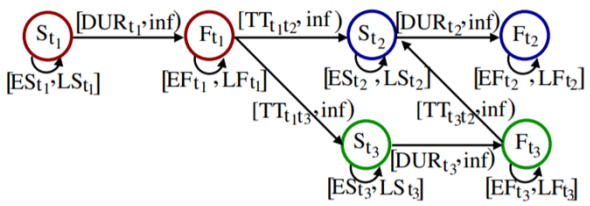


Figure 2.2: STN representation of a three-task schedule. Figure taken from [112].

The TeSSI auction follows the same main logic of the standard cooperative SSI auction discussed in Section 2.6. To value a task t, each agent computes the cost it would incur in adding the task to its local plan and submits it as its bid, with the auctioneer then allocating the task to the lowest bidder. The novelty is that the STN representation provides a polynomial way to schedule and value tasks with temporal constraints. To evaluate a task

t on auction, the agent attempts to insert it into its STN in such a way as to minimize some predefined objective such as the makespan (equivalent to the MINIMAX objective described in Section 2.6) or total path cost (equivalent to the MINISUM objective described in Section 2.6). This procedures is described in Algorithm 1. The agent attempts to insert the task t in all plausible locations in the STN. For each location i, the agent inserts the task time points, along with the associated duration and travel time constraints (line 10), and propagates the STN by using the Floyd-Warshall algorithm (line 11) which runs in $O(n^3)$ time[9]. If the STN is consistent, meaning that there are no negative cycles, the objective of interest to the agent is computed (line 13) and saved if it is the best so far (lines 15-16). The agent bids the lowest value of the objective, that arises from the best insertion position i^*. Every time a new insertion point i is tried, the STN is reset to include only the original tasks and not t (line 17). Note that if the schedule of the agent is empty to start with, it is of no use to search for the optimal insertion and the task is directly inserted as the first task in the schedule to compute the value of the agent's objective (lines 3-6). If the agent is then awarded the task, it inserts it into its STN in position i^*. Nunes et al. [112] focus on minimizing the makespan and a combination of the makespan of the distance traveled, but in principle any bidding rule could be applied.

When a set of tasks T is auctioned, all agents goes through the evaluation procedure for every task t_i and submits a bid for each task. The task t^* that yielded the lowest bid among the entire set of bids (for all tasks) is allocated to the agent a^* with the lowest bid. The procedure is then repeated for $T\backslash\{t^*\}$. In the next round, agent a^* needs to re-evaluate all tasks since its schedule has changed and therefore its bid for the remaining tasks may be different. For the other agents, this is not necessary as their schedules have not changed. Tasks that cannot be assigned to any agent, i.e. they cannot be inserted into their STN in a consistent manner that satisfies all temporal constraints, are simply deleted from the task set. The auction continues until the auctioneer has attempted to allocate all tasks. While TeSSI has been empirically shown to produce optimal allocations for small problems [112], it is not guaranteed to produce optimal solutions in general as it relies on the SSI auction and therefore does not capture all synergies between tasks.

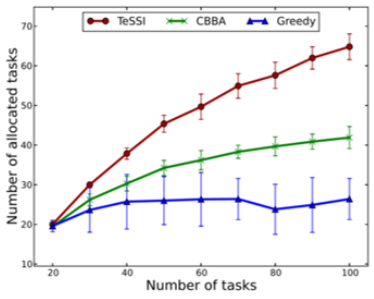
Nunes et al. test the performance of TeSSI with the makespan objective on several offline and online allocation problems and compare it to a version of the Consensus Based Bundle Algorithm (CBBA) that accounts for time windows (covered in Section 2.9), and a greedy approach in which each task is directly awarded to the UAV that can perform it with the lowest makespan. Figure 2.3 summarizes the results for the offline case, in which all tasks are known beforehand and therefore auctioned in a single (multi-round) TeSSI auction. Figure 2.4 shows the results of the online case, in which tasks enter the system in batches, with batch sizes of 1,5,10 and then varying in increments of 10 up to 100. Results show that TeSSI consistently outperforms the CBBA with time windows and the greedy approach, especially for the harder problems. In the online implementation, the quality of the allocation increases substantially with the batch size, since larger batches allow TeSSI to better capture synergies among tasks. Figure 2.5 shows the results for a VRP instance with 100 spatially clustered tasks and tight time windows. TeSSI is able to capture the spatio-temporal synergy present in the VRP instances and produces a much higher-quality and more easily controllable allocation than the other methods.

The TeSSI auction's flexibility makes it a natural way to express the allocation problem for UAV deliveries, in which customers often expect or demand their orders to be executed within a certain time frame. Due to the presence of neighborhoods in urban areas, a spatial clustering of customers is likely a useful feature to capture for effective UAV routing, so the performance of TeSSI observed on the VRP instance in Figure 2.5 is promising. It would be interesting to explore extensions to TeSSI that allow for task swaps in order to make the performance less dependent on the batch size of the tasks being auctioned. It would also be interesting to investigate ways to soften the temporal constraints on the tasks, to be able to allocate tasks also in instances in which the delivery window requested by the customer cannot be met. To the best of our knowledge, no such extensions exist in the literature.

2.8. Sequential single-cluster (SSC) auctions

Koenig et al. introduced the idea of Sequential Bundle Bid (SBB) auctions as a middle ground between SSI and full combinatorial auctions [85]. In SBB auctions, each agent bids on bundles of at most b out of the m tasks to be auctioned. Bundles are then allocated at once, allowing agents to directly capture synergies between up to b tasks. The value of b controls the tradeoff between the SSI ($b = 1$) and the combinatorial ($b = m$) auction. In general, SBB auctions have been shown to reduce the computational and communication overhead by several orders of magnitudes compared to the combinatorial case. However, the SBB auction cannot easily be adapted to the online allocation problem, in which tasks are discovered dynamically throughout the problem and there-

[9] While in principle this is asymptotically no better than running n calls to Dijkstra's algorithm, the Floyd-Warshall algorithm is much more efficient in practice [154].



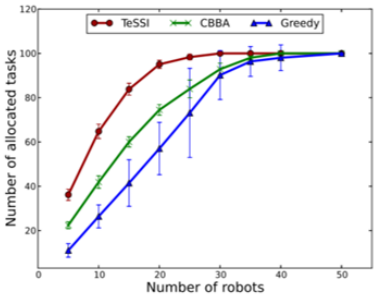


Figure 2.3: Results for allocation problem with 20-100 tasks and 10 agents (left) and 100 tasks with 5-50 agents (right). Taken from [112].

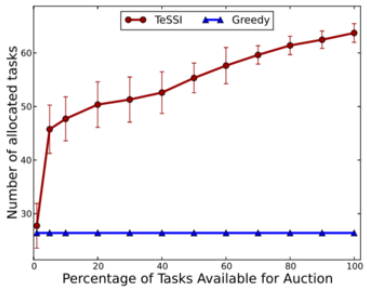


Figure 2.4: Results for allocation problem with 10 agents, 100 tasks discovered dynamically in batches of 1,5,10-100. Taken from [112].

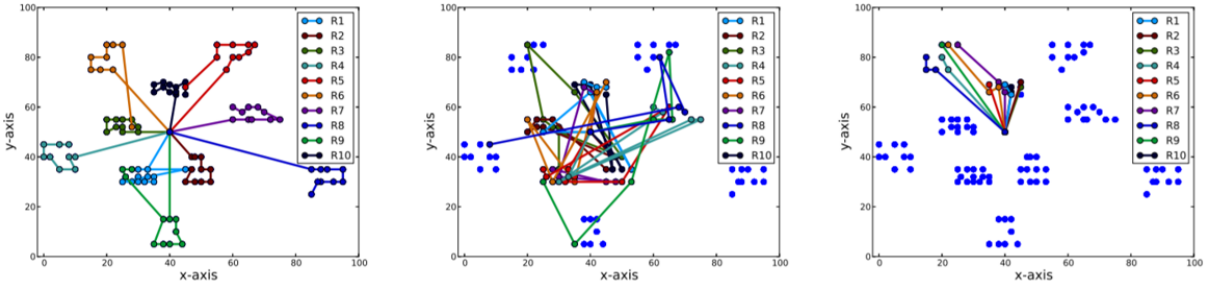


Figure 2.5: Results for Solomon's C101 instance of the VRP, with 10 agents and 100 tasks with tight time windows [112], as solved by TeSSI (left), CBBA (mid) and Greedy (right). Taken from [112].

fore the bundle sizes are not easily controlled. Heap [68] provides a distributed extension of the SBB auction, called the *Sequential Single Cluster* (SSC) auction, that is substantially more flexible and suitable for online allocation in dynamic environments. SSC auctions are based on the idea of clustering the tasks to be assigned before the allocation, and allowing agents to cluster their own uncompleted tasks for dynamic re-auctioning every time they complete a task in their schedule.

Heap and Pagnucco [69] study a variant of the SSC auction designed specifically to handle online pick-up and delivery problems, in which new delivery requests are continuously discovered. The first issue is to develop a framework to cluster pickup and delivery tasks, which is substantially more difficult than clustering tasks consisting of a single location in space. Heap proposes a two-level clustering approach in which tasks are first clustered based on their pickup locations, and then within each cluster they are clustered further based on their delivery location. Figure 2.6 provides an illustration of this approach. This sequential clustering logic allows any existing algorithm that clusters based on single locations in space, such as K-means clustering or single-linkage, to be used on both the pickup location and the delivery location level. While this approach works relatively well in the simple experiments of Heap and Pagnucco [69], it would likely need to be extended to fit the characteristics of the UAV delivery problem. For instance, a temporal layer may need to be introduced in the clustering, to group delivery tasks with close start and finish times. In addition, in large environments it is important to capture the sequential nature of the pickup and delivery tasks and consider distance between the pickup and delivery locations of different tasks, which this two-level approach does not do.

Another challenge in the online SSC auction is that tasks enter the system dynamically. Heap et al. propose a solution to this that is different from other online variants of the SSI auction. When a new task t enters the sytem, it is immediately allocated to one of the agents a^* and added to its local task set. The agent can be chosen at random or based on some characteristic of its task set, e.g. the amount of tasks if we are concerned with balancing the workload among the fleet. Agent a^* can either choose to perform a *local replan*, and replan its schedule and path to account for the new task, or choose to initiate a *global reallocation* procedure, prompting the start of a new SSC auction, which is conducted in a distributed manner. Each agent clusters all of his non-completed tasks, excluding the cluster that includes the task he is currently engaged in, and shares them with all other agents. The agents send their bids for each cluster to all other agents, and parallely receive the other agents' bids. Now that all agents have common situational awareness about the clusters and all bids associated to them, they all perform the same winner evaluation logic independently. Each agent chooses the

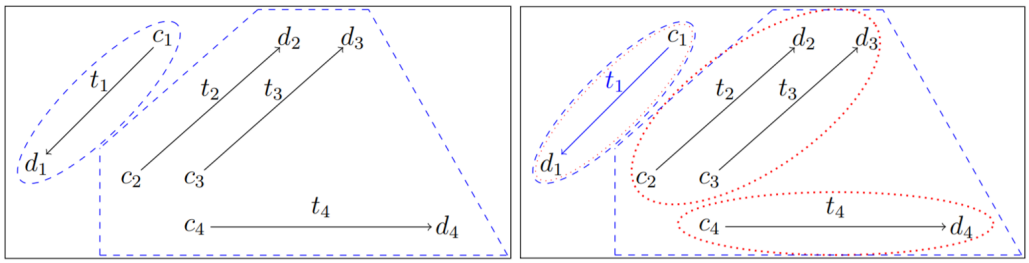


Figure 2.6: Illustration of the clustering procedure for the pickup and delivery SSC auction variant. The first layer of clustering (left) is on the pickup location c_i of the tasks, and the second layer (right) is on the delivery location d_i of the tasks. Taken from [68].

cluster c^* with the lowest bid and removes it from the set of unassigned clusters. The agent with the lowest bid for c^* assigns itself to it and adds all tasks in c^* to its local plan. A new round of the SSC auction then begins for the remaining $C\backslash\{c^*\}$ clusters, until all clusters have been allocated. The same global replanning phase is also initiated every time an agent has completed a task. Note that while in the work of Heap and Pagnucco the SSC auction is performed in a fully distributed manner, alternative implementations are possible. For instance, we could let agent a^* take the role of the auctioneer in the SSC auction. This could be managed by the agents through the passing of a token, which could additionally contain information about the agents' planned paths in order to ensure conflict resolution[10]. Including the role of the auctioneer would sacrifice some robustness but substantially limit the communication overhead in the global replanning phase.

Heap and Pagnucco test the online SSC auction variant on a simulated allocation problem with 10 agents and 60 tasks, to understand the performance advantage that can be achieved via global replanning [68]. They vary the ratio of dynamic to static tasks, with 25%, 50% and 75% of the tasks unknown at the beginning of the problem and use single-linkage clustering with a TPD metric for the clustering routine. Both the `MINIMAX` and `MINISUM` objectives are addressed, and the results are summarized in Table 2.4. The *baseline* case represents the static one-shot SSC case in which all tasks are known beforehand, but no task re-allocations are allowed. The improvement of global replanning over local replanning is showed in parentheses. As expected, global replanning allows for synergies between new tasks entering the system and existing tasks across all agents to be captured, reducing the allocation costs substantially compared to the local replanning case where synergies can only be captured with a single agent's task set. Global replanning improves the team cost up to 36.5% with the `MINIMAX` objective and the least dynamic setting (25% dynamic tasks). Note that the way performance scales with the percentage of dynamic tasks is non-trivial and dependent on the objective. For the global reallocation case, the best performance for the `MINISUM` objective is achieved with 75% of dynamic tasks and not 25% like with the `MINIMAX` objective.

Objective	**Replanning**	**Baseline**	**Percentage dynamic tasks**		
			25%	50%	75%
`MINIMAX`	Local	7857	8228	7276	7275
	Global	7857	5228 (36.5%)	5593 (23.1%)	5911 (18.7%)
`MINISUM`	Local	41539	41527	42025	45305
	Global	41539	37449 (9.8%)	40278 (4.2%)	36907 (18.5%)

Table 2.4: Results of local and global replanning in an online SSC auction for a task allocation problem with 10 agents and 60 tasks, of which a varying percentage is discovered dynamically. Results taken from [68].

2.9. Consensus-based auctions

Consensus-based auctions [20] introduce the idea of a distributed consensus phase in the auction to remove the need of the auctioneer entirely. Instead, the agents converge to a conflict-free allocation independently and without requiring the winner determination phase to be conducted centrally as in a typical auction. The strength of this approach is that it allows for a high degree of flexibility in the network structure, enabling coordination in realistic settings with limited communication ranges, although the convergence rate of the consensus phase is dependent on the network structure. We begin by elaborating on the *Consensus-Based*

[10] This is similar to a recent method from the multi-agent path finding literature called Token Passing with Task Swaps (TPTS) [99], which solves the combined task allocation and path finding problem in the IA case.

Auction Algorithm (CBAA), which solves the single assignment problem and then elaborate on the *Consensus-Based Bundle Algorithm* (CBBA), which generalizes the CBAA and extends it to the multiple-assignment case.

2.9.1. Consensus-Based Auction Algorithm (CBAA)

The *Consensus-Based Auction Algorithm* (CBAA), first introduced in [20], is based on two distinct phases, namely the *auction phase* and the *consensus phase*. By iterating between the two phases, the agents cast their bids and agree on a list of winning bids and hence on the assignment. Throughout the rest of the explanation, we refer to an iteration t as a single run of both the auction and the consensus phase.

Unlike other cooperative auction formulations in which the agents' bid is simply their cost of executing the task, in the original CBAA description the agents attach a *score* to the task and use it to bid, with the highest bidder being awarded the task. We follow this formulation in order to provide a discussion of the convergence properties and limitations of the algorithm (in sections Section 2.9.2 and Section 2.9.3) that is consistent with the theoretical results of Brunet et al. [20]. Note that this is without loss of generality since the score representation is more general than the cost and can be designed to be correlated to the latter, for example by setting the score of a task equal to the negative or reciprocal of its execution cost.

In the auction phase, each agent bids on a task in an asynchronous manner. Let c_{ij} be the bid that agent a_i puts on task t_j, and h_i be agent a_i's availability vector whose jth entry is 1 if the agent is available to perform task t_j. The agent's task list is contained in a vector x_i whose j-th entry is 1 if the agent is assigned to task t_j, and 0 otherwise. Let us also define a vector y_i which contains the agent's current knowledge of the winning bid for each task across all agents. Agent a_i uses its own bids and the knowledge contained in y_i to populate the list h_i of available tasks by checking whether it believes to hold the highest bid for a task j:

$$h_i j = \mathbb{I}(c_{ij} \leq y_{ij}) \tag{2.1}$$

Where $\mathbb{I}$ is the indicator function (1 if the condition is true, and 0 otherwise). Algorithm 2 shows the procedure agent a_i goes through in the auction phase at iteration t. If the agent is unassigned, then it populates the list of valid tasks h_i and finds the task J_i for which it has cast the lowest bid. It then assigns itself to the task, updates the vectors x_i and y_i accordingly and moves to the consensus phase. If the agent was already assigned a task or if there are no valid tasks, the agent moves directly to the consensus phase.

Algorithm 2 CBAA Phase 1 - Task selection procedure for agent a_i at iteration t

```
1: procedure SelectTask(c_i, x_i(t−1), y_i(t−1))
2:     Initialize x_i(t) = x_i(t−1); y_i(t) = y_i(t−1)
3:     if Σ_j x_ij(t) = 0 then
4:         h_i j = 𝕀(c_ij ≤ y_ij) ∀ tasks j
5:         if h_1 ≠ 0 then
6:             J_i = argmax_j c_ij · h_ij          ▷ Find task J_i with highest marginal score
7:             x_{i,J_i} = 1
8:             y_{i,J_i} = c_{i,J_i}
```

In the consensus phase, the agents attempt to converge on the list of winners in the auction. Algorithm 3 shows the consensus procedure for an agent a_i. Let $\mathbb{G}(t)$ be an undirected graph representing the communication network of the agents at iteration t, and let $G(t)$ be its adjacency matrix. If $g_{ij}(t) = 1$, there is therefore a communication link between agents i and k at time t and we denote the agents as *neighbors*. The procedure agent a_i executes in the consensus phase is shown in Algorithm 3. Agent a_i receives the local list of winning bids y_k from each of its neighbors, and replaces the values y_i of its local list with the largest bid for task j between itself and its neighbors. If this update leads agent a_i to find that it has been outbid for a task j, then it unassigns itself from task j. Note that to ensure convergence, tie breaking across equal bids cannot be performed randomly since it needs to be consistent among the entire fleet. A solution proposed by Brunet [20] is to add a small number to each bid ahead of the consensus phase in order to avoid ties.

2.9.2. Consensus-Based Bundle Algorithm (CBBA)

The consensus-based bundle algorithm generalizes the CBAA to the multi-assignment case, and again iterates between two phases: *bundle construction* and *conflict resolution*.

In the bundle building phase, every agent adds tasks to its bundle until it can no longer add any new tasks. The procedure an agent a_i executes in this phase is described in Algorithm 4. Agent a_i has two lists that concern the tasks, namely the bundle b_i in which the tasks are arranged based on the order in which they were added,

Algorithm 3 CBAA Phase 2 - Consensus procedure for agent a_i at iteration t

Send y_i to all neighboring agents k
Receive y_k from all neighboring agents k
procedure UpdateInformation($g_i(t)$, $y_k(t)$ for all neighbors k, J_i)
    $y_{ij}(t) = \max_k g_{ik}(t) \cdot y_{kj}(t) \forall$ tasks j
    $z_{i,J_i} = \text{argmax}_k g_{ik}(t) \cdot y_{k,J_i}(t)$
    if $z_{i,J_i} \neq i$ **then**
        $x_{i,J_i}(t) = 0$

and the path p_i in which the tasks are arranged based on the order in which a_i intends to execute them. Furthermore, let S^{P_i} be the total score agent a_i receives from executing all tasks in the path p_i and let z_i be the list of winning agents based on the local knowledge of agent a_i. When a task j is added to the agent's bundle b_i, it generates a marginal improvement $c_{ij}(b_i)$, which is the result of adding the task to the path in the location that leads to the highest score. The operation $l_i \oplus_n l_j$ refers to adding list l_j just after location n of list l_i. The agent finds the available task J_i that leads to the highest marginal score improvement, and adds it to the end of the bundle b_i and in the location n_j along path p_i that maximizes the total score S^{p_i}. It then assigns itself to task J_i by updating the vector y_i and z_i accordingly. The process continues until there are no more tasks available or the maximum bundle size has been reached.

Algorithm 4 CBBA Phase 1 - Bundle building procedure for agent a_i at iteration t

procedure BuildBundle($z_i(t-1)$, $y_i(t-1)$, $b_i(t-1)$, $p_i(t-1)$)
    Initialize $z_i(t) = z_i(t-1)$; $y_i(t) = y_i(t-1)$; $b_i(t) = b_i(t-1)$; $p_i(t) = p_i(t-1)$
    while $|b_i(t)| < B$ **do**
        $c_{ij} = \max_{n \leq |p_i(t)|+1} \left(S^{p_i(t) \oplus_n [j]} - S^{p_i(t)} \right) \forall$ tasks $j \notin b_i(t)$
        $h_i j = \mathbb{I}(c_{ij} \leq y_{ij})$
        $J_i = \text{argmax}_j \; c_{ij} \cdot h_{ij}$ ▷ Find task J_i with highest marginal score
        $n_{J_i} = \text{argmax}_n \; S^{p_i(t) \oplus_n [J_i]}$ ▷ Find optimal insertion point of J_i into p_i
        $b_i(t) = b_i(t) \oplus_{\text{end}} [J_i]$
        $p_i(t) = p_i(t) \oplus_{n_J} [J_i]$
        $y_{i,J_i} = c_{i,J_i}$
        $z_{i,J_i} = i$

In the consensus phase, the agents communicate with their neighbors and compare bids for tasks in their bundle with those of the other agents, in a manner similar to the single-assignment CBAA case. The main difference with CBAA is that if an agent discovers that it has been outbid for a task, it is not sufficient to simply unassign itself from that task. The agent must also release all other tasks in the bundle that it has added after that task, because their scores are dependent on the task for which it was outbid. Every agent a_i needs to share three distinct vectors in the consensus phase: (1) the list of winning bids y_i, (2) the winning agents list z_i and (3) a vector of time stamps s_i, where the k-th element represents the timestamp of the latest information that the agent received from a_k, either directly or via a neighbor. When agent a_i receives information from an agent a_k, it uses the lists y_i and z_i to merge its local and the incoming information by performing one of the following three actions for task j:

1. `UPDATE`: $y_{ij} = y_{kj}$, $z_{ij} = z_{kj}$
2. `RESET`: $y_{ij} = 0$, $z_{ij} = \emptyset$
3. `LEAVE`: $y_{ij} = y_{ij}$, $z_{ij} = z_{ij}$

For instance, an update action would be undertaken if a better bid for a task j is shared from agent a_k, a reset action would occur if both agents believe that the other is the winner of the task, and a leave action if a lower bid for j is received from k. For a detailed decision table that provides rules on which of the above three actions to take for any combination of y_{ij}, z_{ij} and y_{kj}, z_{kj}, the reader is referred to [20]. Generally, the decision logic is structured to favor higher bids and the most recent information.

Convergence and diminishing marginal gains

In order to ensure convergence, CBAA and CBBA require the scoring function to satisfy a property known as *Diminishing Marginal Gains* (DMG). The DMG property can be seen as a subset of the submodularity condition for a function, and can be formalized as follows:

$$c_{ij}(b_i) \geq c_{ij}(b_i \oplus_{\text{end}} b), \quad \forall b_i, b \tag{2.2}$$

That is, the marginal score of a task j cannot increase due to more tasks being added to the bundle. If the score function is DMG (meets the DMG condition), then CBBA is guaranteed to converge to a conflict-free assignment also for problems in which the underlying network structure changes over time, provided that $\exists\, \rho < \infty$ such that:

$$W(t) = G(t) \cup G(t+1) \cup ... \cup G(t+\rho-1) \quad \text{is fully connected } \forall\, t \tag{2.3}$$

If this is not the case, then there will be agents that will never be able to exchange information through the network and it is therefore impossible to guarantee a conflict-free assignment. If the condition holds, then the assignment produced by CBBA is guaranteed to be no more than twice as costly as the optimal assignment [20].

2.9.3. Extensions to CBBA

While CBBA is a powerful paradigm in that it provides convergence guarantees and provably good solution quality irrespective of the network structure, it fails to capture several characteristics of real-life allocation problems. To fill these gaps, a number of variants of CBBA were developed, the most influential of which are CBBA for time-sensitive tasks and non-DMG score functions.

Time-sensitive tasks

Ponda et al. [120] modify the scoring function of CBBA in order to deal with time-sensitive tasks whose utility decreases over time and that are valid only if executed within a set time window. The modification to the score function is relatively straightforward and consists of two main components:

- *Time-varying score profile* $s_j(t)$: The reward that an agent obtains for performing task j is rendered a function not only of the face value V_j of the task, but also of time. An example score profile is: $s_j(t) = e^{-\lambda_j(t-t_j^{start})} \cdot V_j$, where $\lambda_j \geq 0$ is the discount rate use to penalize tasks that take longer.
- *Mandatory time window* $u_j(t)$: Tasks have a time window $T_j^w = [t_j^{start}, t_j^{end}]$ within which they must be started. The window is incorporated into a validity function $u_j(t)$ as follows:

$$u_j(t) = \begin{cases} 1 & , t \in T_j^w \\ 0 & , else \end{cases} \tag{2.4}$$

The score for task j is then a function of the completion time t_c and is equal to $C_j(t_c) = s_j(t_c) \cdot u_j(t_c)$. Note, however, that for the UAV delivery case we would need to extend this formulation to include constraints on the pickup *and* delivery times. A straightforward way to do this would be to set a delivery time window $u_{j,d}(t)$ as specified by the customer and a pickup window $u_{j,p}(t)$ in which the start time is conditional of the expected availability of the item at the pickup location, and the end time is derived from the duration of the task and the latest allowed delivery time. In the bidding process, the agent finds the optimal manner to insert a task j into its plan, and therefore attempts to perform the task as quickly as possible to maximize $s_j(t)$ and within the time window $u_j(t)$ to ensure a non-zero reward. The issue with this formulation, however, is that we need to sacrifice flexibility in task scheduling in order to ensure that the marginal score c_{ij} to any agent a_i of adding task j to its bundle is DMG. If we add a task j to the bundle, then for no task $k \neq j$ can c_{ik} increase as a result. This is highly limiting, as it implies that we cannot shuffle the order of the tasks $k \neq j$ in the path and therefore cannot capture the synergies between tasks already in the bundle and newly added tasks. The performance advantages of capturing these synergies are evident and discussed extensively in Section 2.6, Section 2.7 and Section 2.8.

Non-DMG score functions

Johnson et al. [77] present a key improvement that allows for non-DMG functions to be used without compromising the convergence guarantees of CBBA. It is based on the idea that the local evaluation function of an agent for a given task need not necessarily be equal to the bid that the agent casts for that task. As long as the *bids* are DMG, then in principle it does not matter which evaluation function the agent is using. Let us define a warping function $G_{ij}(c_{ij}, b_i) = c'_{ij}$ that given the agent's valuation for a task c_{ij} and its current bundle b_i produces a warped valuation c'_{ij}:

$$c'_{ij} = \min(c_{ij}, y_{ik}), \quad \forall\, k \in p_i \tag{2.5}$$

where y_{ik} is agent a_i's bid for task k, since if a task is in his bundle then a_i believes to be its winner. If the agents bid according to c'_{ij} rather than c_{ij}, then no new bid will ever be larger than the bids previously made by the agents for tasks already in their bundle. The warped bids therefore satisfy the DMG property. This is easily seen from the structure of the warping function, but for a formal proof the reader is refered to[77]. The only requirement on the agent's local evaluation function c_{ij} is that the valuations are *reproducible*, meaning that given the same conditions and initial bundle, the valuation will also be the same. The non-DMG score

function requires some changes to the bundle generation procedure described in Algorithm 4. Instead of the real valuation, the warped bid c'_{ij} for a task j is compared with the list of winning bids to determine whether a_i holds the highest bid for j (line 5). The real valuation, however, is still used to determine the best task among the tasks available to a_i for assignment (line 6). The consensus phase remains the same as in standard CBBA, with the exception that the warped bids are shared and used for consensus. Since the warped bids are DMG, the agents are guaranteed to converge to a conflict-free assignment provided the condition in Equation (2.3) holds.

Johnson et al. show the effectiveness of using CBBA with non-DMG score functions by comparing it to standard CBBA on an allocation problem with 2 agents and 30 tasks. The results are shown in Figure 2.7. With standard CBBA, the agents cannot incorporate the actual distance travelled in the score function since it is not DMG. It is straightforward to observe why. Suppose we are at some round t of the bundle-building phase and agent a_i is to add a new task j to its bundle. If we score task j based on the additional distance it would require a_i to travel (in a manner similar to all methods described in Section 2.6, Section 2.7 and Section 2.8), then the bid for task j may very well be higher than the bids for other tasks in b_i, because executing some task $k \in b_i$ may bring a_i closer to task j. To go around this, the authors in [20] submodularize the cost by considering the *total* distance travelled instead of the marginal distance, which does not capture inter-task synergies. With non-DMG CBBA, instead, we can capture inter-task synergies and score tasks based on the marginal distance, which leads to a much higher quality allocation.

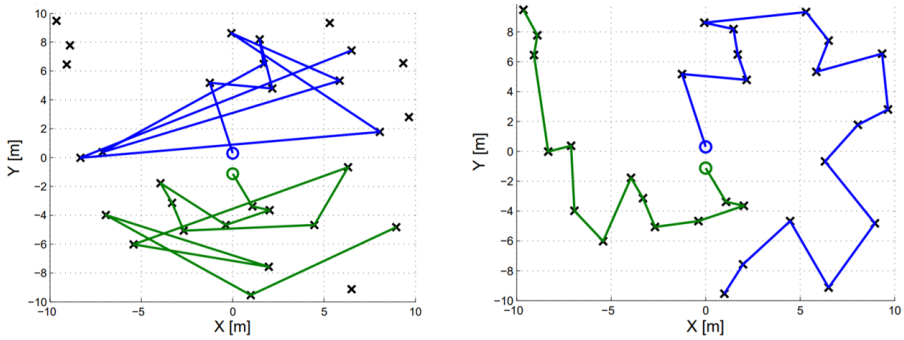


Figure 2.7: Solutions generated by standard CBBA with submodular cost approximation (left) and non-DMG CBBA with cost based on true marginal distance (right) for a one-shot allocation problem with 30 tasks and 2 agents. Taken from [77].

Other improvements

Ponda introduces a variant called CBBA with Relays [120] that attempts to prevent the network from getting disconnected throughout the mission in order to ensure convergence of the consensus phase. The same variant can also be used to ensure that the entire network stays connected to a base station, which may be desirable for safety or regulatory purposes in real-life applications. The method is based on the idea of relay tasks, whose purpose is to ensure the network stays connected in instances where executing a task would cause a disconnect. Just like regular tasks, relay tasks can include locations and time windows, and the CBBA structure is leveraged to allocate them in a distributed manner. Ponda further extends CBBA to improve its performance in problems with uncertainty regarding parameters such as task durations, task locations and vehicle velocity by proposing Robust CBBA, in which agents generate stochastic plans over their task set using a sampling strategy. The recent work of Whitbrook et al. [182] [183] proposes a robustness module ROB-M that when implemented with CBBA largely reduces the sample size required to handle comparable levels of stochasticity as in Robust CBBA. Luders et al. propose a stratified framework [96] that combines CBBA with information-rich RRT for exploration and localization tasks, which to the best of our knowledge is the only attempt to integrate a path planning layer within CBBA.

While several variants of CBBA exist in the literature, there are still a number of challenges to overcome in order to apply consensus-based auctions to the UAV delivery coordination problem. The first is to extend CBBA to account for pickup and delivery tasks, whose temporal constraints are different from single-location tasks. Planning complexity is also higher for pickup and delivery tasks, since both the pickup and the delivery location, together with the associated path segments, can represent sources of uncertainty. Surprisingly, there has been little work on incorporating task swapping or re-allocation procedures within CBBA. This is because the focus of efforts involving consensus-based auctions has typically been on one-shot allocation cases. An exception is the recent work of Buckman et al. [22], which proposes CBBA with Partial Replanning [22], in

which agents only release a subset of their tasks when a new task enters the system and triggers a new auction. However, the approach has a number of limitations because it relies on the DMG scoring function of standard CBBA. This only allows agents to release the n lowest tasks in their bundles, and is therefore blind to other synergies between tasks in the bundle. These could be identified by clustering routines such as in SSC auctions (Section 2.8) or other heuristic approaches.

3
Multi-agent path finding

The problem of multi-agent path finding (MAPF) has received considerable attention in the field of distributed artificial intelligence. It consists of finding conflict-free paths that bring all agents from their target to their goal locations, and is relevant for the UAV delivery problem because UAVs need to plan trajectories to execute their pickup and delivery tasks without colliding with each other or with obstacles in the environment. We begin by briefly discussing the single-agent variant of the path planning problem in Section 3.1 and then elaborate on the multi-agent extension in Section 3.2, including a discussion of different types of MAPF formulations, desirable solver properties, commonly used cost functions and finally an overview of the available solver classes. Each of these classes of solvers is discussed in further detail, and state of the art solvers in each class are singled out. In Section 3.8, we perform a comparative evaluation betwen state of the art solvers in each class and determine the most suitable approaches for the UAV delivery coordination problem.

3.1. Single-agent pathfinding

Single-agent pathfinding is the process of finding a conflict-free path between two vertices in a graph that typically minimizes path length as defined by a cost function, and is a commonly studied problem in artificial intelligence [149]. There are several search approaches available to generate optimal solutions to single-agent pathfinding problems. One of the most well known is A* search [65], which is worthwhile to discuss in more detail given that it still plays an important role in several state of the art multi-agent path finding algorithms.

A* search is a best-first search routine with cost function of the form $f(n) = g(n) + h(n)$, where $g(n)$ is the cost to reach node n from the start node, and $h(n)$ is a heuristic to estimate the cost of reaching the closest goal from node n. A heuristic is *admissible* if it is guaranteed to never overestimate the cost of reaching the goal from node n. If an optimal path exists and h is admissible, then A* search is guaranteed to find it [40]. The most commonly used heuristic is diagonal distance (Euclidean or Manhattan) to the goal neglecting all obstacles.

3.2. Multi-agent path finding

Multi-agent path finding (MAPF) is an extension of the problem to the case with multiple agents. While there are several variants of the MAPF problem, for the purpose of this survey we start by introducing one of the most general and commonly addressed formulations in the literature [149] [159].

3.2.1. Problem Definition

We define a MAPF problem on a graph $G = (V, E)$ and a set of k agents $(a_1, a_2, ..., a_k)$. The vertices of G represent the possible agent locations and the edges the possible transitions between locations. Each agent a_i has a unique start vertex $s_i \in V$ and a unique goal vertex $g_i \in V$. At every discrete time step t_j, each agent can either perform a *move* action and traverse an edge of G to move to a new location, or perform a *wait* action and stay at its current location. A *conflict* occurs when two or more agents occupy the same vertex or traverse the same edge in the same timestep. The problem consists in finding a conflict-free path for each agent from its start to its goal location, where a path p_i for agent a_i is a set of actions such that, if executed from s_i, it will lead the agent to g_i [149].

3.2.2. Centralized vs distributed MAPF

MAPF instances can be grouped into two high-level categories: *centralized* and *distributed* [149]. In centralized MAPF, a central computing body is assumed to have full knowledge of the state and plans of all agents, and uses it to compute a solution for the entire system (all agents) while attempting to minimize a system-wide cost function. In distributed MAPF, each agent is responsible for computing its own path and different conflict resolution strategies can be used, the most common being some form of prioritization [31][32] whereby lower priority agents replan their paths treating the paths of higher priority agents as fixed obstacles. Other conflict resolution approaches that have been tested include taxation schemes [16], auctions [6] and bargaining [125].

Distributed MAPF approaches are useful in the modeling of complex real-life systems with large numbers of agents since they tend to be computationally cheaper than centralized approaches [41], in which complexity scales exponentially with the number of agents. This is especially true in MAPF instances where the agents' tasks are loosely coupled [17] and conflict density is low. Centralized approaches however, offer more tractability over solution cost and typically stronger optimality guarantees.

The urban UAS literature initially focused on centralized approaches to path finding [72][71], which are also vastly more popular within the broader MAPF community. However, as explained in Section 1.4.2 the most recently proposed UTM architecures suggest a high degree of decentralization. Decentralized networks have demonstrated to be substantially more robust, resilient and agile compared to centralized networks for a number of applications [175][12][123] and are seen as a more suitable paradigm for UTM [34]. Recent work has therefore taken a *decentralized* stance to MAPF in an urban UAS context [4], envisioning a situation in which multiple service providers manage their own UAV fleets and coordination between service providers is achieved via negotiation and without the need of centralized directives. However, the path finding problem per service provider is solved centrally via ECBS [11], meaning that a large part of the reasoning remains centralized.

3.2.3. Solver properties

There are several classes of solvers available for MAPF problems, and in order to appreciate the differences between them we must first introduce three key properties of algorithms: soundness, completeness and optimality [13]. Soundness guarantees that any solution that is returned is valid; in the context of pathfinding, an algorithm is sound if any set of paths returned leads all agents from their start to their goal locations in a conflict-free manner. Completeness guarantees that the algorithm will find a solution if it exists. Optimality implies that the algorithm will terminate with a solution that maximizes or minimizes a predefined cost function. Based on these definitions, we can distinguish between three general types of solvers:

- *Optimal*: complete, sound and optimal. Optimal solvers will search all possible arrangements of agents in G for each time step if necessary, and are guaranteed to return a solution that is optimal in the cost function, provided that it exists.
- *Suboptimal*: sound and complete, but not optimal. A set of conflict-free paths will be returned if it exists, but there is no guarantee that this set of paths will be optimal in the cost function.
- *Bounded suboptimal*: sound, complete and guarantee that the solution cost will lie within a set bound (typically defined by the user) of the optimal cost.

Solving MAPF problems optimally has been shown to be NP-hard [187] since the dimension of the state space scales exponentially with the number of agents, so it may be impractical to deploy optimal solvers for problems with a large number of agents. That is why much of the recent literature has focused on proposing bounded suboptimal variants of optimal solvers for use in large MAPF problems where the objective is to find a solution relatively quickly while retaining some degree of tractability over the solution cost.

3.2.4. Cost functions

Most of the work on MAPF has focused on minimizing one of two system-wide cost functions, namely *makespan* and *sum of costs*. *Makespan* is the total number of time steps required until all agents have reached their goal [159][150][149][166]. *Sum-of-costs* is the sum over all agents of the time steps required to reach their goal [45][159][150][148]. Traditionally, search-based approaches have focused on the sum-of-costs objective and reduction-based approaches on the makespan, but recent work has addressed both objectives using both types of solvers [168]. Recently, an alternative MAPF objective related to the makespan has emerged, in which the goal is to maximize the number of agents reaching their goal within a certain time interval [100]. Another high-level cost function measures the total distance travelled by all agents, and is referred to as *fuel* [53].

A series of studies have deviated from the system-wide cost representation and instead formulated and solved MAPF problems in which each agent has its own cost function to minimize [91]. Some have treated agents as self-interested, and devised mechanisms to lead them to cooperate. Bnaya et al. [16] introduced a taxation framework to incentivize self-interested agents to avoid collisions and avoid locations in which a traffic jam would otherwise occur. Their Iterative Taxation Framework (ITF), assigns taxes to specific (location, time) tuples in such a way as to lead agents to choose trajectories that lead to higher social welfare. However, their framework is not guaranteed to maximize social welfare, is not strategyproof since strategic agents can achieve lower taxes by reporting their goals untruthfully, and only supports homogeneous agents since the cost associated to traversing any edge is required to be constant over all agents. Amir et al. [9] address the weaknesses of the taxation approach by formulating self-interested MAPF as an iterative combinatorial auction (CA) [117]. In their representation, agents bid iteratively on bundles of paths and an auctioneer allocates conflict-free paths

to the agents. This CA formulation allows to solve self-interested MAPF instances while guaranteeing strategyproofness and optimality.

3.2.5. Overview of MAPF solvers

There are a number of solvers available for MAPF problems with different computational properties. Figure 3.1 shows a breakdown of the most prominent classes of solvers, which can be broadly split into optimal and (bounded) suboptimal approaches. We study the relevant solver classes and discuss state of the art techniques within each class. Throughout the analysis, we identify the solvers which are most applicable to planning in the context of UAV delivery, and focus our attention on those. We do not discuss all techniques to the same level of depth, but rather provide arguments as to why specific approaches are not deemed highly applicable to the UAV delivery problem and therefore not discussed in as much detail as others. In Section 3.8, we provide a summary of the findings on each class of MAPF techniques, and perform a comparative evaluation between state of the art solvers in each class.

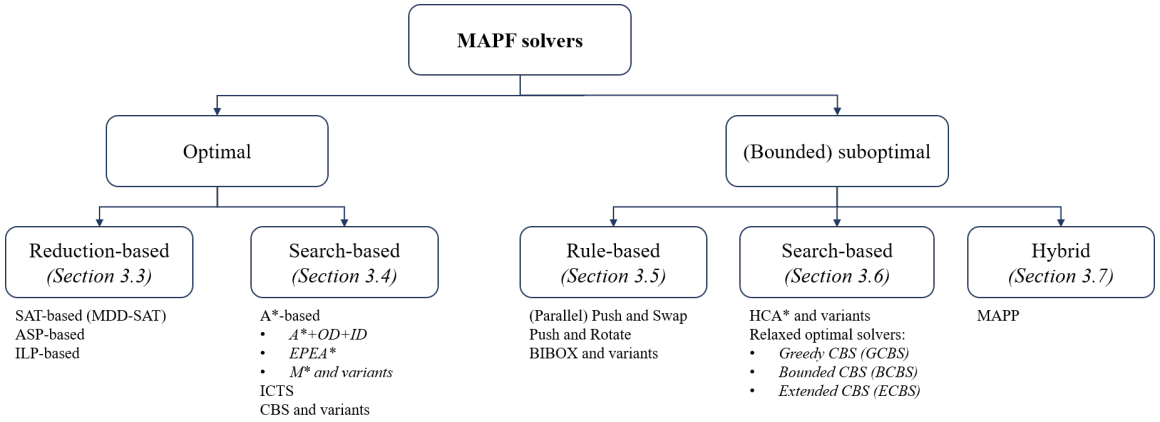


Figure 3.1: Overview of MAPF solver classes.

3.3. Reduction-based solvers

This class of optimal solvers emerged in recent work and reduces MAPF to well-studied problems in mathematics and computer science. Examples include reductions to SAT [167], Answer Set Programming (ASP) [48] and Integer Linear Programming (ILP) [186]. Efficient commercial solvers, such as CPLEX or Gurobi for ILP, can then be used on the reduced problem instance. The majority of reduction-based approaches were developed to minimize the makespan objective, and adjusting them to address other objectives is not trivial and may require an entirely new reduction. This class of approaches therefore provides very little flexibility in the cost function.

In SAT-reduction approaches, the structure of the graph, the agents' locations and all constraints are encoded into boolean variables, and an SAT formula is generated that determines whether there exists a valid solution with cost K. The optimal solution is then determined via a search over all costs K [54]. Surynek et al. developed the first SAT-based solver applicable to the sum-of-costs MAPF variant. They also introduce an improved variant, *MDD-SAT*, that uses special compact data-structures known as multi-value decision diagrams (MDDs) [156] to restrict the number of propositional variables that appear in the SAT formula. They showed that MDD-SAT can be competitive with state of the art search-based solvers for cases in which there is a loose enough runtime constraint ($\approx$ 100 seconds for a Dragon Age problem from the *movingai* repository [161] with 16 agents - Ost003d), especially when the number of agents is low and the obstacle density is high [169]. Note that high obstacle density is an advantage for SAT-based solvers since there are less available nodes and therefore less variables in the SAT formula, whereas they are a disadvantage for search-based solvers.

MDD-SAT is considered to be the state of the art reduction-based solver and the only one able to compete with search-based approaches, albeit only for specific problem instances. However, we are concerned about its applicability to urban UAS operations given that the associated MAPF problem instances will involve a large number of agents and a much lower obstacle density than traditional MAPF formulations, and these are exactly the scenarios in which MDD-SAT performs worst. The limited flexibility in the cost function would also restrict our analysis to minimizing either the makespan or the sum-of-costs, since an entirely new SAT encoding would be required to explore alternatives.

3.4. Search-based optimal solvers

Search-based optimal solvers are perhaps the most influential class of MAPF techniques, and can be classified into three main variants: *A* based optimal solvers* discussed in Section 3.4.1, *Increasing Cost Tree Search (ICTS)* discussed in Section 3.4.2 and *Conflict Based Search (CBS)* discussed in Section 3.4.3.

3.4.1. A*-based optimal solvers

A* is an iconic single-agent pathfinding algorithm (discussed in Section 3.1), but it can also be extended to MAPF problems by deploying it on a global search space that includes the states of all k agents . We call this the *k-agent state space* and it is made up of all the possible ways to arrange the k agents into all $|V|$ vertices. As discussed in Section 3.1, chosing an admissible heuristic guarantees optimality of A*-based approaches. In MAPF, several choices for the heuristic are possible. The simplest is to take the sum of the individual heuristic of every agent, which is typically the Manhattan or Euclidean distance to the goal while ignoring all obstacles [141]. A more informed admissible heuristic is the *sum of individual costs* heuristic, which sums the costs of the optimal paths for each agent to its goal, neglecting all other agents. While the heuristic can be calculated in runtime by solving a single-agent path finding problem at every node [153], it can also be pre-computed by running a breadth-first search from an agent's goal to every free vertex to improve runtime performance [159].

Limitations of A* for MAPF

The *branching factor* b_{agent} of a single agent is the amount of vertices that the agent is able to move to at a given timestep. For instance, on a traditional grid $b_{agent} = 5$, since the agent is able to move in four directions (N, S, W, E) or wait and stay at its current vertex. An important observation regarding A* for MAPF is that the *effective branching factor*, that is the combined branching factor for all k agents, is exponential in k and is bounded above by $b = b_{agent}^k$. While most of the time it will be somewhat less than b_{agent}^k since not all moves for all agents will be legal, the state space will still be prohibitively large for problems with many agents. Consider for instance a MAPF instance with 15 agents on a four-connected grid. The branching factor $b = 5^{15} = 3.05 * 10^{10}$ so we may need to generate $3.05 * 10^{10}$ possible states to explore at every time step, which could be unfeasible from a computational perspective. In addition, when A* expands a state, it stores all of its possible successors in a list OPEN of next states to be expanded. When the list of successors is very large as in our example, we can quickly run into issues when trying to store all of them in memory. Several improvements were made to speed up multi-agent A* and to make it scale up better with the number of agents: *Independence Detection (ID), Operator Decomposition (OD)* and *Enhanced Partial Expansion (EPEA*)*.

Improvement 1: Independence Detection (ID)

Rather than a separate algorithm in itself, Independence Detection is a framework due to Standley [160] that groups agents into distinct groups, determines optimal paths for each group separately and then solves conflicts via merging. Initially, all agents are are assigned to their own group and optimal paths are found independently for each group via A*. The paths of all agents are then checked for conflicts between each other. If a conflict is found, an attempt is made to find an alternative conflict-free optimal path for either of the conflicting agents. If this process fails for both agents or two agents that had already conflicted before are found to conflict again, then they are merged into a *group* and a MAPF instance is solved for the group collectively. This process is repeated until there are no conflicts between any of the groups. Note that groups consisting of multiple agents are also merged in the same way as described above. When a conflict between two groups G_1 and G_2 is found and it cannot be resolved by finding alternative optimal paths for the agents in either group or the groups have already conflicted in the past, then they are merged into a single group $G_{1,2} = G_1 \cup G_2$.

Surplus nodes in A*

Provided certain restrictions are met, it is proven that A* *expands* the minimum number of nodes required to find the lowest cost solution [40]. However, A* will also *generate* nodes that are actually not required to find an optimal solution and will therefore never be expanded. These nodes are known as *surplus nodes*, and avoiding these nodes from being generated can make the search much faster [59] [60]. The following two improvements are designed to circumvent the generation of surplus nodes.

Improvement 2: Operator Decomposition (OD)

Similarly to ID, Operator Decomposition (OD) is a framework due to Standley [160] that can be applied on top of A*-based algorithms to improve efficiency. The driving principle is to consider agents one at a time at each time step. Agents are given an arbitrary order. When we expand an A* state, we now consider and assign only the moves of the *first* agent. When we assign a move to the first agent we move to an *intermediate state*, which

is defined as a state in which at least one agent has been assigned a move. From the intermediate node, we then consider the moves of the *next* unassigned agent based on the arbitrary order we had fixed. generating new intermediate states. Only once we have assigned a move to the last agent and therefore generated k intermediates states, do we move to the next time step and generate a *standard state*. This effectively scales down the branching factor b from b^k_{agent} to b_{agent}, while increasing the depth of any goal state by a factor k.

Improvement 3: Enhanced Partial Expansion (EPEA*)

Enhanced Partial Expansion A* (EPEA*) is a highly efficient A* variant introduced by Goldenberg et al. [60], that uses a priori knowledge of the problem to avoid generating surplus nodes.. It is actually an extension of an algorithm that long preceded it, Partial Expansion A* [185] (PEA*), which already deals with the memory aspect of surplus nodes. When PEA* expands a node N, all $b = b^k_{agent}$ children nodes are generated, but only the ones with $f = f(N)$ are then placed into OPEN with the cost of the best ignored child (the one with the lowest f-value). EPEA* takes this one step further and only *generates* the children of N with $f = f(N)$. This is achieved via an *Operator Selection Function (OSF)* that, for a given expansion of a node N, is able to return: 1) the set of children with cost $f = f(N)$ and 2) the minimum f-value among the nodes with cost $f > f(N)$. The OSF is constructed by making smart use of knowledge regarding the domain and heuristic of the problem. For instance, given the situation depicted in Figure 3.2 we could implement the following OSF: "if the goal is to the North-West of the current location of the agent, then moving North or West will result in the same f-value, while moving South or East will increase the f-value by two" [60]. The OSF would therefore tell us that from N we only need to generate the nodes resulting from the "North" and "West" action. We store those nodes in OPEN with cost $f = f(N)+2$ because the OSF tells us that is the cost of the best children with $f > f(N)$. If $f(N)+2$ becomes the new best cost in OPEN, then we will re-expand node N and generate the nodes resulting from the "South" and "East" actions.

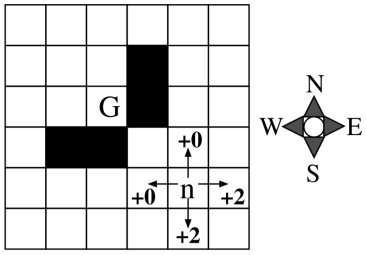


Figure 3.2: Example of the implementation of an OSF in EPEA*. G is the goal and n is the node being expanded. Taken from [60].

M*

Another prominent A*-based algorithm called M* was introduced by Wagner et al. [178]. M* aims to reduce the branching factor where possible by dynamically adjusting it based on the number of conflicts. Crucial to M* is the concept of *dimensionality* of a node N in the k-agent search space, which is the number of agents that are *not* allowed to conflict. M* works on the global search space but starts by only allowing single agents to make moves. Every time a node N is expanded, we generate one child node in which each agent executes one of its optimal paths to the goal while disregarding other agents. We continue going down the search tree in this fashion until a conflict between $c \geq 2$ agents occurs at node N_c. We then move back to all the ancestor nodes of N_c, increase their dimensionality to c and put them back into OPEN. Upon expanding one of these nodes again, our branching factor will be b^c_{agent} since we will generate children for all combinations in which the c conflicting agents make *all* available moves and the rest of the agents simply take their own individual optimal action (disregarding the other agents). Since it was first introduced, M* was refined in several other contributions. A variant called recursive M* (rM*) adopts the same idea behind ID and essentially splits the c conflicting agents into groups of agents that have independent conflicts, and then solves the resulting subproblems recursively. The latest extension of the algorithm termed ODrM* [55] uses rM* on top of A*+OD rather than simple A*, and has been experimentally shown to be the best performing variant in the M* family both in terms of runtime and hardness of solved problems. Felner et al. test it experimentally against rM* and A*+OD on an eight-connected 32x32 grid where each cell has 20% probability of being instantiated as an obstacle. The number of agents was varied from 5 to 60 and 100 random environments were generated for each number of agents. Figure 3.3 summarizes the results, with the percentage of instances solved within 5 minutes plotted on the left and the median time to compute a solution plotted in logarithmic scale on the right. The time to solution plot is cut off when a given percentage of trials reaches the 5 minute limit. ODrM* is shown to outperform both A*+OD and rM*,

solving substantially more instances in all but the easiest problems with up to 10 agents. ODrM* solves 20% to 30% more cases than simple rM* for large problems and up to 60% more cases than A*+OD for problems with 20 agents, after which A*+OD is barely able to solve any instances within the time limit. Runtime performance of ODrM* is also superior to rM*, because the OD framework allows ODrM* to expand less nodes than basic A*, which underlies the rM* algorithm.

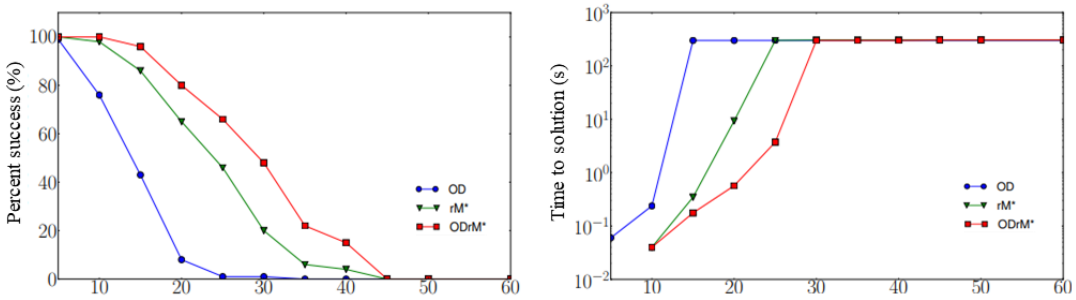


Figure 3.3: Results on an eight-connected 32x32 grid with 20% obstacle density of OD, rM* and ODrM* for varying numbers of agents. Figure taken from [55].

3.4.2. Increasing Cost Tree Search (ICTS)

The Increasing Cost Tree Search (ICTS) algorithm is a widely used search algorithm for MAPF introduced by Sharon et al. [150] that belongs to the recent class of algorithms which are fundamentally different from A*. It is based on a two-level search, in which the high level searches for a minimum cost solution across combinations of costs of individual agents, and the low level takes the cost combination from the high level and performs a *goal test* (searches for a valid solution that achieves the cost combination).

High level: At the high level, ICTS performs search on a tree termed the *increasing cost tree (ICT)*. Every node N in the ICT consists of a k-vector of costs $f(N) = \langle C_1, C_2, ..., C_k \rangle$ which represent the individual path costs of each agent. All valid solutions that generate cost vector C, and which therefore have total cost $\sum_{i=1}^{k} C_i$ are represented by node N. Every level of the ICT contains only nodes with the same total cost, but different distributions of costs between agents. The root R of the ICT has $f(R) = \langle C_1^*, C_2^*, ..., C_k^* \rangle$ where C_i^* is the cost of an optimal path for agent a_i to its goal disregarding all other agents. A child node of N is generated by increasing the cost of *one* of the agents by one unit while keeping the other agents' costs constant. We will generate k children (an increase in cost for each agent) per node and can avoid duplicates by pruning. Figure 3.4 provides an example of an ICT for a MAPF problem with three agents; red links represent duplicates which are pruned. For each node N, we call upon the low level search to perform a goal check and verify whether there exists a solution that leads to cost vector $f(N)$. The process is repeated until a goal node is found, at some depth Δ in the ICT. Since at every level of the ICT the cost is increased by one, Δ will be equal to the the difference between the total cost of the goal node and that of the root node. Since the branching factor of the ICT (before pruning) is k, the number of nodes generated is $O(k^{\Delta})$. ICTS is therefore exponential in the parameter Δ, and not in the number of agents k. This is a fundamental property of ICTS that makes it very different from A*-based

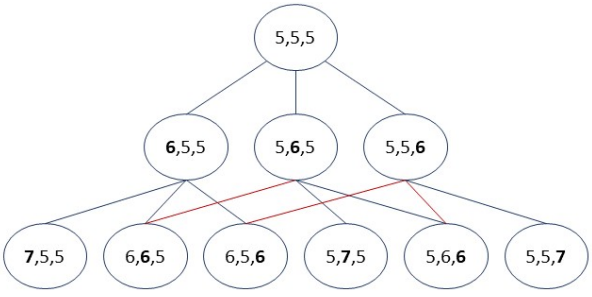


Figure 3.4: Example of an ICT for a three-agent MAPF problem with optimal single-agent path costs $f(R) = \langle 5,5,5 \rangle$. Red links represent duplicate children nodes to be pruned.

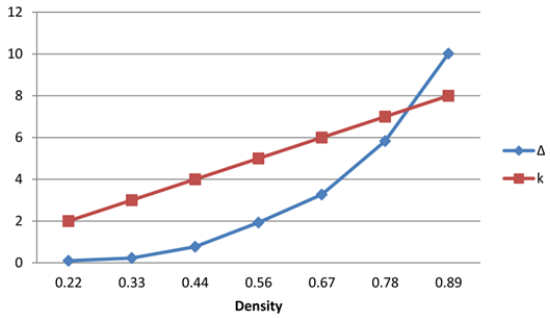


Figure 3.5: Experimental results showing the relationship between Δ (the depth of the goal node in the ICT) and k when using ICTS on a 3x3 grid. Taken from [150].

approaches which are instead exponential in the number of agents. It is important to note that the parameter Δ is a constant for a given MAPF problem, since it is only dependent on the sum of costs of the optimal solution and the sum of the optimal single agent costs. However, it can of course only be known a posteriori since it requires knowledge of the optimal solution cost, and it is difficult to predict in advance although experimental studies have shed light on which factors of MAPF problems most affect Δ. Sharon et al. test ICTS experimentally and conclude that the parameter Δ is affected by i.a. the number of agents k, map topology, and *density* (ratio of number of agents and number of vertices). Figure 3.5 shows the relationship between Δ and k on a 3x3 grid, with the horizontal axis representing the density of the MAPF problem. Results show that while k affects the Δ, Δ can be substantially smaller than k for low densities. However, Δ is superlinear in the density so eventually catches up with k and from a certain density it will be larger than k. This suggests that theoretically ICTS should perform better than A*-based approaches in maps with low density, such as urban UAS environments which are three-dimensional and feature large open spaces.

Low level: At the low level, given a node N the objective is to find a valid solution with cost vector equal to $f(N) = \langle C_1, C_2, ..., C_k \rangle$. If such a solution is found, then it is the goal and the search is halted, else we return to the high-level search and move on to a new node with a different cost vector. A simple approach to perform the goal test would be to list all possible paths for each agent a_i that have cost C_i, and then search for a conflict-free combination of those paths. However, this can be impractical because for an agent a_i the number of paths with cost C_i is often exponential [150]. Sharon et al. circumvent this problem by using a special data structure known as the *multi-value decision diagram (MDD)* [156], which compactly represents all the paths with a specific cost available to an agent, and is structured as follows. Let us denote the MDD for agent a_i containing all paths with cost c as MDD_i^c. The depth t of each node in the MDD corresponds to a location at which agent a_i could be at time t, that is on a path of total cost c from the start to the goal. MDD_i^c always has a single source node which corresponds to the starting location of the agent (level 0 corresponds to time 0), and a single end node which corresponds to the agent being at the goal location at time t_c. The MDD can be built by performing a breadth-first search from the start location to discover each of the nodes reachable within c time steps. Figure 3.6 provides an example of a MAPF problem and its respective MDDs. The low level search is then performed on the *k-agent MDD search space*, which is the cross product of all k single-agent MDDs that are associated with node N in the ICT, with conflicting combinations discarded. Every node in the k-agent MDD space contains a vector of k different locations, one per each agent, at time t. Sharon et al. use a greedy depth first search routine to search the k-agent MDD space, but note that any systematic exhaustive search would in principle be suitable [150].

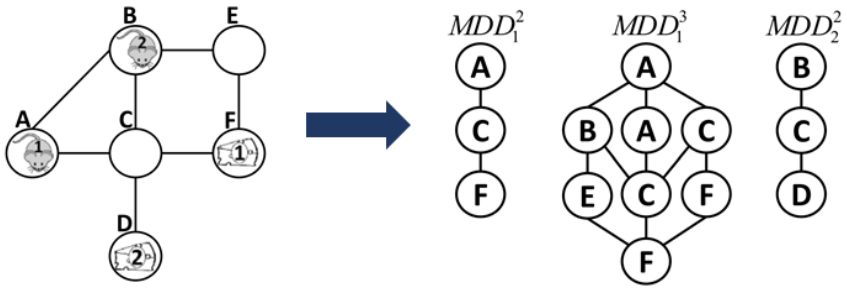


Figure 3.6: Example of a two-agent MAPF problem (left) and associated MDDs for ICTS. Two and three step MDDs are shown for agent 1 and the two step MDD is shown for agent 2. Adapted from [150].

3.4.3. Conflict-Based Search (CBS)

Conflict-based search [149] is among the most widely used optimal search algorithms for the MAPF problem. It has recently been applied to solve a MAPF formulation of the pre-flight conflict detection and resolution for a UAV delivery service provider [72]. The driving principle behind CBS is the decomposition of the MAPF problem into a series of constrained single-agent path finding problems. All agents are first initialized with default and possibly conflicting paths, and then a two-level search is conducted.

High-level: Algorithm 5 shows the pseudocode for the high-level of CBS and MA-CBS (Section 3.4.4). At the high-level, CBS performs search on a constraint tree CT whose nodes contain location and time constraints for a single agent. Every node N in CT contains: (1) A set of constraints (N.constraints), each belonging to a single agent. (2) A set of solutions (N.solution), which is a set of optimal paths for each agent consistent with that agent's constraints. (3) The total cost (N.cost) of the solution (typically calculated as the sum of all individual agents' path costs), which we refer to as the f-value of the node. The root node (R) of CT is created with

an empty set of constraints (lines 2-3), optimal paths are planned independently for all agents via the low-level solver and the associated cost is updated accordingly (lines 4-6). From the root R, the high-level search begins by finding the conflict $\langle a_j, a_i, v, t\rangle$ between a pair of agents that occurs first (line 10), and splitting it into two child nodes of R with one constraint added to each node. One node will have constraint $\langle a_i, v, t\rangle$ specifying that agent a_i cannot be in vertex v at time t, and the other node will have constraint $\langle a_j, v, t\rangle$ specifying the same for agent a_j. Figure 3.7 illustrates this process of splitting a conflict into child nodes, based on the simple explicative MAPF problem provided in [149]. Agents 1 and 2 will first collide at vertex D at time step 2 (conflict $\langle 1, 2, D, 2\rangle$), so the two child nodes generated have constraints $\langle 1, D, 1\rangle$ and $\langle 1, D, 2\rangle$. For each of the child nodes, the low-level search is invoked to generate shortest paths for each agent a_i while satisfying the constraints associated with a_i in node N and the node's f-value is updated (lines 21-26). Conflict detection is then performed again, by iterating through all time steps and all $\langle v, t\rangle$ tuples reserved by the agents. If no conflict is found, the solution has been found and is returned (lines 11-12). Else, new children nodes are created with the appropriate constraint and the process is repeated. Every new child node is added to an OPEN list (lines 27-28), and the high-level search proceeds with a best-first philosophy, meaning that the node in OPEN with the lowest cost is selected as the next to be expanded.

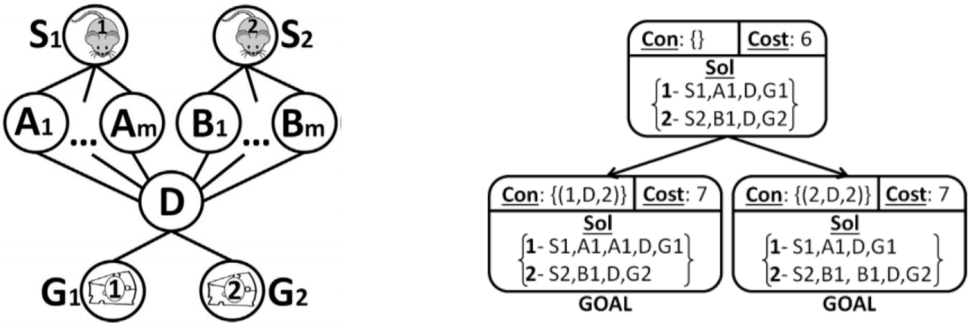


Figure 3.7: Illustration of the split operation in CBS. Taken from [149].

Low Level: The low level search takes an agent a_i and the set of constraints associated to a_i at some node N. It then generates an optimal path for agent a_i that satisfies all constraints, while ignoring all other agents' plans. It is a single-agent pathfinding routine that treats conflicts translated into constraints from the high-level search as hard constraints (e.g. we could think of them as obstacles in the environment). In principle, any optimal single-agent pathfinding routine could be applied. In [149] the authors use A* and choose the spatial length of the shortest path to the goal ignoring all other agents and constraints as their heuristic. In case two low-level A* states have equal f-values, a tie-breaking policy based on a conflict avoidance table [159] is used. States that lead to a conflict with a smaller amount of agents are preferred. This leads to paths that are less coupled and drastically speeds up the low-level search routine, improving total runtime by up to a factor of 2 compared to random tie-breaking.

3.4.4. Improvements to CBS

In its basic form, CBS arbitrarily chooses which conflicts to split into child nodes and arbitrarily chooses paths in the low level. Poor choices can substantially increase the size of the CT and slow down the high-level search, reducing the overall performance of CBS. Four separate improvements to CBS were presented to tackle this problem [54]. The variant of CBS which combines all four improvement is termed Improved CBS (*ICBS*) [19], and has been shown to be substantially more efficient than basic CBS.

Improvement 1: Meta-Agent CBS (MA-CBS)

MA-CBS [148] [149] was designed as a framework to mitigate the worst-case performance of CBS. CBS has been shown to perform very poorly for sets of agents whose paths are strongly coupled, i.e. there is high conflict density between these agents. The CT grows exponentially with the number of conflicts so these cases CBS will tend to expand a substantially larger number of nodes than A*-based approaches (Section 3.4.1).

MA-CBS solves this problem by adding the possibility to *merge* agents whose paths are strongly coupled into a single *meta-agent* (lines 13-20). The meta-agent is considered a single agent by the high-level routine of CBS, but its state is a vector with the locations of the merged agents. Upon merging two agents a_i and a_j, their constraints are first combined (lines 14-15) and then the low-level search is conducted only for the new meta-agent $a_{(i,j)}$ (line 16) and the node's f-value is updated accordingly (line 17). It is not necessary to perform the

Algorithm 5 CBS and MA-CBS

```
1:  procedure CBS(agents A, graph G)
2:      R = new_node
3:      R.constraints = ∅
4:      for a_i ∈ A do
5:          R.solution = low_level_search(a_i)
6:      R.cost = sum_individual_costs(R.solution)
7:      insert R into open
8:      while OPEN is not ∅ do
9:          N = lowest cost node from OPEN
10:         C = search_first_conflict(N)
11:         if C is ∅ then
12:             return N.solution                          ▷ N is the goal
13:         if shouldMerge(a_i, a_j) then                  ▷ If block executes only in MA-CBS
14:             a_(i,j) = merge(a_i, a_j,v,t)
15:             Update N.constraints
16:             N.solution = low_level_search(a_(i,j))
17:             N.cost = sum_individual_costs(N.solution)
18:             if N.cost < ∞ then                         ▷ A solution was found
19:                 Insert N back into OPEN
20:             continue                                   ▷ Go back to the while condition
21:         for a_i ∈ C do
22:             P = new_node()
23:             P.constraints = N.constraints + (a_i, v, t)
24:             P.solution = N.solution
25:             P.solution = low_level_search(a_i)
26:             P.cost = sum_individual_costs(P.solution)
27:             if P.cost < ∞ then                         ▷ A solution was found
28:                 Insert P to OPEN
```

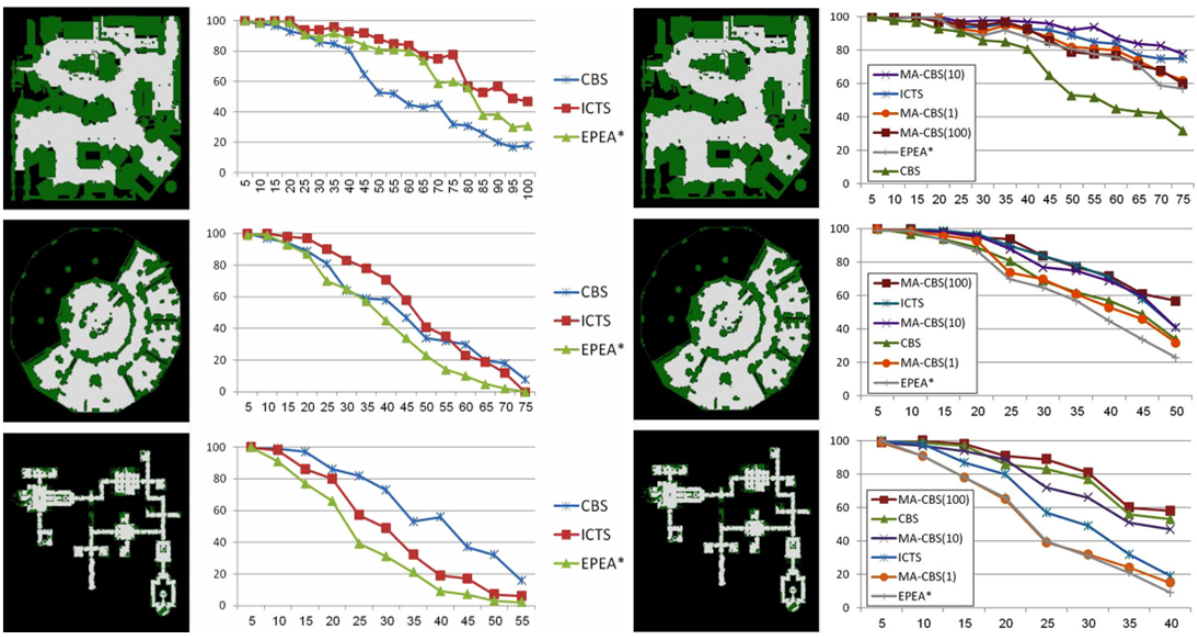


Figure 3.8: Experimental evaluation presented in [149] of CBS (a) and MA-CBS (b) compared to ICTS and EPEA* on three standard maps from Dragon Age: Origin [161]. The horizontal axis shows the number of agents and the vertical axis the success rate (in % of problem instances solved).

low-level search again for all agents since their plans are not influenced by the merge action. Note that at the low-level, the paths for the merged agents will be planned concurrently so the low-level solver will have to be a MAPF solver rather than a SAPF solver as in basic CBS. As the high-level search progresses, a meta-agent can be merged again with other agents or meta-agents. The merge action is only executed if the *shouldMerge* condition (line 13) is satisfied for a pair of (meta-) agents a_i and a_j. The condition is set according to a merge policy which aims to determine whether two agents' plans are coupled strongly enough to warrant a merging action. Sharon et al. [149] use a simple policy which proved effective in their experiments. Two agents a_i and a_j are merged into a meta-agent if the amount of conflicts between them is found to be greater than some quantity B, known as the *conflict bound parameter*. We denote the use of MA-CBS with this policy as MA-CBS(B). Note that MA-CBS(∞) is equivalent to basic CBS since no merge action will ever be executed, and MA-CBS(0) is equivalent to just using the search algorithm deployed at the low level to solve the entire MAPF instance since all agents will be merged.

Sharon et al. evaluate MA-CBS experimentally on three maps from Dragon Age: Origin [161], the results are shown in Figure 3.8. Three different maps are considered: map *den520d* (top) features a large amount of open space and no bottlenecks, map *ost003d* (middle) has some open spaces and bottlenecks and map *brc202d* (bottom) has very limited open space and abundant bottlenecks. The solver used for the low-level search is EPEA* across all experiments (EPEA* is therefore equivalent to MA-CBS(0)). Results show that MA-CBS(B) with intermediate B values ($0<B<\infty$) solves more problem instances than basic CBS, EPEA* and in most cases ICTS. Basic CBS performs poorly on *den520d* (top) due to the large amount of open spaces, which allow for the possibility of large amounts of conflicts to occur in many different locations and result in a prohibitively large CT. Merging on this map is very beneficial, which is why MA-CBS(10) is the best performing algorithm. As we decrease the amount of open space and increase the amount of bottlenecks, the performance of CBS gets comparatively better and on *brc202d* (bottom) it even outperforms MA-CBS(10). The best performing algorithm on *ost003d* (middle) and *brc202d* (bottom) is the MA variant with the most conservative merging policy, MA-CBS(100). Merging is less beneficial than in *den520d* (top), but nonetheless improves the performance of CBS with a loose enough B. The conclusion to be drawn is that, while merging can improve the performance of CBS, it is important to choose a suitable merge policy based on the characteristics of the MAPF problem at hand. For environments with abundant open spaces and little bottlenecks, such as urban UAS environments, low values of B are more efficient.

Improvement 2: Merge and Restart (MR)

A simple improvement on MA-CBS introduced by Boyarski et al. [19] consists in moving merge actions to the root node of the CT. The rationale is that, if we had known a priori that two agents were going to be merged, it would have been more efficient to perform the merge action ab-initio at the root of the CT rather than at the node in which we discovered it. The insight is implemented as follows. Once a decision to merge a set of agents has been taken at some node N of the CT, we delete the current CT and essentially *restart* the search by creating a new CT in which at the root node those agents are already merged into a meta-agent. This scheme is called *Merge and Restart (MR)* and, while very straightforward and simple to implement, it has been shown to significantly reduce computational effort in MA-CBS.

Improvement 3: Prioritization of Cardinal Conflicts (PC)

A conflict $C = \langle a_i, a_j, v, t\rangle$ between a pair of agents at some CT node N can be classified in three types [54]:

- *Cardinal*: if adding any of the two constraints that follow from conflict C ($\langle a_i, v, t\rangle$ or $\langle a_j, v, t\rangle$) to N and then performing the low-level search for the constrained agent results in a path of higher cost for that agent compared to the cost in N, therefore increasing N.cost. This means that for both agents a_i and a_j, all feasible optimal paths to their goals required traversing vertex v at time t.
- *Semi-cardinal*: if adding one of the two constraints that follow from C to N increases N.cost, but the other constraint does not change N.cost. This means that for one of the agents, all feasible paths to its goal required traversing vertex v at time t, but for the other agent there is at least one alternative that does not require traversing vertex v at time t.
- *Non-cardinal*: if neither of the two constraints that follow from C increase N.cost. This means both agents have optimal paths to their goals that do not go through vertex v at time t.

In CBS+PC, when some node N is expanded, all its conflicts are examined and if a conflict is found to be cardinal, it is chosen for splitting. Let us suppose that N.cost=c, then the children nodes will both have cost greater than c. If there exists another node in OPEN with cost c, then it will be chosen next for expansion. Otherwise, if we had chosen a semi-cardinal or non-cardinal conflict, we might first have to expand the children nodes and could thereby end up generating a large subtree of N with the same cost c, carrying the cardinal conflict

through every node until we choose it and resolve it [19]. This may cause many more nodes to be expanded than necessary and slow down the high-level search routine.

Improvement 4: Bypassing conflicts (BP)

In case a non-cardinal or semi-cardinal conflict at node N is chosen for expansion, the BP routine tries to avoid splitting the conflict by changing the path of one of the agents instead. BP looks ahead to the children of N. If the solution in one of the children of N includes a different path with the same cost for the conflicting agent but without the conflict, then BP takes this path and incorporates it into N without needing to generate any child nodes. This process can reduce the size of the CT substantially and improve efficiency in the high-level search.

3.5. Rule-based solvers

These solvers specify agent movement rules for several scenarios and usually do not involve exhaustive search procedures. They are structured in such a way as to guarantee fast convergence to a solution by trading off solution cost, and are therefore known to often return solutions which are far from optimal [54]. They also usually only work on graphs with special properties, which can make them inapplicable to a wide range of problems.

Recent examples of algorithms in this class are the tree-based agent swapping strategy (TASS) [82] and Push and Swap and its variants [142][97][39]. Both algorithms run in polynomial time but are not optimal and are only complete on specific types of graphs. TASS is complete only for tree graphs (bipartite and acyclic), a requirement which we deem far too restrictive in order to apply it to MAPF within the context of urban UAS operations, since it would not be possible to represent a realistic operational environment in which UAVs need to reach goal nodes (to e.g. deliver a parcel) and then return to the starting position (e.g. an operator's hub or charging station) this way. Push and Swap, although it has been experimentally shown not to fail on general graphs in which there are at least two unoccupied vertices [23], is formally only complete for tree graphs and therefore also deemed unsuitable. Another example of a class of frequently studied rule-based solvers is BIBOX [164] and its variants, namely diBOX [18] and BIBOX-θ [165]. While these solvers have been shown to be highly efficient and complete on biconnected graphs which is also highly restrictive, since already a traditional grid-like environment in which no diagonal moves are allowed fails to fit this description. A rule-based solver which is shown to be complete on a useful set of graphs, namely those with at least two unoccupied vertices, is a recent variant to Push and Swap called Push and Rotate [39], which is considered the most general rule-based solver available. However, it does not provide any performance guarantees and solutions are typically far from optimal on large problems [54].

3.6. Search-based suboptimal solvers

Search-based suboptimal solvers are typically concerned with finding relatively good solutions (often close to optimal), typically trading off completeness. There are two distinct classes of suboptimal search approaches. The first is a decentralized A* framework called *Hierarchical Cooperative A** whereby agents plan paths individually and different approaches can be implemented for conflict resolution. The second class is made up of relaxed version of popular optimal search-based algorithms.

3.6.1. Hierarchical Cooperative A* (HCA*)

The Hierarchical Cooperative A* framework was first introduced by Silver [153], and the driving principle behind it is to decouple the MAPF problem into a series of single agent searches. Each agent is planned individually according to a predefined order, which is to be chosen sensibly based on the characteristics of the problem at hand. Upon planning their paths to their goals, the agents store the states they will occupy while traversing the paths into a system-wide *reservation table*. The structure of the reservation table should be tailored to the specific MAPF problem and made flexible enough to account for e.g. agents of different sizes, speeds, or dynamic definitions of conflicts (for instance varying separation requirements for UAVs in urban areas) if required by the problem. Once paths are in the reservation table, they are blocked and other agents are forbidden from occupying the same states. A simple way to implement a reservation table in a classic three-dimensional MAPF problem is as a four-dimensional grid $\langle x, y, z, t\rangle$ where agents reserve combinations of space and time. On problems with low density only a relatively small proportion of entries in the grid will be reserved, so an efficient way to implement it is as a hash table, hashing on a randomly distributed function of the $\langle x, y, z, t\rangle$ key [153]. In principle, any admissible heuristic can be used to guide the search in HCA*, with distance (Manhattan or Euclidean) to the goal being the most commonly used.

An online variant of HCA* called *Windowed Hierarchical A* (WHCA*)*[153] only performs the cooperative search to a fixed depth w, meaning that other agents' reservations are only considered within the planning window associated to w and are ignored in the rest of the search. Cooperation is performed in a rolling way in runtime, as the agents execute their paths. Each agent first plans a partial conflict-free path within the window (ignoring other agents outside the window) and once the agents have reached a set point in their paths (e.g. halfway) the window shifts forward and new partial paths are computed if conflicts with other agents' reservations are found. Every time the window shifts forward, the agents' priority order can be shuffled to avoid a biased distribution of the reservation table among the agents. A recent enhancement of WHCA* [15] only places windows around conflicts which are discovered in an offline planning phase and agent prioritization within these windows is established via a mechanism called *conflict oriented prioritization*, which assigns an order to the agents in a conflict window based on an estimate of which order will result in the minimal sum of costs upon replanning. All possible orderings are considered, and for each order the cost is estimated heuristically by summing all the resulting individual agent path costs while ignoring all future conflicts. In a window for conflict $c = \langle v, t, A \rangle$, where A is the set of conflicting agents, we only need to check $|A|!$ combinations and not all $k!$ possible orderings as the other agents will not replan. So this approach works well in practice since $|A|$ is usually a relatively small number, with the exception of bottlenecks in which several agents can conflict at the same time step. In principle, other prioritization mechanisms could be used, for example based on direction maps [75] or negotiation mechanisms in settings where the agents are self-interested.

One of the main weaknesses of HCA*-based approaches is that they are impractical to use on very dense problems, in particular those with bottlenecks. This is because one agent's reservation can block all paths available for the other agent to reach its goal, causing a *deadlock*. HCA*-based approaches are therefore *not complete* since in the presence of deadlocks there may not exist a priority order that allows all agents to execute their plans and the algorithm cannot guarantee that a solution will be returned if it exists. They also provide no optimality guarantees and can therefore return solutions which are far from optimal, particularly in dense environments with a large amount of potential conflicts. In these environments we expect most of the agents' optimal paths to be substantially different from their own single-agent optimal paths, and therefore more sophisticated cooperation mechanisms should perform better. However, in environments with large open spaces HCA* could prove an interesting alternative because it is highly efficient and flexible, and the presence of deadlocks is unlikely. It is a distributed MAPF approach and therefore lends itself well to problems in which agents have different cost functions or interests, which may be the case for UAVs in the shared low altitude airspace. It would be interesting to compare performance to optimal search-based solvers and verify experimentally whether an HCA*-based approach could be a useful substitute by analyzing the trade-off between computing time and solution cost in an urban UAS scenario. The lack of completeness, however, remains a major point of concern and further work should be conducted into understanding which problem instances will cause deadlocks and how to circumvent these, should this algorithm be chosen for implementation.

3.6.2. Suboptimal variants of CBS

While CBS has and its variants perform very well on a wide range of problems as discussed in Section 3.4.3, they are still exponential in the number of conflicts and therefore do not scale very well in environments where a large number of conflicts are possible. Two main suboptimal variants were devised to improve the speed of CBS in these scenarios: *Greedy CBS* and *Extended CBS*.

Greedy CBS (GCBS)

GCBS is a suboptimal and complete variant of CBS in which the main idea is to relax the optimality constraint in both the high-level and low-level search of CBS, by using a greedy approach in which nodes that are most likely to produce a valid but possibly suboptimal solution quickly are preferred [11]. At the high-level, GCBS chooses the node that is seemingly closest to the goal, as measured by a *conflict heuristic* h_c whose objective is to prefer nodes that will generate less conflicts down the line and are therefore more likely to lead to a goal node. While several choices of h_c are possible, Barer et al. investigate the following ones experimentally:

- h_1 **Number of conflicts**: number of conflicts encountered at the CT node
- h_2 **Number of conflicting agents**: number of conflicting agents that have at least one conflict at the CT node
- h_3 **Number of conflicting pairs**: number of conflicting pairs of agents at the CT node
- h_4: **Vertex cover**: vertex-cover of a graph in which the agents are the nodes and edges only exist if agents are conflicting.
- h_5: **Alternating heuristic**: existing work shows that search performance can often be improved by cycling through different heuristics if multiple ones exist, rather than using the same one throughout the

problem [136] [172]. In h_5, the above heuristics are alternated in round robin fashion.

While the best performance is achieved by using h_5 and the fastest heuristic is h_1, Barer et al. note that the overall performance of GCBS does not vary much with the choice of heuristic. It is suggested to choose h_3 because it offers the best tradeoff between simplicity and performance, and is robust across different kinds of problems. All heuristics above are not admissible or bounded admissible, meaning that as noted GCBS will not generate optimal or bounded suboptimal solutions.

To relax the low-level search, simple suboptimal solvers (such as Weighted A*) should be avoided, because they may return longer paths with a large number of high-level conflicts. Rather, we should use conflict heuristics to relax the low-level. Instead of returning the optimal single-agent path that does not violate any of the constraints imposed by the high-level, we perform a best-first search in which paths with a lower number of conflicts are preferred. This way, the low-level search will sacrifice optimality but return paths with the minimum number of conflicts with already planned agents, which speeds up the high level.

Bounded suboptimal extensions to CBS

Bounded CBS (BCBS) and Enhanced CBS (ECBS) [11] are complete and bounded suboptimal extensions of GCBS that use the concept of *focal search* at the high and low level.

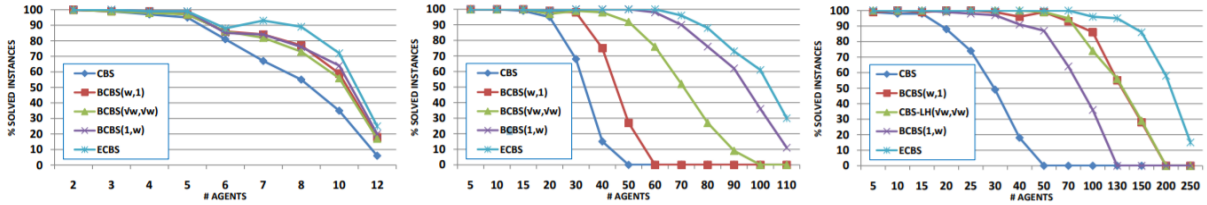
Focal Search: In focal search there are two sets of nodes. The first is the regular OPEN list used in A*-based approaches. The second is the FOCAL list which contains a subset of the nodes in OPEN. The search is based on two arbitrary functions, f_1 to determine which nodes of OPEN are to be inserted in FOCAL and f_2 which guides the search and chooses which node in FOCAL to expand next. This is denoted as $focal\_search(f_1, f_2)$. The FOCAL list is constructed by including all nodes n in OPEN that satisfy $f_1(n) \leq w * f_{1,min}$, where w is a user-defined suboptimality factor and $f_{1,min}$ is the minimum value of f_1 across all nodes in OPEN. Provided that the heuristic f_1 is admissible, focal search guarantees that the solution will have cost no larger than $w \cdot C^*$, where C^* is the cost of an optimal solution.

BCBS: In BCBS, the high-level uses $focal\_search(f_h, h_c)$ on the CT, where f_h is the cost of the CT node n and h_c is a conflict heuristic used in GCBS. At the low level, $focal\_search(f_l, h_c)$ is applied, where $f_l(n)$ corresponds to the regular A* cost $g(n) + h(n)$ and h_c is again one of the conflict heuristics from GCBS. We use the notation BCBS(w_h,w_l) for the use of BCBS with a bound w_h on the high-level focal search and a bound w_l on the low-level focal search. If any of the weights is equal to 1, we are effectively not using focal search for that level but are rather running an optimal search since FOCAL only takes the minimum cost node(s) from OPEN. Barer et al. show that for any $w_h, w_l > 1$ the cost of the solution is bounded above by $w_h \cdot w_l \cdot C^*$, so if the objective is to generate a solution within a factor w of the optimal cost, then we can select any combination of weights such that $w_h \cdot w_l = w$. The case BCBS(∞,∞) corresponds to GCBS since FOCAL will take all nodes from OPEN, and consistently with the definition above is unbounded since the product of the weights is infinitely large.

ECBS: ECBS is considered the state of the art suboptimal CBS solver and it extends the idea of BCBS further by providing more flexibility in the focal search. ECBS(w) uses the same low-level as BCBS(1,w), with w therefore referring to the bound for the low-level focal search. For a given CT node n, let $f_{min}(i)$ be the lower bound on the cost of an optimal consistent path for an agent a_i being planned in the low-level. Then $LB(n) = f_{min}(n) = \sum_{i=1}^{k} f_{min}$ is the lower bound on the cost of n and the upper bound is $w \cdot LB(n)$ since the low-level will only consider paths with cost of at most $w \cdot f_{min}$. Every time the low-level search is run for a CT node n, it returns $n.cost$ and $LB(n)$. $LB(n)$ is then used to populate the FOCAL list as follows: FOCAL = $\{n | n \in \text{OPEN}, n.cost \leq LB \cdot w_L\}$, where LB is the minimum $LB(n)$ across all nodes n in OPEN. LB is clearly the lower bound on the solution to the entire problem ($LB = C^*$), given that it corresponds to the lowest cost solution that could be possibly generated by the low-level for the best node in n. Therefore, every node in FOCAL has cost within factor w times the cost of an optimal solution, and the cost of a solution returned by ECBS is at most $w \cdot C^*$.

Barer et al. test the bounded suboptimal variants of CBS experimentally with varying bounds in a range of domains. Figure 3.9 shows the results for 5x5 (a) and 32x32 (b) grids, and a DAO map (c) which is substantially larger and therefore the MAPF problems tested in that domain are much less dense than on the smaller grid maps [11]. Results are only shown for the best values of w tested, which vary significantly across the different domains. In the DAO map, only very small values of w led to faster solution times, whereas in the smaller grid environments only large values of w showed substantial improvements over CBS (which is equivalent to BCBS(1,1)). Results show that with the appropriate choice of w ECBS is able to solve more instances than all other variants across the three domains tested, scales much better than CBS with the number of agents and is able to maintain high solution rates even at relatively high densities. On the 32x32 grid for instance, ECBS(1.1) is able to solve nearly 100% of the instances at a density of ≈ 6% (60 agents) and over 60% of the instances

at a density of ≈ 10% (100 agents)[1]. ECBS is therefore an attractive alternative to the search-based solvers described in section Section 3.4 if the MAPF problem at hand proves too hard to be solved optimally within reasonable computing time. While Barer et al. suggest that low values of w are more effective in low density environments, such as environments for urban UAS MAPF, there are few experimental studies that actually describe the nuances of the behavior of ECBS with varying w. Since it is clear that the optimal value of w is largely dependent on the characteristics of the specific MAPF problem, it is suggested to experiment with a grid of w values and analyze the performance of ECBS in terms of solved instances and computing time, should this algorithm be chosen for implementation.



(a) 5×5 - 20% obstacles. $w = 1.5$ (b) 32×32 - 20% obstacles. $w = 1.1$ (c) DAO Map - BRC202. $w = 1.01$

Figure 3.9: Experimental comparison of bounded CBS variants in different domains [11].

3.7. Hybrid solvers

A class of suboptimal MAPF algorithms consists of a hybrid between search-based and reduction-based algorithms, combining search routines with movement rules. For example, hybrid approaches for grid-based environments have been developed in which each vertex is associated to a set of directions in which the agents are forced to move, with the goal of limiting the chance of conflicts and reducing the branching factor of each node and therefore the dimension of the state space [74] [180]. These approaches are designed to be highly efficient and require good a priori knowledge of the domain and the problem in order to generate effective movement rules, otherwise the solutions should be expected to be far from optimal. In general, these approaches are not complete as they may run into deadlocks, particularly in the presence of bottlenecks. Given the complexity associated to establishing movement rules for each vertex in the large, open three-dimensional domains that characterize urban UAS problems, and the lack of completeness, these algorithms in their general form are not deemed suitable. The idea of associating movement rules to specific areas or scenarios, however, is a useful takeaway from these approaches. For instance, it could be useful to establish specific movement rules for UAVs in proximity of high-density areas such as charging stations, pick-up or delivery locations.

Another example of a hybrid MAPF approach is called *MAPP* [181]. The main logic behind it is to compute a main path P_i for each agent along with an alternative sub-path for each pair of successive steps of the agent. Whenever an agent's path is found to be blocked by another agent, it must avoid the conflict by switching to one of its alternative paths. However, this approach is only complete on *slideable* graphs, meaning that: (1) given three nodes n_{i-1}, n_i, n_{i+1} on a path P there must always exist an alternative path A that connects n_{i-1} and n_{i+1} while not not going through n_i. A bottleneck is a good example of a common MAPF situation that does not meet this property. (2) The first node in any agent's path must be empty, and (3) no agent's goal location can interfere with the main or alternative paths of other agents. This definition is highly restrictive and no variants of MAPP have been proposed to generalize the solver to a broader class of problems.

3.8. Comparative evaluation of MAPF solvers

Table 3.1 presents a summary of the findings on the MAPF solver classes discussed in this chapter. State of the art techniques in each class are identified and compared further by means of a tradeoff. The tradeoff is based on the following equally weighted criteria, which are given a score between 1 and 3 roughly corresponding to "Low", "Medium" and "High" in comparative terms unless an alternative description is given:

- **Completeness:** since this is a highly desired property of a MAPF solver, a score of 3 is given if the solver is complete on classes of graphs that are general enough to represent a realistic grid-like urban environment and a score of 0 if it isn't.
- **Optimality:** the optimality guarantees of the algorithm translate into the following scores: (3) optimal, (2) bounded suboptimal with tunable bounds, (1) bounded suboptimal with fixed bounds, (0) no solution quality guarantees.

[1] recall that density in a MAPF problem is the number of agents divided by the number of vertices. The 32x32 grid has 1024 vertices, so the density for 60 agents is $\frac{60}{1024} \approx 6\%$ and for 100 agents it is $\frac{100}{1024} \approx 10\%$.

		Solver class	State of the art technique	Characteristics
OPTIMAL	Search-based	A*-based	A*+OD+ID and EPEA*	• Exponential in number of agents • Best in dense maps with small number of agents • Well-studied, relatively easy to implement and functions as low-level solver for several other algorithms
		ICTS	ICTS	• Two-level search approach that is exponential in δ, so effective in low density maps • δ is smaller than the number of agents for low density maps, larger for high density maps. See Figure 3.5 for more details • Not as extensively studied as other search-based optimal approaches, less improvements and extensions available
		CBS	(MA/I)-CBS	• Exponential in number of conflicts, so highly effective in structured maps and in the presence of bottlenecks • Considered the most mature coupled MAPF solver, extensively tested and several improvements available
	Reduction-based	SAT	MDD-SAT	• Reduces MAPF to boolean satisfiability problem • Extremely fast for small, dense graphs but does not perform well in large open spaces • Very limited flexibility in the cost function
(BOUNDED) SUBOPTIMAL	Search-based	CA*	HCA*	• Most effective and well-studied decoupled MAPF solver • Highly computationally efficient and suitable for online use • No optimality or completeness guarantees, but empirically shown to perform well in large open spaces and several deadlock resolution mechanisms effectively implemented in the literature
		Relaxed optimal solvers	ECBS	• State of the art suboptimal variant of CBS based on focal search • Optimality bound can be specified by the user, but computational time scales non-trivially with the bound
	Rule-based	Push & Swap and variants	Push & Rotate	• Rule-based methods are generally only effective for specific kinds of graphs • Push & Rotate is only complete for graphs in which at least 2 vertices are free at all times • Performs poorly on large maps, and solutions are typically far from optimal
	Hybrid		MAPP	• Only complete on slideable graphs, highly restrictive representation that does not translate to urban UAV delivery setting

Table 3.1: Summary of findings on MAPF solvers.

- **Computational efficiency:** is a measure of the running time required for the solver to return a conflict-free solution, which is related to its complexity and the degree of parallelization that it allows.
- **Flexibility:** reflects how easily the solver can be adapted to a different type of MAPF problem and the extent to which it can retain its properties in the process. This includes the amount of modifications required to change the cost function representation, and how well performance generalizes to different problems and environments, with e.g. varying size, obstacle density, number of agents, heterogeneous agents.
- **Scalability:** is a measure of the extent to which the solver is able to retain its computational efficiency as the number of agents and the size of the environment increase.
- **Suitability for online use:** reflects how suitable the solver is to be used as part of an online planning mechanism in the UAV delivery coordination context. This is not only related to the robustness and the computational efficiency of the solver, but also to the degree of centralization in the planning. Decoupled MAPF solvers are more naturally suitable for online use because a single UAV can react to changes in local knowledge by replanning its own path without affecting the other agents' plans. In a standard coupled MAPF scenario, instead, all agents would have to replan their paths if a single UAV wishes to updated its own path due to a change in local knowledge.
- **Performance in urban-like environment:** relates to how well the solver performs, in terms of success rate, in an environment that is representative of the urban low-altitude airspace. Such an environment features large open spaces and low fixed obstacle density, although in high-density applications such as deliveries the number of agents can be large.
- **Implementation complexity:** is a measure of how complex the solver is to implement, including how memory-efficient it is.

	A*+OD+ID	EPEA*	ICTS	(MA/I)-CBS	MDD-SAT	HCA*	ECBS	Push & Rotate	MAPP
Completeness	3	3	3	3	3	0	3	3	0
Optimality	3	3	3	3	3	0	2	0	0
Computational efficiency	2	2	1	1	1	3	3	2	1
Flexibility	2	2	2	2	1	3	2	1	1
Scalability	1	1	1	2	1	3	2	1	2
Suitability for online use	1	1	1	1	1	3	1	2	2
Performance in urban -like environment	1	1	3	2	1	2	2	1	1
Implementation complexity	2	2	2	2	2	3	1	2	2
Maturity	3	2	2	3	2	3	2	2	1
Total	**18**	**17**	**18**	**19**	**15**	**20**	**18**	**14**	**10**

Table 3.2: Comparative evaluation of MAPF solvers.

The results of the tradeoff are shown in Table 3.2. The reduction-based (MDD-SAT), rule-based (Push & Rotate) and hybrid (MAPP) solvers receive the lowest total score. This is expected because they are tailored to specific types of MAPF problems, for which they are complete and typically substantially more efficient than search-based approaches. However, their properties do not translate well to less structured problems and large, open environments. The state of the art optimal search-based approaches have similar properties which result in comparable overall scores, with (MA/I)-CBS slightly outperforming the other solvers. It beats the A* variants since the latters's complexity is exponential in the number of agents and they therefore have poor scalability and performance in an urban-airspace like environment, in which the agent density may be high especially for delivery applications. (MA/I)-CBS outperforms ICTS due to higher maturity, since it is much more thoroughly studied and tested. In case the runtime of CBS is found to be too high for realtime use, then the solver could be enhanced via focal search to the suboptimal ECBS variant, which gives tractability over the solution quality-computational time tradeoff by allowing user-defined suboptimality bounds. Given the added complexity associated to the ECBS variant, it would be recommended to use it only if it was empirically shown to be necessary to reduce the computational time of the MAPF solver.

The decoupled suboptimal HCA* outscores (MA/I)-CBS and obtains the highest score among all solvers. It is therefore deemed the most applicable MAPF solver to the UAV delivery coordination problem. While it is not complete and provides no performance guarantees, it is highly computationally efficient and has been shown to perform well in large, open environments where the probability of deadlocks is low. Several deadlock resolution mechanisms have also been successfully implemented in the literature. Among the main strengths of HCA* is that it is highly suitable for online use, since changes in the local knowledge of a UAV can be acted upon through path replanning without affecting the other UAVs' plans.

A key observation is that HCA* also fits very well with the sequential auction-based task allocation framework chosen in Chapter 2. Suppose we are at some round in which task t is on auction and let $S = \{S_1, ..., S_n\}$ be the allocation at the beginning of the round, where S_i is the subset of tasks allocated to agent a_i. Suppose winner determination is conducted and agent a_j is found to be the winner of task t. The auctioneer can leverage the decoupled nature of HCA* and simply pass to a_j the planned paths of the other agents. The agent a_j can then add task t to its local schedule and plan a path through its new task set $S_j \cup t$ while treating the other agents' paths as obstacles, as per the HCA* approach. Since this does not change the plans of the other agents, the act of assigning task t does not affect the cost of S. If we were to use a search-based optimal solver such as CBS, instead, we would need to change the paths of multiple (possibly all) UAVs upon allocating task t. This would change the distance travelled to execute tasks allocated previously, and thereby change the cost of the allocation S. This contradicts the hill-climbing logic that sequential auction-based task allocation mechanisms are built on and leads to a loss of tractability over the final cost of the allocation. The simple theoretical guarantees described in [89] and explained in Section 2.6, for instance, would no longer hold. To provide performance guarantees using a coupled solver like CBS, bidding rules that take into account the effect of path planning on previous allocations would need to be devised. This would require each agent to solve the path finding problem for the entire system before bidding, which is impractical from a computational perspective and would also require the agents to have full knowledge of each others' schedules at all times.

4
Research Proposal

In this chapter, we propose a research framework to address the UAV delivery coordination problem based on the results of the literature study. We begin by defining the research objective in Section 4.1, which we translate into a series of four research questions discussed in Section 4.2. Finally, in Section 4.3 we propose a work breakdown to answer the research questions in a structured manner. Figure 4.1 provides an overview of the research framework described in this chapter.

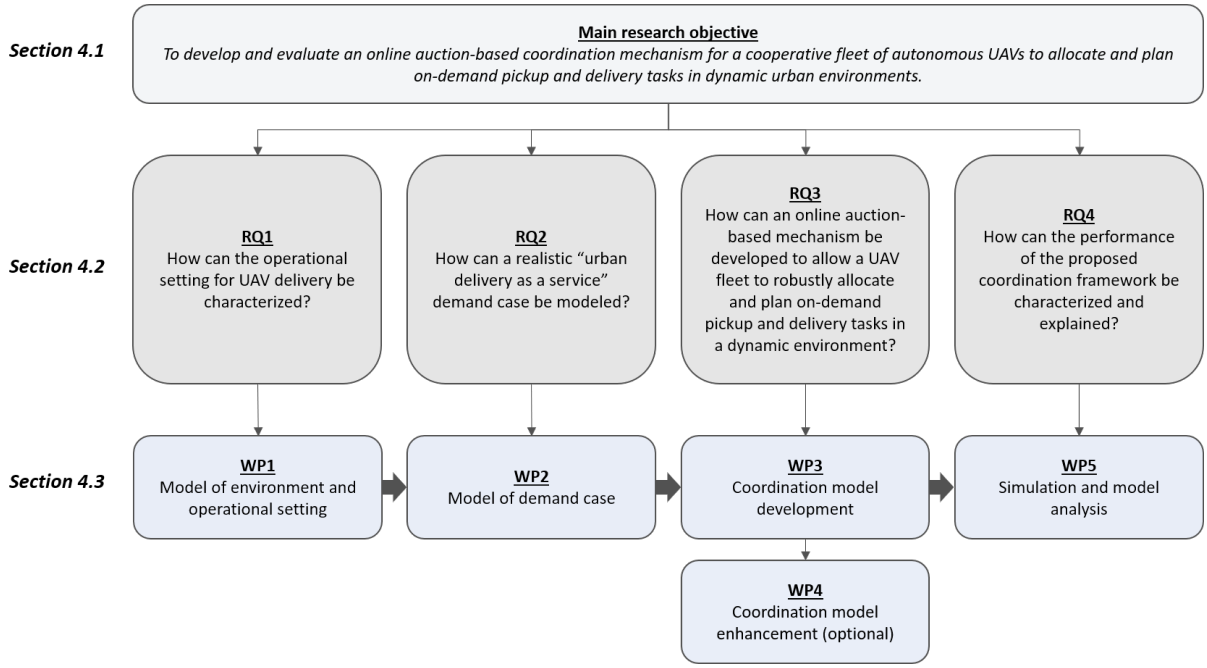


Figure 4.1: Research framework and work breakdown.

4.1. Research objective

Based on our analysis of urban UAV delivery in Chapter 1, we choose to focus on the coordination problem because it is considered a fundamental technical challenge to solve in order to enable UAV deliveries in urban areas. The focus is therefore on the development of a coordination framework for a fully cooperative fleet of autonomous delivery UAVs. Not only will such a framework incentivize operators to make economical use of the airspace, but it will also contribute to the development of UTM technology by allowing for a more accurate modeling of delivery operations which are considered the most high density application. The problem also poses novel challenges in multi-agent coordination, since it is substantially less structured and features much more uncertainty than multi-agent routing or pickup and delivery problems already explored in the literature. UAVs need to coordinate online and be able to accommodate changes in the characteristics of the tasks (e.g. delays at the pickup point changing the start time of a task) as well as knowledge regarding the environment (e.g. change in the location of no-fly zones or other airspace constraints communicated from the ANSP). In light of this, the research objective is:

To develop and evaluate an online auction-based coordination mechanism for a cooperative fleet of autonomous UAVs to allocate and plan on-demand pickup and delivery tasks in dynamic urban environments.

4.2. Research questions

In order to achieve the main research objective, the following four core sub-questions are identified and broken down further:

1. *How can the operational setting for UAV delivery be characterized?*
   (a) What are the key characteristics of the urban low-altitude airspace?
   (b) How can a typical urban environment be approximated and modeled?
   (c) What assumptions can be made regarding the locations of the operator's charging hubs?
   (d) What assumptions can be made regarding the structure of the airspace and location of no-fly zones?
   (e) What communication architecture will exist between the operator and the UAVs?
   (f) What are realistic performance limitations (e.g. range, top speed) for the UAVs?
2. *How can a realistic "urban delivery as a service" (see Section 1.1) demand case be modeled?*
   (a) How can the demand for deliveries throughout the operating window be realistically simulated?
   (b) Can open-source data regarding on-demand deliveries be identified and used to build the demand case?
   (c) Can census data be leveraged to determine a distribution of population density and thereby of customers?
3. *How can an online auction-based mechanism be developed to allow a UAV fleet to robustly allocate and plan on-demand pickup and delivery tasks in a dynamic environment?*
   (a) How can auction-based task allocation mechanisms be extended to allow the UAVs to exchange enough information to plan conflict-free trajectories?
   (b) How should the UAVs value the delivery tasks and which bidding rule should be used?
   (c) How should the UAVs perform local planning and scheduling of their tasks, and how should battery life constraints be accounted for in this process?
   (d) How can real-time global replanning be leveraged to improve the overall allocation cost?
      - Can task swaps be used to achieve a good tradeoff between solution quality and computational cost? If so, how should these swaps be performed?
      - Does clustering the tasks based on spatial and/or temporal characteristics improve the above tradeoff? If so, which clustering procedures work best?
   (e) Can multi-agent learning be used to improve system performance or bring down the computational cost?
      - Can the UAVs learn to "follow the demand" and thereby incorporate future expectations into their valuation of a task?
      - Can the UAVs learn which tasks to bid on and which not to? And can this be used to reduce the computation and communication cost?
   (f) How can the coordination framework be made robust to UAV failures or losses of communication?
4. *How can the performance of the proposed coordination framework be characterized and explained?*
   (a) What global performance indicators are most relevant to measure the performance of a UAV delivery system?
   (b) Does the proposed framework provide any guarantees with respect to solution quality? If so, which and how can these be proven?
   (c) How can the robustness of the framework be quantified and tested?
   (d) How well does the system respond to realistic operational problems such as communication failures, delays at pickup locations, changes in delivery locations or changing UTM constraints (e.g. introduction of new no-fly zones)?
   (e) How does the performance of the system vary with changing parameters (e.g. demand rate, environment size, fleet size, obstacle density etc)?
   (f) How do the structural assumptions of the model affect performance and how can they be validated?
   (g) Can the results of the model be used to provide insight on the circumstances in which UAV delivery is most advantageous compared to existing delivery modes?

4.3. Work breakdown

In order to answer the above research questions in a structured manner, the following work breakdown structure is proposed. Note that a high-level formulation of the UAV delivery coordination problem, including relevant types of agents and their characteristics, is already provided in Section 1.5.

4.3.1. Model of environment and operational setting

This section of the project is concerned with answering research question 1, and developing a model of the operational setting in which urban UAV deliveries will take place. Several of the subquestions are already answered, to a certain extent, by the analysis provided in Chapter 1. Key modeling efforts to be addressed at this stage include:

- Development of a two-dimensional model that approximates the layout of a modern city. A key choice to be made here is whether to follow the layout of a specific city, or attempt to build a model that is generalizable to several kinds of cities. Regardless, enough flexibility should be allowed to change the structure of the environment (e.g. location and amount of obstacles) in the model analysis stage. This is essential to understand the performance of the coordination framework.
- Placement of no-fly zones and other possible UTM-induced restrictions in the environment.
- Placement of the operator's charging hub(s). In the model analysis stage, it may be of interest to change the location or amount of charging hubs, so these parameters should be made as flexible as possible.
- Definition of relevant performance limitations of the UAVs. The analysis summarized in Table 1.1 already provides good starting choices, but these should be further investigated and validated.

4.3.2. Model of delivery demand case

This package is concerned with answering research question 2 by developing a realistic demand case for the urban delivery as a service operating model. While there are several ways to set up the demand case, the following general guidelines are proposed at this stage:

- Customer orders should be generated continuously throughout the problem and follow a sensible distributions (e.g. demand highs during typical dinner times for a food delivery case). In addition, the probability of a customer spawning in a specific area of the city could be made proportional to the local population density.
- Customer orders should include: a pickup location, a delivery location, and a desired delivery window that the operator will try to respect.

4.3.3. Coordination model development

This is the main work package and involves developing the coordination framework that constitutes the basis of this thesis, thereby answering most parts of research question 3. The framework should allow agents to perform online task allocation and path planning, such that delivery tasks received by the operator can be allocated in real-time and without causing plan conflicts. A few preliminary choices regarding the structure of the model follow directly from the findings of this literature study and can already be made. Firstly, the task allocation problem is considered the driving element of the coordination framework and the one to be investigated in most detail. This is because it is highly complex and several extensions to state of the art techniques are required in order to capture the dynamism and planning constraints associated to the UAV delivery problem. Path finding is considered of secondary importance, and this motivates the choice of approximating the airspace as two-dimensional and not performing a detailed modeling of UAV dynamics and performance constraints in the steps described in Section 4.3.1. We are, however, interested in capturing the *coordination* aspect of path finding and ensuring that the UAVs are able to share enough information to plan conflict-free 4D trajectories, since given the high density of UAVs it may be impractical to rely solely on in-flight (sense and avoid) conflict avoidance. This is why the techniques from the multi-agent path finding literature, which focus precisely on the conflict resolution aspect of path finding but operate in simple grid-like environments, were explored in Chapter 3, rather than approaches tailored to operate in three-dimensional continuous environments such as 3D-Field D* [25], or UAV-specific paradigms such as the highly effective quadratic programming approach proposed by Mellinger and Kumar [104] and later extended by Bry and Richter [21] [134].

Figure 4.2 shows an overview of the basic coordination model we envision to treat the allocation of a customer order in real-time. The representation draws upon the classes of agents described in Section 1.5. When a Customer executes an order for an item to be delivered, it informs the Operator of the desired delivery window and location. The Operator then coordinates with the Seller and collects information regarding the time at which the item will be available at the pickup location, and the task allocation procedure is initiated. Based on the analysis conducted in Chapter 2, we propose to use an auction mechanism for task allocation that builds on the state of the art sequential auction variants studied. Each UAV computes its valuation for the task on auction by reasoning about the marginal cost or profit that it would incur in adding the task to its local plan, and submits its bid to the auctioneer. The winning UAV is passed a token with the other agents' plans and the latest airspace constraints, and plans a path through its new schedule in a decoupled manner following the HCA* approach described in Section 3.6.1. The UAV's new path is logged in the token, which is then returned to the

system. Once a UAV has finished executing a task, it informs the Operator and moves on to the next task in its schedule. Note that the proposed architecture hinges on the assumption that the operator will be able to communicate with all UAVs throughout the problem. This is deemed achievable with emerging cellular (and/or satellite) technology as discussed in Section 1.4.2. It is also the expectation of NASA and the FAA that the operator should maintain a stable communication link with its entire fleet. In the work package of Section 4.3.4, we discuss the possibility of relaxing parts of this assumption and consider the situation in which communication is limited or unreliable.

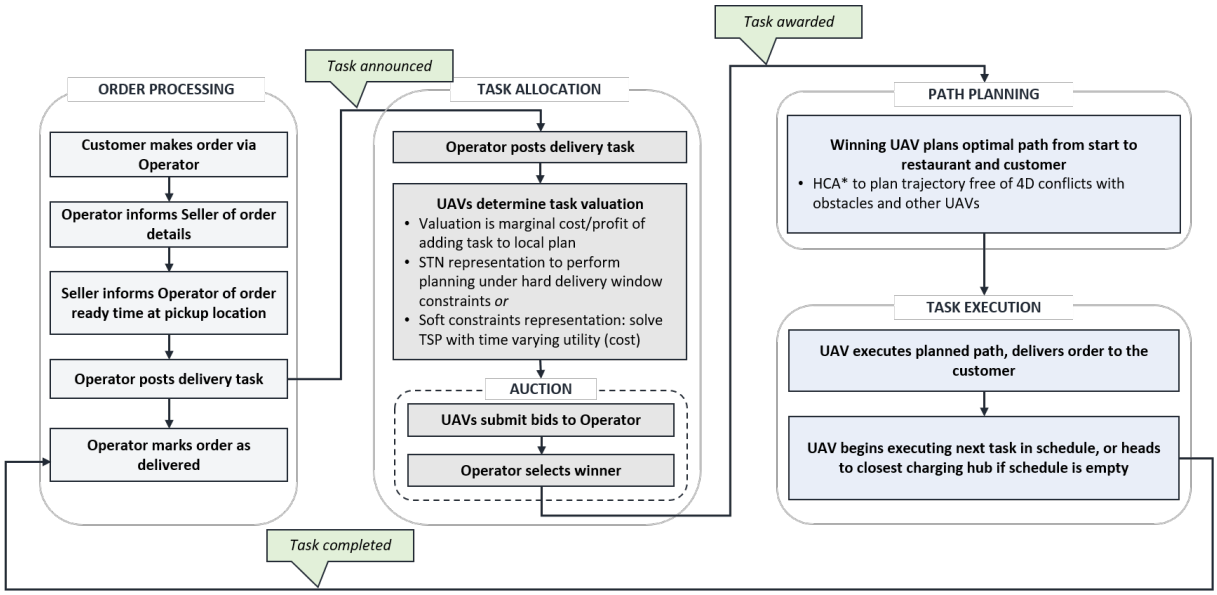


Figure 4.2: Preliminary model overview. Note that this represents a basic version of the model to be developed, which only considers the allocation of a single task. It is to be extended to account for task swaps and replanning procedures to better capture intertask synergies.

Within this work package, the basic version of the model depicted in Figure 4.2 should be enhanced by investigating the following additional functionalities:

- Support task swaps among agents via a global replanning phase. This can occur either in a peer-to-peer manner using a protocol such as the K-swaps mechanism described in Section 2.6.2 or via the auctioneer.
- Investigate different task clustering techniques to alleviate the computational burden in global replanning. Sequential single cluster (SSC) auctions, discussed in Section 2.8, consider the clustering of pickup and delivery tasks. However, the approach does not apply to time-constrained tasks as it does not consider a temporal dimension in the clustering. It is also simplistic and does not capture inter-task synergies. While the framework represents a good starting point, it should be substantially modified to capture the complexity of the problem addressed in this thesis.
- Investigate different planning and task valuation frameworks at the UAV level. Representations that have been used in the literature for temporally constrained tasks include simple temporal networks (see Section 2.7) or looser formulations that involve solving a TSP with time-varying utilities. In the latter case, an important choice to be made is whether to solve the problem exactly or heuristically.

4.3.4. Coordination model enhancement

While in principle fulfilling the steps of Section 4.3.3 would already constitute a substantial research contribution to both the multi-agent systems and the urban UAS communities, several opportunities for improvement have been identified in this literature study and are captured by some of the sub-questions in research question 3. **If time allows**, one or more of the following improvements can be investigated since they are deemed promising and have not been addressed in the literature:

- Allow UAVs to explicitly consider uncertainty in their local planning and bidding procedures. Within the UAV delivery setting, uncertainty can come both in the form of delays at the pickup point or in travel times, for instance due to airspace congestion or the introduction of new no-fly zones during task execution. While there is extensive work in the VRP literature on strategies to consider planning uncertainty [116] [115] [157], all are from a centralized perspective. In decentralized routing and task allocation, most approaches take a *reactive* stance and perform replanning once an agent encounters disturbances that

render the task execution no longer feasible [101]. Strategies to explicitly consider planning uncertainty in decentralized task allocation are relatively unexplored, so it would be interesting to investigate whether these can improve system performance in highly dynamic settings such as UAV delivery.

- Investigate the use of learning techniques to allow the agents to incorporate expectations of future demand into their task valuations, and whether this can be used to improve system performance. This would allow, for instance, a task that leads a UAV to a high-demand area and thereby offers high probability of a low-cost task in the future, to be valued at a premium. An interesting approach was proposed by Mes et al. for distributed routing problems, where schedules are assigned an *end value* which is representative of how desirable the location-time pair at the end of the schedule is to the agent [106]. This setup lends itself quite well to urban areas, in which features of neighborhoods such as population density are likely to correlate well to delivery demand and thereby to the desirability of a given location.
- Investigate the use of learning techniques to allow the agents to decide which tasks they should go through the computational effort of performing valuation (through hypothetical local planning) and bidding. Given the inherent structure in routing-based task allocation problems such as the one addressed in this thesis, it is likely that certain features of a task (e.g. time window or location) can be used to predict whether the UAV's bid will be sufficiently high to effectively win the task. If the UAV recognizes that it is highly likely not to be assigned the task, then it could decide not to participate in the auction at all, saving computation and communication cost.
- Allow the system to function under loss of cellular communication in the network. The principles of consensus-based auctions could be used to allow the UAVs to converge to conflict-free plans under communication via direct radio links with limited range, and thereby continue operating.
- Enhance the framework to handle multiple self-interested operators, each with its own UAV fleet. The challenge here is to develop a negotiation protocol that the competing operators can use to efficiently and fairly deconflict their operations, and incorporate it into the coordination framework. The most applicable protocol from the multi-agent negotiation literature is the alternating offers protocol, whose theoretical foundations are discussed in Appendix A.

4.3.5. Simulation and model analysis

Simulation experiments should be performed in order to validate the model, and measure and understand its performance, as identified in research question 4. The demand case developed in the work package from Section 4.3.2 will be refined and used for this analysis. The following are examples of parameters that could be varied:

- *Demand case*: UAV fleet size, demand rate and distribution throughout the operating window, distribution of population density.
- *Environment*: environment size, obstacle density and location, location and/or number of charging hubs.
- *Traffic rules*: simulations could be run under different traffic and routing rules which are being investigated under the UTM framework (discussed in Section 1.4.2). No-fly zones could also be varied dynamically throughout the problem to reflect changes in UTM constraints communicated in real-time by the ANSP (see Section 1.4.2). This would allow us to test how traffic restrictions that could realistically be imposed on delivery UAVs would affect the performance of the proposed coordination framework.
- *Operational issues*: to assess robustness, various issues that could occur in real life could be simulated. These include communication issues such as ground or V2V link failure, as well as the complete failure of the ground station or one or more of the UAVs.

A
The alternating offers protocol for multi-agent negotiation

The alternating offers protocol is a highly influential multi-agent negotiation protocol with a vast amount of applications. It has been studied extensively in areas as diverse as economics, computer science and psychology. In this appendix, we provide a general discussion of the theoretical foundations of this protocol, the reason for which is twofold. Firstly, the protocol is a natural choice to address the problem of resolving conflicts among multiple self-interested operators in the shared low-altitude airspace. It fits many of the desiderata provided by NASA and the FAA as discussed in Section 1.4.2, and in recent work the protocol has been applied precisely to strategic deconfliction within UTM [4]. Since the deconfliction problem is considered a possible extension to the project or highly relevant direction for future work, it is important to review the foundations of the state of the art technique to solve it. In addition, variants of the protocol could be used to complement the task allocation mechanism, for instance to allow distributed task swapping with limited information sharing. We begin by elaborating on the bilateral negotiation case in Appendix A.1, discussing both single-issue and multiple-issue negotiation. In Appendix A.2, we discuss how the alternating offers protocol can be extended to the multilateral case, which is no trivial task. In line with the majority of the literature, we adopt the perspective that negotiation constitutes a *strategic* activity and therefore that the negotiating agents act to maximize their own utility and assume that other agents will do the same. Our analysis of the alternating offers protocol is therefore grounded in game theory [102] [152].

A.1. Bilateral negotiation

Bilateral negotiation considers the scenario in which there are only two negotiating parties. Their preferences may be directly contradicting as in the familiar buyer-seller example in which the issue is the price of the good in question, or coupled in a more complex manner. The task allocation domain is an example of the latter case, since one agent may be better suited for a particular task than the other and the agents may therefore have different utility functions across the set of tasks to be allocated. We start by considering the simpler case in which agents negotiate over a *single issue*, and subsequently elaborate on techniques which can be extended to the *multiple issue* case.

A.1.1. Single issue negotiation

Negotiation over a single issue is the most intuitive formulation in the literature as it relates to several familiar real-life situations. Examples include, for instance, the negotiation between a buyer and a seller over the price of a secondhand car. Within the UAV delivery framework, this could instead be the negotiation between two operators over priority in the airspace or how to split the cost of replanning required to avoid a conflict. We begin by considering Rubenstein's formulation of the single issue negotiation problem, in which two agents a_1 and a_2 are bargaining over how to divide a "pie" (a resource) among each other [140]. For simplicity, we normalize the value of the entire pie to be 1 unit. The pie is assumed to be *continuously divisible*, meaning that any way in which the agents choose to split the pie is valid, provided that (1) each agent's share of the pie is between 0 and 1 and (2) the shares of the two agents sum to 1. The negotiation set is therefore $N : \{(x, 1-x) : 0 \leq x \leq 1\}$ where x is the share of the pie that goes to a_1.

The most widely used protocol for single-issue negotiation is the *alternating offers protocol*, first explored by Stahl [158] and later generalized by Rubinstein [140] [114]. It takes place over a series of rounds $0, 1, ..., M$. At round 0, agent a_1 begins by proposing a solution $X^0 \in N$, which agent a_2 can either choose to accept or reject. If the proposal is rejected, then a new round begins in which a_2 makes a counteroffer which in turn can be accepted or rejected by a_1. The process, which is summarized in Figure A.1 continues until the agents are in agreement. If the agents cannot reach an agreement, we call the outcome of the protocol the *conflict deal* C.

In order to analyze this protocol, we make the assumption that the agents seek to arrive at an agreement no matter what and therefore prefer any possible solution $X \in N$ to the conflict deal C [114] [49].

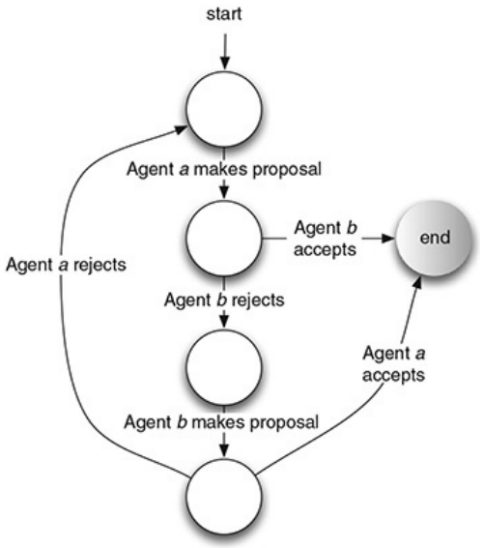


Figure A.1: Illustration of the alteranting offers protocol. Figure taken from [49].

Infinite-horizon alternating offers protocol

Let us begin by considering the *infinite-horizon* case, in which there is no time limit on the negotiation and therefore the amount of offer rounds is in principle unbounded. The strategy set of both agents is $S(t = n) = \{\text{Offer}(x^n \in N)\} \cup \{\text{Accept}, \text{Reject}\}$. That is, at each time step n the agent can either propose an offer in the negotiation set, or accept or reject a proposal. Suppose that agent a_1 plays the following strategy: *always propose* (1,0) *and reject any counteroffer that is different from* (1,0). Since agent a_2 prefers any deal to the conflict deal C, its best response to a_1's strategy is simply to accept the proposal even if it is the worst available deal to him aside from C. This situation constitutes a Nash equilibrium since both agents are best responding to each other's strategies. It is straightforward to see that in this setting, there will exist a Nash equilibrium pair of strategies for any given solution $X \in N$ and it will be reached in the first round [49]. Therefore, there exist an infinite number of Nash equilibria. To observe this, note that we could replace a_1's strategy with always offering any deal $X \in N$ and rejecting any different deal, and a_2's strategy would still be of accepting the offer.

Clearly, in the above scenario the concept of Nash equilibrium is not particularly useful because it allows the negotiation to end in an infinite number of ways. The reason is because we failed to account for the fact that bargaining itself has a cost. In practice, the agents' time has some value and therefore both agents will prefer any solution X to occur at round n rather than round $n+1$. By incorporating this degree of "impatience" in the agents' utility functions, we can arrive at a unique equilibrium solution. There are two fundamental modifications we can make to the agents' utility function in order to achieve this, namely assuming that the cost of bargaining is *fixed*, or that it compounds over time and is captured by an exponential *discount factor*.

Fixed bargaining cost: If we assume the cost of bargaining to agent a_i to be a constant $c_i > 0$, then we can write a_i's utility from a deal in which he gets share x at time t as $u_i(x,t) = x - c_i$. Given this utility function, Rubinstein proves the following [140]. Suppose a_1 moves first:

- If $c_1 < c_2$ there is a unique subgame perfect equilibrium at (1,0) in the first round[1].
- If $c_1 > c_2$ there is a unique subgame perfect equilibrium at $(c_2, 1-c_2)$ in the first round.
- If $c_1 = c_2 = c$ there is a subgame perfect equilibrium for any split in which both agents get at least c.

Discount factor: Now we assume instead that the cost of bargaining for the agent compounds over time. This model is more representative of the way in which humans value time[2] and is much more widely used than the fixed bargaining cost model. The utility function for a share x at time t is then $u_i(x,t) = x \cdot \delta_i^t$, where δ_i

[1] This is easily seen, because a_1 is more "patient" than a_2 (time is worth less to a_1 than to a_2) so he can credibly make the same threat as previously discussed of always offering (1,0) and rejecting any other proposal. Given this strategy, a_2 has no choice but to accept and is better off doing so immediately.

[2] In the vast majority of the literature tackling real-life human negotiation problems, utility takes the form of money. Human "impatience" is then captured by the fact that money today is worth more than the same quantity at a later date due to its potential earning capacity, which is exponential since money can earn compound interest. This phenomenon is best captured by the discount factor model.

is a constant discount factor between 0 and 1. The larger δ_i is, the more patient agent a_i is since time is less valuable to him. On the other hand, the smaller the discount factor, the larger the opportunity cost of failing to reach an agreement within a round, and therefore the more impatient the agent will be.

Let us attempt to derive an equilibrium for a general negotiation problem carried out via the alternating offers protocol with the above discount factor formulation, using the method outlined in [140] [147] [130]. Suppose agent a_1 is the first mover and that the game starts at time t. Let M be the maximum share that a_1 can possibly get at time t. Now let us proceed by backward induction and consider the time step $t-1$ in which a_2 proposes and a_1 responds. Given that M is the maximum a_1 will receive at time t, at time $t-1$ agent a_1 will accept any share greater than or equal to $\delta_1 M$. Agent a_2 is therefore certain to be able to propose $\delta_1 M$ and retain $1-\delta_1 M$. It follows that at time step $t-2$, a_2 will accept a proposal if it grants him at least $\delta_2(1-\delta_1 M)$. Therefore a_1 will never be able to secure a share greater than 1 - $\delta_2(1-\delta_1 M)$, implying that:

$$M = 1-\delta_2(1-\delta_1 M) \implies M = \frac{1-\delta_2}{1-\delta_1\delta_2} \tag{A.1}$$

It is straightforward to observe that if we repeat the same reasoning above to find the *minimum* value that a_1 could secure at time t, we arrive at the same expression as in Equation (A.1). Therefore the situation in which the negotiation is concluded in the first round with a_1 getting share M constitutes a unique equilibrium solution X_E where:

$$X_E = \left(\frac{1-\delta_2}{1-\delta_1\delta_2}, \frac{\delta_2(1-\delta_1)}{1-\delta_1\delta_2}\right) \tag{A.2}$$

Note that this equilibrium is clearly subgame perfect since we have arrived to it via backward induction. It is also Pareto optimal since the shares sum to 1, and it is reached instantly. These properties are highly desirable, but it must be noted that the agents' shares are highly dependent on who moves first.

Finite horizon alternating offers protocol

In practice, many negotiation problems will need to be concluded within a finite amount of rounds. We can enforce this by imposing a deadline d by which the agents need to have reached an agreement, otherwise the negotiation is deemed to have failed resulting in the conflict deal C. If we let the negotiation begin at time $t = 1$, then a_i's utility from receiving a share x at time t is:

$$u_i(x,t) = \begin{cases} x\cdot\delta_i^{t-1} & t \le d \\ 0 & t > d \end{cases} \tag{A.3}$$

In the case where $\delta_1 = \delta_2 = \delta$, it is relatively straightforward to show that there is an equilibrium and which will lead to an agreement in the first time step. Suppose once more that a_1 is the first mover, then a_1 will look ahead until the deadline d and reason by backward induction. If $d = 1$, then a_1 is in an absolutely dominant position and can offer (1,0) and a_2 needs to accept to avoid the conflict deal C. If $d = 2$, then a_2 can guarantee at least δ by rejecting the first offer and proposing to get the entire share δ at the second time step; a_1 will therefore offer $(1-\delta,\delta)$ and a_2 will accept. If $d = 3$, a_1 can guarantee δ^2 by rejecting until the third time step and proposing to get the entire remaining share δ^2. However, he can do better than this by observing that a_2 can do no better than offering $(\delta^2, \delta-\delta^2)$ in the second round because he knows a_1 can guarantee δ^2 via the aforementioned strategy. a_1 can therefore offer $(1-\delta+\delta^2, \delta-\delta^2)$ at the first time step and a_2 will accept. Extending this reasoning to an arbitrary deadline d, we obtain that the equilibrium X_E is:

$$X_E = \left(\sum_{k=0}^{d-1}(-1)^k\cdot\delta^k, 1-\sum_{k=0}^{d-1}(-1)^k\cdot\delta^k\right) \tag{A.4}$$

The above equilibrium is Pareto optimal, and dependent on the discount factor δ, the deadline d and which agent moves first[3]. Figure A.2 summarizes an insightful analysis conducted in [49] showing the effect of the deadline d and discount factor δ on the equilibrium solution. Much of the negotiating power comes from the share of offer rounds the agents have. We can view every odd round as an increase in the utility a_1 can generate and every even round as an increase in the utility a_2 can generate, which explains the (out of phase) oscillatory nature of the two agents' equilibrium shares. The magnitude of the oscillations decreases exponentially and converges to zero with increasing d. We can also observe that patience dilutes the advantage of the first mover. With increasing δ, the advantage of the first mover converges to zero. Note that for $\delta = 1$, we have the situation in which the last mover gets the entire pie because time no longer plays a role in the agents' utilities and the last mover's threat of continuously proposing (1,0) and rejecting any other offer is therefore credible.

[3] Note that the same reasoning can be applied to the case in which $\delta_1 \neq \delta_2$ and an equilibrium which will lead to an agreement in the first time step can still be found via backward induction, provided both agents know each others' discount factors. For a thorough analysis of this scenario see the original work of Rubinstein [140].

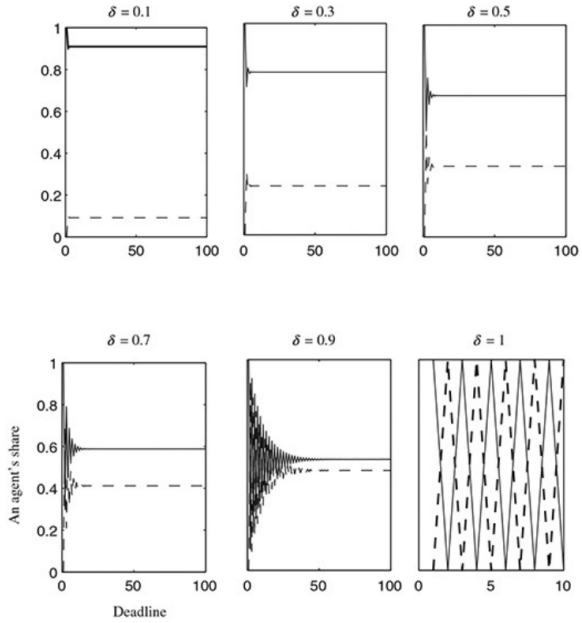


Figure A.2: Variation of the equilibrium outcome with the discount factor δ and deadline n. The continous line represents the first mover, the dotted line the opponent. Figure taken from [49].

Imperfect information

For now, we have analyzed the alternating offers protocol assuming *perfect information*, which has allowed us to derive tractable equilibria for both the infinite and finite horizon cases and reason about which characteristics of the negotiation problem affect such equilibria. In many practical applications, however, the agents will be missing some information about parameters of the negotiation problem, such as the other agent's discount factor. These situations should be modeled as games of *imperfect information*[64]. A practical way to approach this is to follow the method of Harsanyi [63] and assume that every agent follows the Bayesian approach to solve the imperfect information negotiation game[49]. Every agent generates a probability distribution over all parameters which are unknown to him or on which there is some degree of uncertainty. The agents will then play in such a way as to maximize their utility given such probability distribution. This means that they will play a mixed strategy profile with the pure strategies and associated weights matching point values in said probability distribution[4]. This may increase the amount of time required to reach an equilibrium agreement and lead to non Pareto optimal outcomes.

A.1.2. Multiple issue negotiation

We now extend our scope to include situations in which two agents need to negotiate over more than one issue. This would allow us to represent, for instance, the situation in which two UAV delivery operators need to negotiate over how to distribute the replanning costs due to multiple conflicts in the airspace or over how to divide planning priority among these conflicts. In order to analyze the problem of negotiation over multiple issues, we need to reason about the sequence in which the issues will be addressed. We call this the *negotiation procedure*. The procedure could, for instance, dictate that the issues be resolved one by one in sequence or all at once. It should be carefully designed because, along with the way in which the agents spread their priority among the issues, the procedure dictates the agent's strategic behavior [144] [170] and the quality of the resulting equilibria[49].

Let us denote the set of issues in contention as $I = \{1, 2, ..., m\}$ and normalize the magnitude of each issue to one as in the single-issue case, meaning that every issue constitutes a "pie" of unit size. We will focus on the situation in which the negotiation must occur within a deadline d and the cost of bargaining is modeled via the discount factor representation, which as discussed in Appendix A.1.1 is the most realistic approach. The space of possible outcomes is no longer one dimensional as in the single-issue case, but rather the cross product of the negotiation sets of all individual issues. A deal X over all the issues is therefore an m-tuple $X=\langle X_1,...,X_m\rangle$ of single-issue deals. The agents may need to reason about the *cumulative* utility that arises from all the single-issue utilities contained in X, weighted by some personal evaluation of the importance of said issues. Each

[4]Formally, the aforementioned probability distribution represents the agents' belief system.

agent therefore has a cumulative utility function $U(X,t)$ that maps a time t and deal X to a single value of utility.

There are three main negotiation procedures for multiple issue negotiation, namely the *package-deal* procedure, the *simultaneous* procedure, and the *sequential* procedure [121][56]. We will discuss each in the context of perfect and imperfect information.

Package-deal procedure

The package-deal procedure (PDP) is essentially an extension of the alternating offers protocol discussed in Appendix A.1.1 to the multiple-issue case. The rules are similar, in that one agent makes an offer and the other can choose to accept it, or reject it and make a counteroffer at the next time step. However, an offer must include a proposed split for every issue in contention, and is therefore of the form $\langle (x_1,y_1),...,(x_m,y_m)\rangle$ where x_i is the share of issue i to agent a_1 and y_i is the share of issue i to agent a_2. The responding agent can only accept or reject the *entire* offer and is not allowed to respond only to a subset of issues. Agent a_i's cumulative utility function takes the following form:

$$U_1(X,t) = \sum_{j=1}^{m} u_{i,j}(X,t) \tag{A.5}$$

Where $u_{i,j}$ is a_i's utility for issue j alone. This single-issue utility function is similar to that discussed in Appendix A.1.1, but includes a weight $w_{i,j}$ for issue j. For a_1, this is:

$$u_{1,j}(x,t) = \begin{cases} w_{1,j}\cdot x_j \cdot \delta_i^{t-1} & t \le d \\ 0 & t > d \end{cases} \tag{A.6}$$

For both agents, the key here is to make an effective tradeoff between the shares gained in each of the issues, in such a way as to maximize cumulative utility [81]. Let us analyze such a tradeoff. Suppose that at time t it is a_1's turn to offer. Then a_1 should make a proposal X such that a_2 gets its discounted utility for $t+1$. If the proposed share were smaller, a_2 would not accept. For a single issue, there existed a single offer satisfying this condition and it was relatively straightforward to determine it via backward induction as we did previously. In the multiple-issue case, however, there are multiple suitable offers and the offering agent needs to determine the one that maximizes its own utility. This results in the following optimization problem:

For agent a_1:

$$\begin{aligned} \max \quad & \sum_{k=1}^{m} w_{1,k}x_k \\ \text{s.t.} \quad & \delta^{t-1}\sum_{k=1}^{m}(1-x_k)\,w_{2,k} = \mathrm{UB}(t+1) \\ & 0 \le x_k \le 1 \quad \text{for} \quad 1 \le k \le m \end{aligned} \tag{A.7}$$

For agent a_2:

$$\begin{aligned} \max \quad & \sum_{k=1}^{m} w_{2,k}x_k \\ \text{s.t.} \quad & \delta^{t-1}\sum_{k=1}^{m}(1-x_k)\,w_{1,k} = \mathrm{UA}(t+1) \\ & 0 \le x_k \le 1 \quad \text{for} \quad 1 \le k \le m \end{aligned} \tag{A.8}$$

The above optimization problem is a fractional knapsack problem and can be solved in polynomial time with a simple greedy approach[35]. Let us explain a suitable greedy approach from the perspective of a_1. Compared to a traditional fractional knapsack problem, the difference is that we effectively start with a full knapsack, which is the deal $\langle \mathbf{1},\mathbf{0}\rangle = \langle (1,0),...,(1,0)\rangle$ in which a_1 gets the entire share of all issues. Then, a_1 needs to gradually empty the knapsack by conceding utility to a_2 across the issues until a_2 is granted $\mathrm{U2}(t+1)$. To do this, a_1 should take as much as possible in the issue k with the highest $\frac{w_{1,k}}{w_{2,k}}$, then move on to the issue with the second highest weight ratio and so forth. This is because the higher $\frac{w_{1,k}}{w_{2,k}}$, the more utility can be granted to a_2 per unit loss of a_1's utility.

We can proceed once more via backward induction, looking ahead to the deadline $t=d$ and then reasoning backwards, to determine the agents' equilibrium strategies. If negotiation reaches $t=d$, then the offering agent has all the power and can propose to get the entire share of all the issues in contention and the responding agent is forced to accept to avoid the conflict deal. For any round other than the deadline, the offering agent should propose the deal that maximizes its own utility while ensuring that the responding agent's utility is equal to the cumulative utility it would get by rejecting and proposing its optimal offer in the next time step. If such condition is satisfied, the receiving agent should accept the offer. This is the solution to the optimization problem in Equation (A.7) and Equation (A.8). It is easy to see that, when both agents play the above strategy at $t=1$, the equilibrium agreement is instant. The outcome is also Pareto optimal, since all deals that constitute a solution to Equation (A.7) and Equation (A.8) sum to $\delta^{(t-1)}$ over all issues, and the agreement is instant meaning that the cost of bargaining is null and the equilibrium outcome sums to 1 over all issues.

Simultaneous procedure

In the simultaneous procedure, negotiations over separate issues are considered as independent. There are effectively m single-issue negotiations running in parallel as separate problems. In each single-issue negotiation problem, the agents use the alternating offers protocol. From the discussion in Appendix A.1.1, we know that each single-issue problem has a unique equilibrium which occurs at the first time step and that can be derived via backward induction for both the infinite and finite horizon cases. Since all negotiations are conducted in parallel, we will reach a unique agreement over all m issues in the first time step. As we have seen in Appendix A.1.1, the first mover has an advantage which varies based on characteristics such as the discount factor and the negotiation deadline. In this procedure, the agents need to decide who will move first in each of the m single-issue negotiation games. While this could be the same agent across all games, the first mover is usually varied or alternated across the issues to dilute the first mover advantage.

This procedure is highly time-efficient, tractable and simple to implement. However, the outcome will not necessarily be Pareto optimal. As we have seen in the PDP case, there are multiple deals which the receiving agent will accept, but only one which will maximize the offering agent's utility. In order to find such deal, the offering agents needs to perform a trade-off across the issues and decide which issues to concede most on and which to be more aggressive on (see Equation (A.7)). By forcing the agents to conduct all negotiations simultaneously, we do not allow them to conduct such tradeoffs and therefore generate an inefficiency which compromises the Pareto optimality of the outcome.

Sequential procedure

The sequential procedure is divided into m stages, one for each issue, in which the issues are negotiated one by one. Every stage is split into discrete time periods and is assigned a deadline $\bar{d}$. If an agreement is reached for an issue at some time period before the deadline, then the stage for that issue terminates and the stage for the next issue begins. There are two different ways of implementing this procedure, namely the *independent* and *simultaneous* implementations [49].

Independent implementation: Here, we consider the case in which the agreement over an issue is effective immediately, within the stage in which it is being negotiated. Each stage is therefore analogous to the finite horizon single-issue negotiation problem. From Appendix A.1.1, we know that the offering agent's equilibrium share of the issue is $\sum_{k=0}^{n-1}\left((-1)^k \cdot \delta^k\right)$. We also know that the equilibrium agreement is unique and is reached at the first time step. Therefore, if we treat the issues in sequence it will take exactly m time steps to reach an agreement over all issues. However, the outcome may not be Pareto optimal because we are again not allowing the agents to perform trade offs across the issues.

Simultaneous implementation: The difference here is that, while we still treat the issues sequentially in stages, we only allow an agreement to be effective after *all* issues have been agreed upon. If the agents fail to reach an agreement on any issue, then all agreements on previous issues will be dropped and the entire negotiation for all issues will fail. Fatima et al. show that also in this case there exists a unique equilibrium that will be reached in m time steps and can be found via backward induction [49]. Compared to the independent implementation case, the equilibrium outcome is distributed differently across the issues because all issues are discounted until the final agreement is made at time step m. The equilibrium is, once more, not Pareto optimal for the reasons discussed previously.

Imperfect information

There are several possible sources of uncertainty in multiple-issue negotiation, namely the opponent's discount factor, utility function and the negotiation deadline. A straightforward way to deal with this uncertainty is, like in the single-issue case, to apply standard expected utility theory [57].

Let us illustrate this concept first to the PDP by considering a scenario with symmetric uncertainty about the discount factor δ. Both agents have the same discount factor, but they do not know its value with certainty. The discount factor δ can take a finite number of values $\delta_1, ..., \delta_n$. The agents build a probability distribution over all possible values such that $P_i = P(\delta = \delta_i)$. The agent's expected utility is then $E[U_a](x,t) = \sum P_i \cdot \delta_i^{t-1} \cdot u_{i,j}(x, t=0)$

The problem is now the same as in the perfect information case, except that instead of fixed utilities we consider *expected* utilities. Suppose a_1 is the offering agent at time t. Agent a_2 will propose a deal that maximizes its own expected cumulative utility while providing a_2 with at least its expected cumulative utility for the next time step. This is the solution to a fractional knapsack problem virtually identical to Equation (A.7), but with expected (probability weighted) utilities instead of fixed utilities [49]. It can be solved using the same greedy

approach discussed for the perfect information case and the same strategy profile[5] will lead to a unique subgame perfect and Pareto optimal equilibrium. Fatima et al. show that these same equilibrium properties also hold in games with symmetric and asymmetric uncertainty about the agents' utility functions [51], and with symmetric uncertainty about the deadlines [50].

We can apply the expected utility logic to the simultaneous and sequential procedures as well. In these two settings, the procedure to derive equilibria with imperfect information is virtually identical to that applied in the perfect information case. The only difference is that the utilities for the single-issue negotiation problems are expected utilities given the agents' (joint) probability distribution over the uncertain parameter(s). Fatima et al. prove that in games with symmetric and asymmetric uncertainty about the utility functions [51], and symmetric uncertainty about the deadline [50] or discount factor [49], both for the simultaneous and sequential procedure[6] there exists a unique equilibrium. Clearly, the equilibrium does not guarantee Pareto optimality since for neither of the two procedures was this the case in the perfect information setting. Little fundamental research has been conducted regarding games with asymmetric uncertainty on the deadlines and discount factors. To the best of our knowledge, there exists no formal proof regarding the behavior of the PDP, simultaneous or sequential procedures in these types of games.

A.2. Multilateral negotiation

We have until now discussed protocols and procedures for negotiation between two agents only. However, within the UAV delivery coordination problem we will come across instances in which more than two agents need to negotiate. This will happen, for instance, if more than two operators are involved in a conflict and need to negotiate to resolve it. In this section, we discuss extensions of the alternating offers protocol that apply to the multilateral case.

A.2.1. Extensions of the alternating offers protocol

Extending the alternating offers protocol to the multilateral case is not straightforward. Consider the situation in which there are three agents and each agent's approval is required for every offer that is made. Suppose a_1 begins by making an offer, and a_2 and a_3 must approve or reject. If both approve, an agreement is reached and the game ends. If a_2 or a_3 reject, then a new round begins in which a_2 offers and a_1 and a_3 respond, and this process continues until an agreement is made. Every player here has, at every time step, the power to veto any offer. This means that for any deal to be final it needs to be agreed upon *unanimously* by all agents. Osborne and Rubinstein [114] study this game extensively and show that it leads to multiple equilibria. In fact, they show that for sufficiently patient players, any deal in the negotiation set is backed by a subgame perfect equilibrium.

Several attempts have been made in the literature to alter the alternating offers protocol with rules that lead to a unique and tractable equilibrium. Chae and Yang propose a protocol in which agents perform a sequence of bilateral negotiations [27]. Let us return to our three-player example and suppose a_1 is the first to offer. Then a_1 chooses a specific player, say a_2, to make an offer to. Say a_2 accepts, then he exits the negotiation and can no longer offer in the same round. a_1 then makes an offer to a_3. If a_3 accepts, an agreement has been reached. If a_3 rejects, a new round begins in which a_3 offers (takes the role of a_1 here). Several similar sequential bilateral protocols have been explored in the literature [80] [26], and each of them is shown to result in a unique subgame perfect equilibrium.

Other approaches obtain desirable equilibrium properties by imposing negotiation deadlines [47]. In particular, Wu et al. [184] provide a protocol which is highly general and can produce Pareto optimal outcomes also in the presence of multiple issues. For the general case with m issues and n players, the protocol is as follows. In each round, every agent proposes a deal over all m issues, but only including the share of the resource he wants for *himself*. The next agent can then read the offers that have already been made and either accept them and make an offer for his own desired share, or reject the entire sequence of offers and refrain from making an offer in the current round. If any agent rejects the offers then the round ends and a new one begins, possibly with a different ordering. If, instead, in any given round, all agents make an offer, then an agreement is reached. If the negotiation exceeds the deadline, the result is the conflict deal C. The agents bid according to an order that must be specified before the negotiation begins and cannot be changed, but can be different for each round. The authors make the assumption that all agents are *benevolent*, meaning that given two outcomes to which they are indifferent but to which the other agents are not, they will choose the outcome that to the best of their knowledge maximizes social welfare (and therefore is best for the other agents). Under the benevolence

[5] Again with expected instead of fixed utilities.

[6] Meaning the sequential procedure with both the simultaneous and independent implementation mechanisms.

assumption and in a complete information setting, there exists a strategy profile for all agents that constitutes a unique subgame perfect equilibrium that leads to a Pareto optimal agreement in the first round, if such an agreement exists. It is straightforward to see why this is the case. Similarly to the PDP, the agents here perform a tradeoff over all issues and identify the shares of all issues they should offer to keep for themselves in order maximize their own utility and provide the other players with the discounted utility they would receive in the next round by performing the same tradeoff. If there exist multiple offers that satisfy this criterion, due to the benevolence assumption the offering agent will choose the one that maximizes social welfare. Since all agents prefer an agreement to occur earlier rather than later and they are benevolent, this agreement will occur in the first round and no value is wasted due to the cost of bargaining. The outcome is therefore Pareto optimal.

III

Supporting work

A
Sensitivity and robustness analysis

In the scientific paper, we focus on analyzing the sensitivity of the system to the parameters that are most interesting in the context of the algorithms put forward in our work. We vary, for instance, the risk threshold and the weight of the risk in the bidding rule. This is to understand to what extent the system can leverage execution risk and test how much weight it should be given in the bids in order for pTeSSI to distributed it effectively among the agents. We also vary the re-auction threshold to gain insight into how different levels of re-auctioning affect the performance of pTeSSI, and do the same for the distance threshold in our bundling mechanism for pTeSSB. These analyses are all fundamental as they form the basis of the main conclusions of the study.

In this chapter, we perform additional analysis by varying two structural parameters which were kept fixed across the simulations, namely:

- the width of the delivery windows of the customer orders
- the size of the UAV fleet

A.1. Overview

We seek to understand whether large variations in the above parameters could affect the main conclusions of this work. We vary each parameter in both the positive and negative direction, and assess the *sensitivity* of the total deliveries executed by the system and the average auction duration under both the pTeSSI and TeSSI auction mechanisms. We then test whether main conclusions of our work are *robust* to these variations by determining whether they are still statistically significant, and compare the effect sizes to the nominal case described in the scientific paper. We focus on the most important hypotheses related to the effectiveness of the approaches put forward in our work, and follow the same naming used in the scientific paper. The hypotheses we test for robustness are the following:

- $\mathbf{H_{A1}}$: pTeSSI leads to a larger number of successful deliveries than TeSSI.
- $\mathbf{H_{B1}}$: pTeSSI-re leads to a larger number of total deliveries than pTeSSI.
- Performing bundling in combination with periodic re-auctioning is effective in retaining the performance advantage while mitigating the associated increase in auction duration. We split this claim into two testable components:
  - $\mathbf{H_{C1}}$: pTeSSB with periodic re-auctioning outperforms pTeSSI in terms of the total number of deliveries executed by the system.
  - $\mathbf{H_{C2}}$: pTeSSB with periodic re-auctioning leads to a lower average auction duration than pTeSSI with periodic re-auctioning.

  We consider the bundling mechanism effective if and only if both $\mathbf{H_{C1}}$ and $\mathbf{H_{C2}}$ hold. For compactness, we denote the hypothesis that both parts hold true and therefore bundling is effective as $\mathbf{H_C}$.

We run each simulation 40 times, which was found to be enough to stabilize the coefficient of variation for all cases. For each claim, statistical significance is established via the Wilcoxon signed-rank test and the effect sizes are measured via the Vargha-Delaney A value. Note that while testing each claim, we always compare different pairs of simulation result sets. There is therefore no need to control for the family-wise error rate via a post hoc correction such as the Bonferroni one. Note that we do not test $\mathbf{H_{B1}}$ and $\mathbf{H_C}$ for the lowest demand case investigated. This is because, as discussed in the scientific paper, we had already noted that the agents were able to fulfill virtually all deliveries with pTeSSI, so there is no reason to attempt to increase the number of deliveries by dynamically re-auctioning tasks.

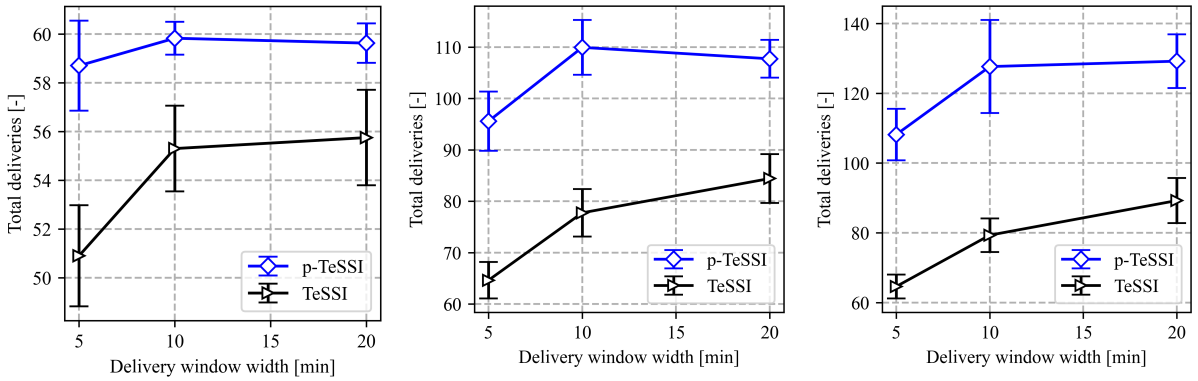
Following from the analysis conducted in the paper, we use $R = 0.5$ and $\rho = 10$ across all runs, as these values were found to maximize the performance of pTeSSI. We also run the analysis for the `MAX-T` bidding rule only, since it was found to be substantially more efficient than `SUM-DIST` and we did not observe statistically significant differences in the results across the two bidding rules.

A.2. Delivery window width

In the experiments presented in the paper, we use a delivery window width of 10 minutes. That is, the earliest and latest delivery time of a customer order are always 10 minutes apart. Here, we perform additional simulations with delivery window widths of 5 and 20 minutes.

A.2.1. Sensitivity

Figure A.1 the total number of deliveries executed by the system under pTeSSI and TeSSI for the three demand levels investigated in the paper, plotted against the delivery window width. We note that the performance of the system drops substantially when decreasing the window width from 10 to 5 minutes, for both mechanisms. This is expected as the time constraints become more difficult for the agents to comply with, thereby making the problem more difficult to solve. When increasing the width to 20 minutes, however, we do not observe a significant improvement in performance for pTeSSI in any of the demand cases.



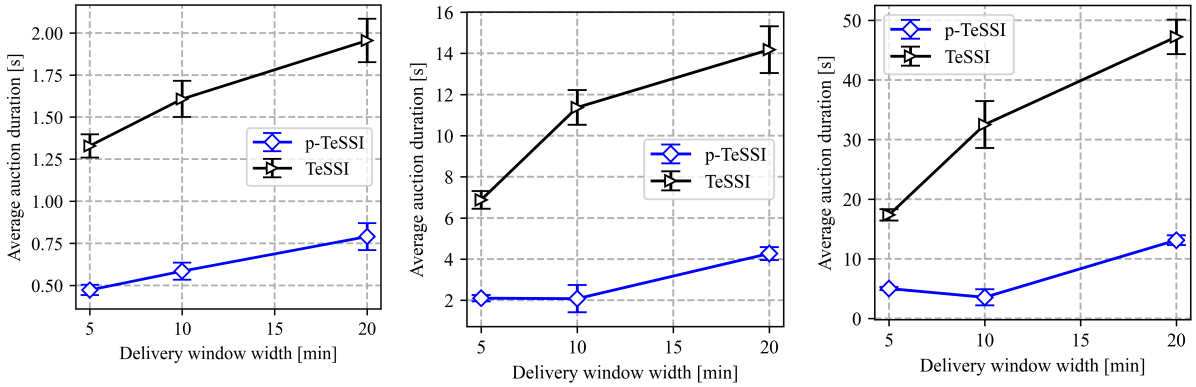
(a) 60 orders **(b)** 120 orders **(c)** 180 orders

Figure A.1: Total number of deliveries executed by the system under pTeSSI and TeSSI with varying delivery window widths for the three demand levels investigated in the scientific paper.

Figure A.2 shows the average auction duration plotted against the delivery window width. Across both mechanisms, the auction duration tends to increase substantially with the delivery window width. We observe up to a two-fold increase in the auction duration for pTeSSI and a three-fold increase for TeSSI when comparing the 5 minutes to the 20 minutes case. This is because larger windows tend to increase the computational overhead in the bidding phase outlined in Algorithm 1 of the scientific paper. There are two reasons for this, which are best explained by examining the bidding phase in more detail. Consider an agent who needs to bid on a task T on auction. According to algorithm 1, it will attempt to insert T in all positions of its schedule, and compute a bid for each *valid* position. Recall that a position is valid unless T's earliest delivery time exceeds the latest delivery time of the task in the schedule that would follow it, in which case we can immediately conclude that it will lead to an inconsistent schedule. The larger the delivery window of the tasks, the more likely that any given insertion point in the agent's schedule will be valid since the above condition for the invalidity of an insertion point will be less likely to hold. In addition, larger delivery windows make the constraints easier to satisfy, meaning that the agents can accommodate more tasks in their schedule. This further increases the computational overhead of the auction, since the runtime of the algorithms run on the temporal networks in the bidding phase of both TeSSI and pTeSSI grows with the size of the network.

A.2.2. Robustness

Table A.1 shows the p-value and effect size (A) for the three claims discussed in the beginning of this chapter. We observe that $\mathbf{H_{A1}}$ is statistically significant with very large effect sizes for all tested values of the delivery window width and across all demand levels. This indicates that the performance difference between pTeSSI and TeSSI is robust to large changes in this parameter. The same holds for $\mathbf{H_{B1}}$, but the effect sizes appear to increase with the width of the delivery window, indicating that the performance advantage obtained via re-auctioning becomes stronger. The only instance in which we cannot accept $\mathbf{H_C}$ is with the mid demand level and a delivery window width of 5 minutes. In this case, bundling cannot decrease the auction duration ($\mathbf{H_{C2}}$ has $p > 0.05$) while maintaining a significant performance improvement over the case with no re-auctioning. Upon closer inspection, this is because the performance increase due to re-auctioning is already not large (0.66 effect size), leaving little room for bundling to trade-off a portion of the performance increase for faster runtimes. In



(a) 60 orders **(b)** 120 orders **(c)** 180 orders

Figure A.2: Average duration of the pTeSSI and TeSSI auction with varying delivery window widths for the three demand levels investigated in the scientific paper.

addition, the auction is at its fastest in the case with the narrowest delivery windows (as discussed in section Appendix A.2.1), so there is relatively less to gain.

Demand case	Delivery window width		H_{A1}	H_{B1}	H_C	
					H_{C1}	H_{C2}
60 orders	5 min	p	3.32e-8	-	-	-
		A	0.99	-	-	-
	10 min (Nominal)	p	3.1e-8	-	-	-
		A	0.99	-	-	-
	20 min	p	4.59e-8	-	-	-
		A	0.98	-	-	-
120 orders	5 min	p	3.48e-8	7.07e-4	3.92e-2	0.11
		A	1.0	0.66	0.62	0.62
	10 min (Nominal)	p	3.5e-8	5.87e-7	2.60e-4	3.10e-6
		A	1.0	0.83	0.73	0.81
	20 min	p	3.44e-8	3.42e-8	7.62e-7	6.33e-3
		A	1.0	1.0	1.0	0.68
180 orders	5 min	p	3.53e-8	2.22e-6	1.76e-3	3.47e-5
		A	1.0	0.84	0.71	0.81
	10 min (Nominal)	p	3.5e-8	2.82e-6	2.42e-2	4.16e-8
		A	1.0	0.86	0.62	0.97
	20 min	p	3.54e-8	3.55e-8	2.51e-6	7.1e-3
		A	1.0	1.0	1.0	0.73

Table A.1: Analysis of robustness of three main hypotheses made in this work to large variations in the width of the task delivery windows.

A.3. Fleet size

In the paper, we had fixed the fleet size to 10 UAVs. Here, we run simulations with additional fleet sizes of 4 and 20 UAVs. In doing so, we hold the demand *per UAV* constant in order to assess system performance in a manner that is agnostic to the size of the UAV fleet. The demand levels corresponding to the different fleet sizes are outlined in Table A.2.

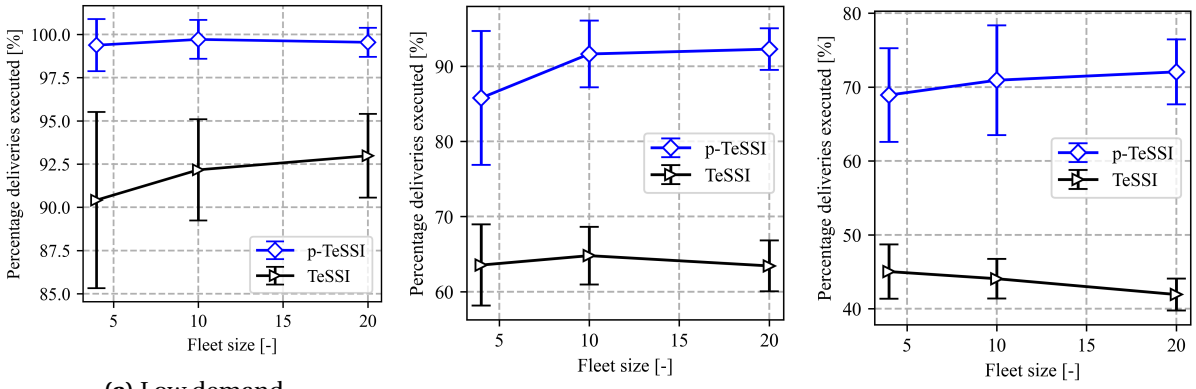
A.3.1. Sensitivity

Figure A.3 shows the deliveries executed by pTeSSI and TeSSI plotted against the UAV fleet size for the three demand levels. Rather than the total amount of deliveries, we show the *percentage* of the customer orders that the fleet was able to satisfy, which constitutes a fair comparison since the demand *per UAV* is fixed across the different fleet sizes. We observe only a slight variation with the size of the UAV fleet for both methods, with

		Total orders		
Demand level	**Orders per UAV**	4 UAVs	10 UAVs	20 UAVs
Low	**6**	24	60	120
Medium	**12**	48	120	240
High	**18**	62	180	360

Table A.2: Outline of the demand levels used for the different fleet sizes. The number of orders per UAV is kept constant in order to draw conclusions among the different fleet sizes.

the largest difference being ≈ 5% for pTeSSI and ≈ 3% for TeSSI. Across all demand cases, the performance of pTeSSI is at its worst for the smallest fleet size, although this also leads to the largest variability in the simulation data. Overall, the same demand per UAV leads to a similar percentage of the demand case being satisfied, and across all fleet sizes pTeSSI still clearly outperforms TeSSI.



(a) Low demand **(b)** Medium demand **(c)** High demand

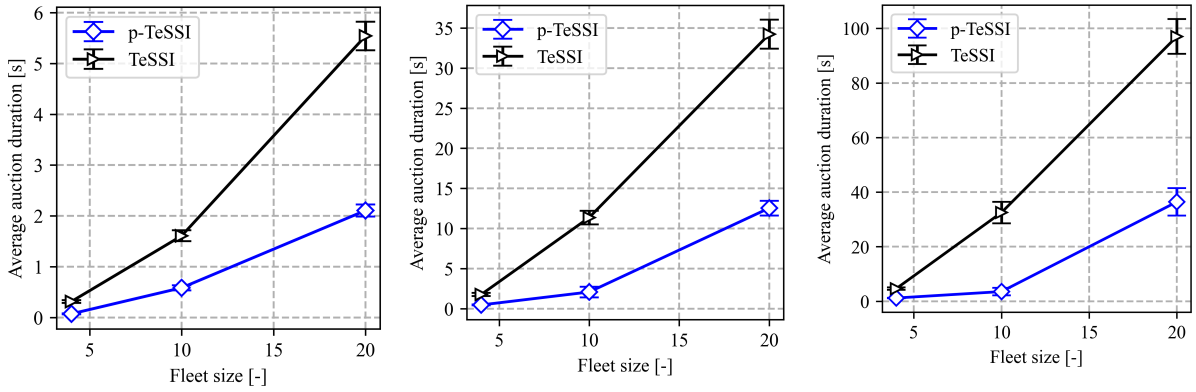
Figure A.3: Percentage of the customer orders successfully delivered by the system under pTeSSI and TeSSI with varying fleet sizes. Results are shown for each of the three demand levels outlined in Table A.2.

Figure A.3 shows the average auction duration plotted against the fleet size for both pTeSSI and TeSSI. The auction duration is found to increase significantly with the fleet size. The rate of change grows with the demand level, and is consistently higher when moving from 10 to 20 UAVs than from 5 to 10 UAVs, especially with pTeSSI. With pTeSSI, the auction duration increases at most by ≈ 2 times and ≈ 8 times when moving from a fleet size of 5 to 10 and 10 to 20 respectively. In section II of the paper, we show that pTeSSI is $\approx O(\frac{m^6}{n^4})$ when the tasks are approximately evenly distributed among the fleet, where m is the number of tasks on auction and n is the number of UAVs. TeSSI also has the same worst-case complexity as shown in [112]. In each demand case, when we increase the fleet size, the number of tasks on auction grows proportionally since we keep the number of orders *per UAV* constant (see Table A.2). Therefore, $O(m) = O(n)$, implying that the overall complexity of the auction is $\approx O(m^2)$. The increase in auction duration observed in Figure A.4 is therefore explained by the analysis conducted in section II of the scientific paper. Note that the objective of the sensitivity analysis shown in Figure A.4 is *not* to empirically assess the complexity of the auction algorithm, which would require expanding the range of fleet sizes to properly identify and quantify the asymptotic behavior of the auction runtime. We already analyzed the complexity of the auction algorithm from a theoretical standpoint in section II of the paper.

A.3.2. Thresholds for re-auctioning and bundling

With the fleet size parameter, we perform additional analysis by repeating more of the experiments conducted in Section V.B and V.C of the paper. We assess how the system performs under different levels of the reauction threshold and the distance threshold when applying bundling. This is because when varying the fleet size, we also increase the number of orders and thereby the number of tasks on auction. We are interested in assessing whether this affects the optimal level of re-auctioning and the nature of the trade-off between the auction duration and the total deliveries encountered when applying pTeSSB.

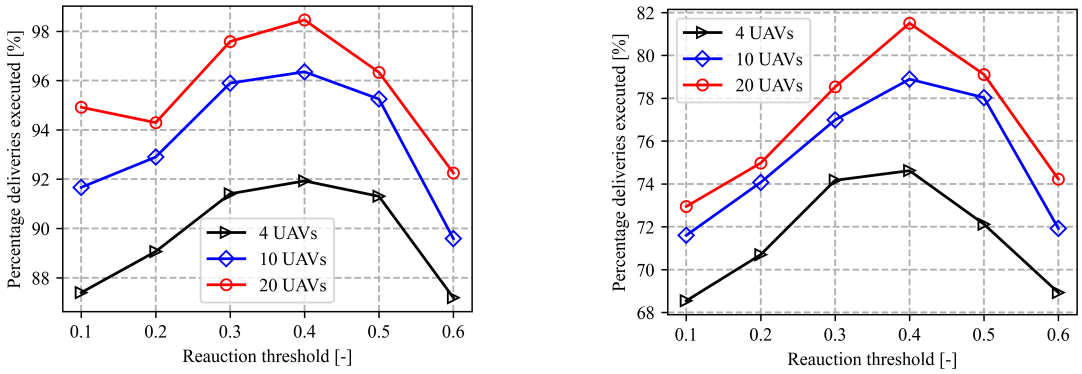
In Section V.B of the paper we observe that excessive re-auctioning led to a drop in performance, due to pTeSSI



(a) Low demand **(b)** Med (120 orders) **(c)** High (180 orders)

Figure A.4: Average duration of the pTeSSI and TeSSI auctions with varying fleet sizes. Results are shown for each of the three demand levels outlined in Table A.2.

not being able to capture the additional synergies but rather making decisions worse than the existing partial allocations in the early rounds of the auction, and conclude that a reauction threshold of 0.4 led to the best performance across both demand cases for which re-auctioning was explored. Figure A.5 shows the percentage of deliveries executed by the system for the different fleet sizes in the medium and high demand cases. The trend is remarkably similar across all fleet sizes, with the best performance consistently exhibited at a reauction threshold of 0.4, as observed in the paper. We therefore conclude that the trend is not sensitive to the fleet size. This suggests that it is truly the difference in the system objective that the re-auctioned tasks generate, and not the *number* of tasks that are re-auctioned, that dictates to what extent the re-auctioning routine will be successful and generate a better partial allocation. This further motivates our stance of attempting to select only those tasks which cause large enough changes in the system objective, rather than all tasks in the schedule as done in earlier work on online auction-based task allocation [69].

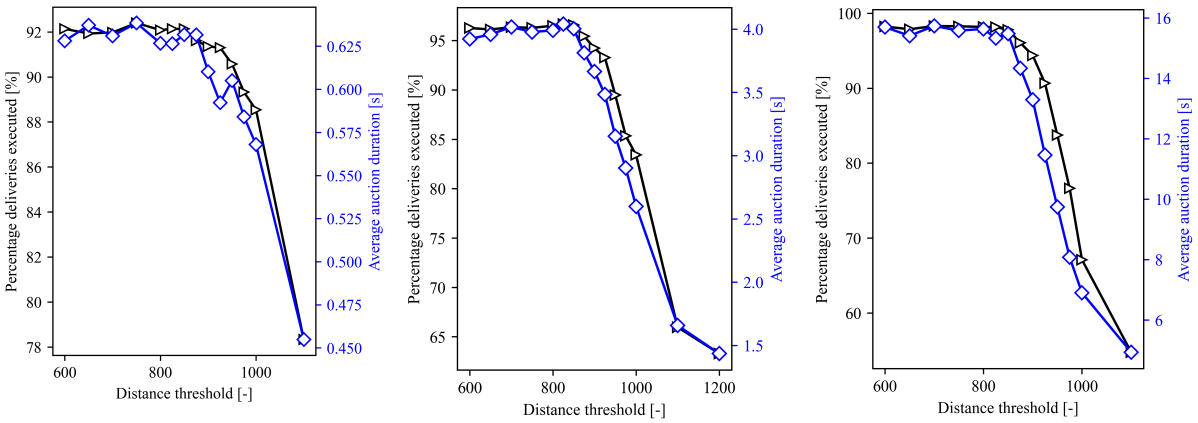


(a) Medium demand **(b)** High demand

Figure A.5: Effect of the reauction threshold on the total deliveries executed by the system with different UAV fleet sizes for the (a) medium and (b) high demand levels outlined in Table A.2.

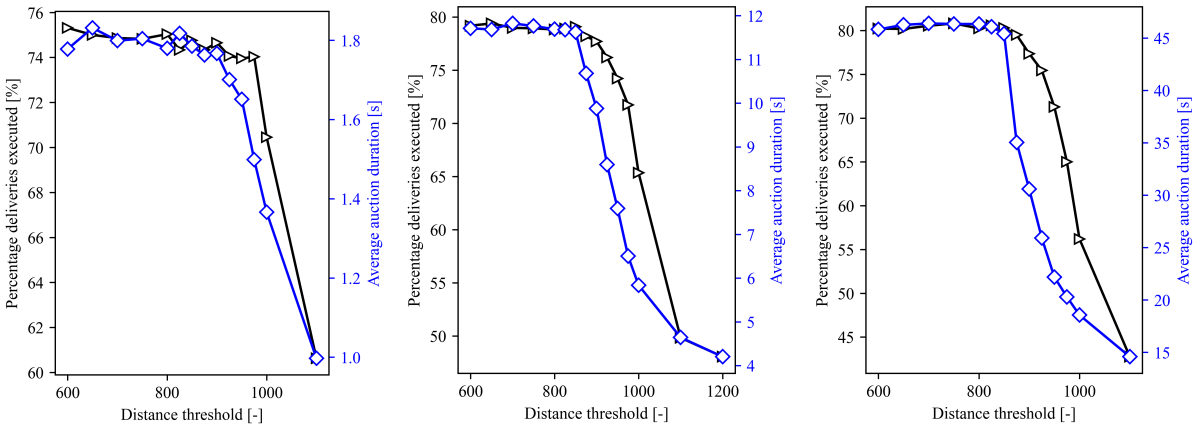
Figure A.6 and Figure A.7 show the percentage of deliveries executed and the average auction duration when combining periodic re-auctioning and bundling, plotted against the distance threshold for the three fleet sizes in the medium and high demand case respectively. Across all fleet sizes, we observe the same trend discussed in the paper. Increasing the distance threshold (and thereby generating larger bundles) reduces the auction duration but also reduces the total number of deliveries the system can execute. As already noted in the paper, the trade-off is significantly more favorable for the higher demand case. What we also observe here, however, is that bundling becomes more effective when we increase the fleet size. This is especially evident in the higher demand case (Figure A.7). When we increase the fleet size, the auction duration drops more steeply relative to the number of deliveries, as can easily be noted by observing the growth in the area between the two curves grows. These two insights provide further confirmation of the fact that bundling is more effective when the *task*

set on auction is larger, which we also observe in the offline analysis conducted in Appendix B.2.



(a) 4 UAVs **(b)** 10 UAVs **(c)** 20 UAVs

Figure A.6: Percentage of the customer orders successfully executed by the system plotted against the distance threshold in pTeSSB with re-auctioning for the medium demand level in Table A.2. Results are shown for fleet sizes of (a) 4 UAVs (b) 10 UAVs and (C) 20 UAVs.



(a) 4 UAVs **(b)** 10 UAVs **(c)** 20 UAVs

Figure A.7: Percentage of the customer orders successfully executed by the system plotted against the distance threshold in pTeSSB with re-auctioning for the highest demand level in Table A.2. Results are shown for fleet sizes of (a) 4 UAVs (b) 10 UAVs and (c) 20 UAVs.

A.3.3. Robustness

Table A.3 shows the p-value and effect size (A) for our three claims across all fleet sizes and demand levels investigated. $\mathbf{H_{A1}}$ is statistically significant across all fleet sizes and demand cases, indicating that pTeSSI strongly outperforms TeSSI independently of the fleet size. The same is true for $\mathbf{H_{B1}}$, through which we confirm that the increase in the amount of deliveries executed by the system due to our periodic re-auctioning mechanism is robust to the size of the fleet. $\mathbf{H_C}$ is also always statistically significant, and as expected the effect sizes increase with the size of the fleet, confirming the conclusion drawn earlier that bundling is more effective for larger fleet sizes.

Demand case	Fleet size		$\mathbf{H_{A1}}$	$\mathbf{H_{B1}}$	$\mathbf{H_C}$	
					$\mathbf{H_{C1}}$	$\mathbf{H_{C2}}$
6 orders per UAV	4	p	1.36e-7	-	-	-
		A	0.95	-	-	-
	10 min (Nominal)	p	3.1e-8	-	-	-
		A	0.99	-	-	-
	20	p	3.25e-8	-	-	-
		A	1.0	-	-	-
12 orders per UAV	4	p	4.40e-8	2.46e-4	2.2e-2	3.5e-2
		A	0.97	0.74	0.66	0.64
	10 min (Nominal)	p	3.5e-8	5.87e-7	2.60e-4	3.10e-6
		A	1.0	0.83	0.73	0.81
	20	p	3.54e-8	5.03e-3	2.3e-3	5.06e-3
		A	1.0	0.99	0.71	0.99
18 orders per UAV	4	p	3.51e-8	8.03e-6	2.3e-2	9.9e-3
		A	1.0	0.78	0.76	0.68
	10 min (Nominal)	p	3.5e-8	2.82e-6	2.42e-2	4.16e-8
		A	1.0	0.86	0.62	0.97
	20	p	3.55e-8	5.01e-3	1.25e-5	5.06e-3
		A	1.0	1.0	0.83	1.0

Table A.3: Analysis of robustness of three main claims made in this work to large variations in the UAV fleet size.

B
Additional verification and validation

In the scientific paper, we discuss several efforts made to verify and validate our pTeSSI auction and related extensions proposed in this work. These include carefully investigating and explaining how key parameters of the auction mechanism affect system performance in simulation and whether this matches expectations, and comparing the performance of pTeSSI to the existing (non-probabilistic) TeSSI auction under a range of conditions. We also provide some further verification in Appendix A where we demonstrate that varying additional parameters results in effects that can be explained based on the theoretical properties of the auction as explained in the paper. In this chapter, we discuss a few additional verification and validation steps that were conducted but not yet discussed. In Appendix B.1 we discuss our validation of the effectiveness of the degree of dynamic controllability and in Appendix B.2 we discuss further analysis of pTeSSB.

B.1. Degree of dynamic controllability

The suitability of the degree of dynamic controllability as a predictor for the dispatch success rate of STNUs has already been validated empirically in [8] using a modified version of the CAR-SHARING and ROVERS datasets [143]. Nonetheless, as it is an approximate and extremely recent method, it is still important to validate its performance when applied to networks of the type explored in our work (see section II of the paper).

To do this, we generate several schedules by creating customer orders with different time windows and compute their degree of dynamic controllability (DDC). We then simulate dispatch on these networks by applying the early first strategy proposed in [110] and used by our agents in the online simulations. Since the strategy is guaranteed to succeed on dynamically controllable networks, the agent should be able to successfully dispatch its schedule exactly the amount of times that it is dynamically controllable in practice - that is, those times in which the realization of the contingent edges is such that conflicts are avoided at execution. We generate 200 schedules and for each schedule, we simulate dispatch 300 times. We record the dispatch success rate - the percentage of times we are able to successfully dispatch the schedule - and compare it to the DDC. If the relaxations made by `ODC-Relax` are optimal, then the dispatch success rate should equal the DDC.

Figure B.1 shows the results for the case in which the edges are uniformly distributed. The graph shows a strong linear relationship $R^2 = 0.992$, with the DDC consistently predicting the dispatch success rate within 5% (dotted red lines) of the value we measure empirically. From this analysis we conclude that the DDC is a very strong predictor for the true dispatch success rate when the edges are uniformly distributed.

We perform the same analysis in the case where the edges are normally distributed. As explained in section II.4, we construct STNU edges from the PSTN by generating bounds which capture 1-α probability mass of the underlying distribution, truncating an equal amount of mass from each tail. In the online experiments, we find that low values of α lead to the best performance, so we use the lowest value tested, namely $\alpha = 0.01$. We therefore construct STNU edges which capture 99% of the underlying normal distribution. Here, we use the same value of α and show the results in Figure B.2. While there is still evidence that the DDC is a strong predictor of the empirical dispatch success rate and the plot exhibits a strong linear relationship ($R^2 = 0.946$), we observe larger errors (peaking at $\approx +10\%$ and $\approx -15\%$ as shown by the dotted red lines in Figure B.2) and note that the curve displays an 'S' shape with inflection point at the $\approx 50\%$ dispatch success rate mark. When the dispatch success rate is less than 50%, the DDC consistently over-estimates it, and under-estimates it when it is larger than 50%. The errors are very close to zero near the inflection point, when the dispatch success rate is between 40% and 60%.

B.2. Bundling

In order to further validate the bundling mechanism in pTeSSB, we analyze its performance on an offline version of the task allocation problem discussed in the paper and verify whether we obtain similar results when varying the distance threshold and the size of the task set on auction. We consider the problem in which tasks

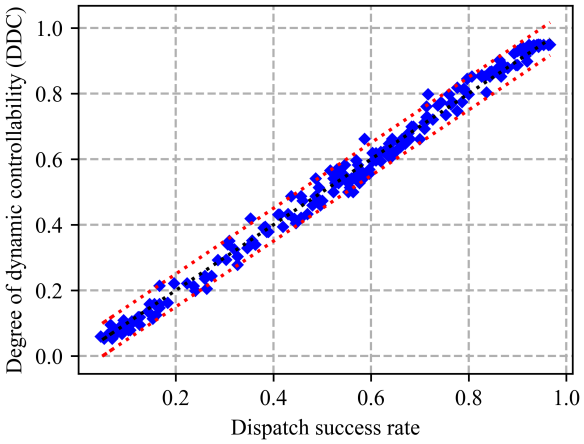


Figure B.1: Empirical dispatch success rate vs degree of dynamic controllability (DDC) for uniformly distributed edges. Dotted red lines are for reference and show the interval spanning ± 5% from the true dispatch success rate.

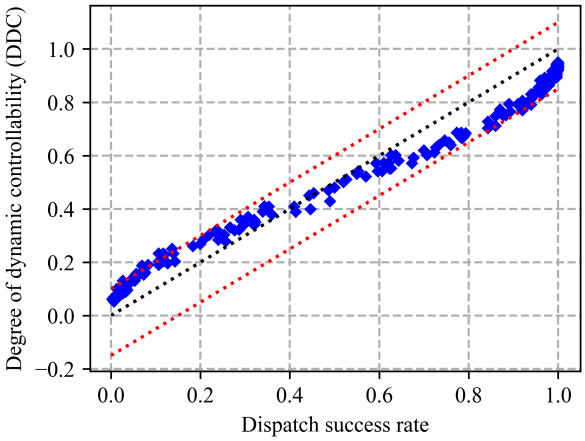


Figure B.2: Empirical dispatch success rate vs degree of dynamic controllability (DDC) for normally distributed edges. Dotted red lines are for reference and show the interval spanning +10% and -15% from the true dispatch success rate.

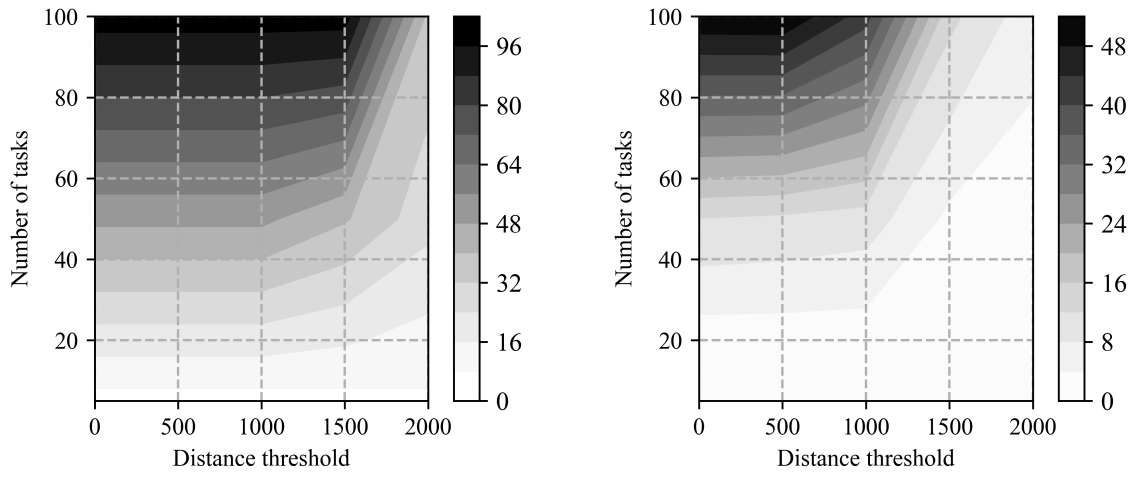
need to be auctioned at once, and examine the number of tasks that pTeSSB is able to allocate as well as its runtime. We fix the number of UAV bidders at 50.

Figure B.3 shows a contour plot of the tasks allocated (a) and auction runtime (b) with varying distance threshold and size of the task set on auction. The offline experiment validates and sheds further light on the trends discussed in the scientific paper. We observe a similar trade-off appearing when varying the distance threshold. Increasing the distance threshold leads to larger and looser clusters, decreasing the auction duration but also the number of tasks that pTeSSB can allocate. Note that in the online experiments of the scientific paper we focus on the number of deliveries executed by the system, whereas here we do not simulate dispatch and look at the number of tasks that can be *allocated* to the agents. Observing the same trend here is confirmation that the performance drop observed in the online experiments is mainly due due to the fact that larger bundles are more difficult for the agents to feasibly insert in their schedules. While conducting these experiments, we also verify that the bundles always constitute partial schedules with risk at most equal to the agents' risk threshold of $R = 0.5$, which is further verification of our implementation of the bundling mechanism.

Remarkably, for the larger problems (80+ tasks), we observe barely any drop in the number of allocated tasks up to a distance threshold of ≈ 1500, but a very large (nearly four-fold) drop in the auction duration. In the smaller problems (60 or less tasks), the trade-off is much less favorable, as performance starts degrading from a distance threshold of ≈ 1000, by which we have generated a much more modest improvement in runtime. This reinforces one of the findings from our paper, namely that bundling is more effective for larger number of tasks. In the online experiments, pTeSSB was only able to generate a substantial drop in the auction runtime

while retaining a large portion of the performance advantage in the higher demand cases. The fact that larger number of tasks lead to better bundling is also suggested by the sensitivity analysis in Appendix A.3 and is not surprising, as the larger the size of the task set on auction, the more likely that there will be inter-task synergies that will translate to good partial schedules.

It is also important to note that in the offline experiments the timelines of the tasks were naturally aligned, as all orders are generated at the same time and the deadlines therefore have the same distribution. In the online experiments, on the other hand, several of the tasks are re-auctioned opportunistically by the agents, likely leading to less aligned deadlines, which may have also made it more difficult to generate compact clusters compared to the offline case.



(a) Number of tasks allocated [-] **(b)** Auction duration [s]

Figure B.3: Effect of distance threshold and number of tasks on auction on the number of tasks allocated and runtime of pTeSSB in the offline experiments.

C
Variability of simulation results

In this chapter, we show the evolution of the coefficient of variation for all experiments discussed in Section V of the scientific paper. In experiment runs where multiple combinations of parameters are explored, we show the evolution of the coefficient of variation in the most volatile case. Each plot in this chapter refers to runs summarized in a plot or table of the paper, which is mentioned in the caption.

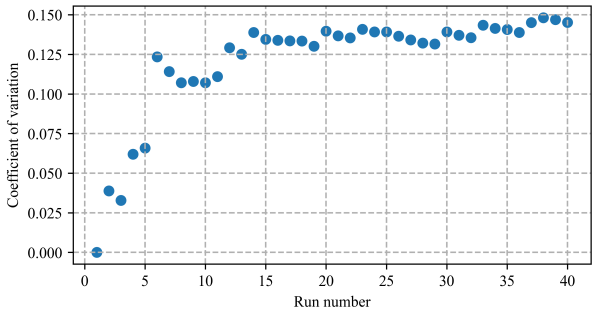


Figure C.1: Evolution of coefficient of variation for the results shown in Figure 3 of the paper. The bidding rule is `MAX-T`. The case shown is with the highest demand (180 orders), highest risk threshold (0.9) and lowest value of ρ (0).

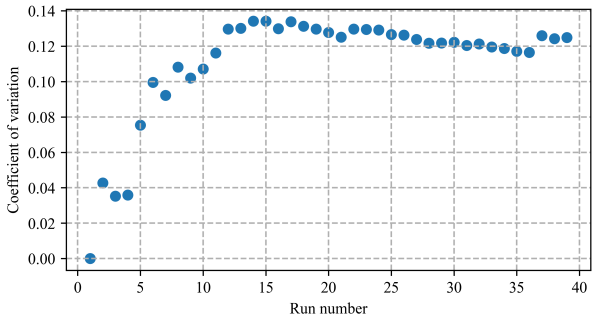


Figure C.2: Evolution of coefficient of variation for the results shown in Figure 4 of the paper. The bidding rule is `SUM-DIST`. The case shown is with the highest demand (180 orders), highest risk threshold (0.9) and lowest value of ρ (0).

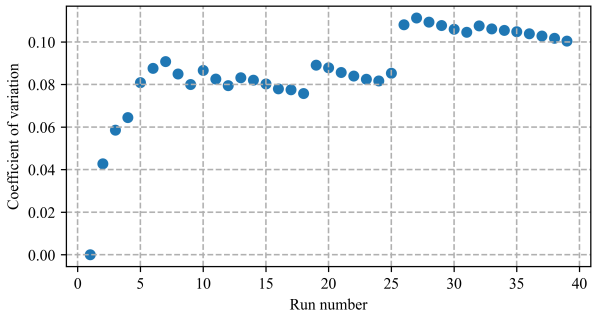


Figure C.3: Evolution of coefficient of variation of the total deliveries. Results shown for pTeSSI with the `MAX-T` bidding rule in Table 2 of the paper. The case shown is with the highest demand (180 orders).

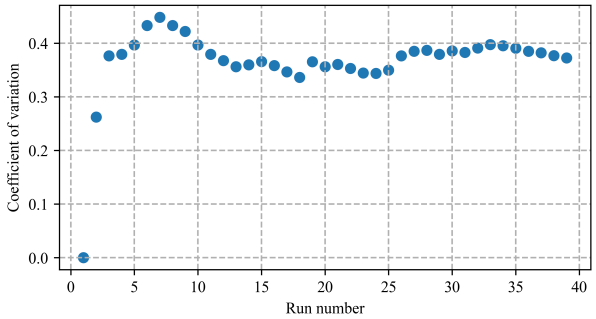


Figure C.4: Evolution of coefficient of variation of the average auction duration. Results shown for pTeSSI with the `MAX-T` bidding rule in Table 2 of the paper. The case shown is with the highest demand (180 orders).

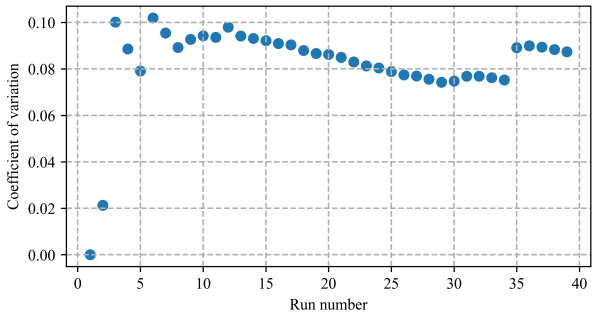


Figure C.5: Evolution of coefficient of variation of the total deliveries. Results shown for pTeSSI with the `SUM-DIST` bidding rule in Table 2 of the paper. The case shown is with the highest demand (180 orders).

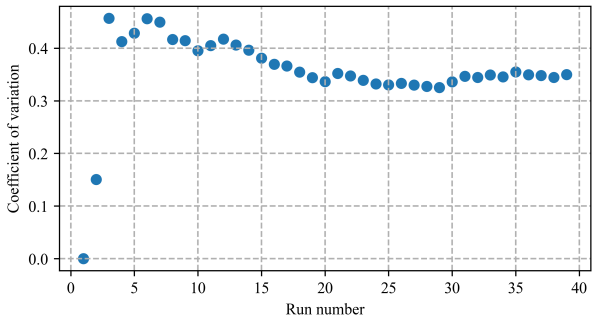


Figure C.6: Evolution of coefficient of variation of the average auction duration. Results shown for pTeSSI with the `SUM-DIST` bidding rule in Table 2 of the paper. The case shown is with the highest demand (180 orders).

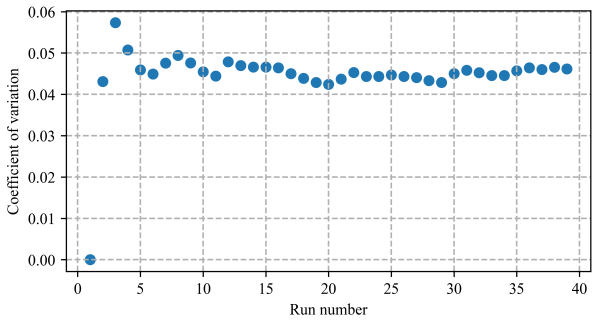


Figure C.7: Evolution of coefficient of variation of the distance per delivery. Results shown for pTeSSI with the `SUM-DIST` bidding rule in Table 2 of the paper. The case shown is with the highest demand (180 orders).

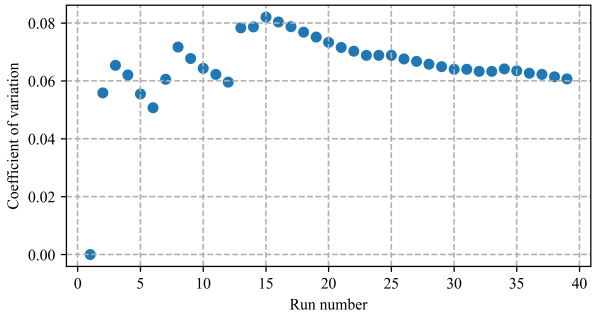


Figure C.8: Evolution of coefficient of variation of the total deliveries. Results shown for TeSSI with the `MAX-T` bidding rule in Table 2 of the paper. The case shown is with the highest demand (180 orders).

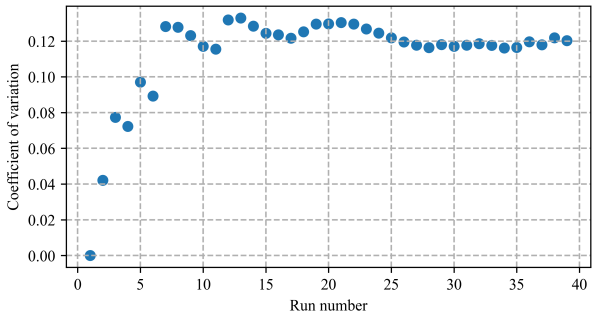


Figure C.9: Evolution of coefficient of variation of the average auction duration. Results shown for TeSSI with the `MAX-T` bidding rule in Table 2 of the paper. The case shown is with the highest demand (180 orders).

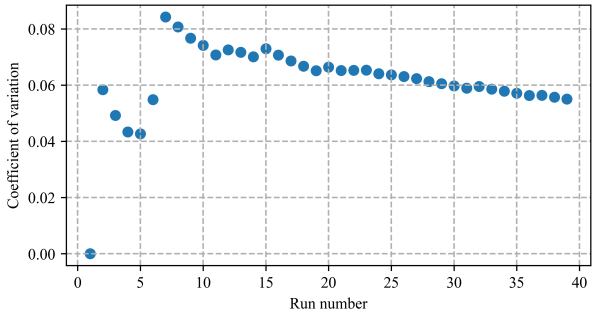


Figure C.10: Evolution of coefficient of variation of the distance per delivery. Results shown for TeSSI with the `MAX-T` bidding rule in Table 2 of the paper. The case shown is with the highest demand (180 orders).

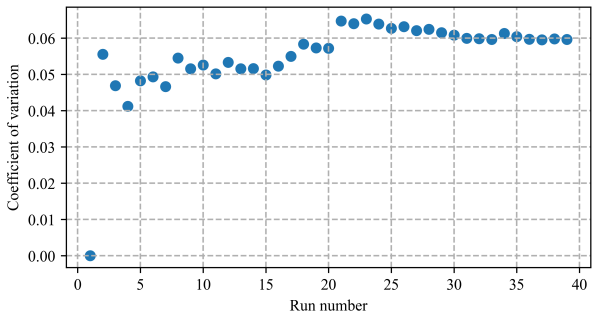


Figure C.11: Evolution of coefficient of variation of the total deliveries. Results shown for TeSSI with the `SUM-DIST` bidding rule in Table 2 of the paper. The case shown is with the highest demand (180 orders).

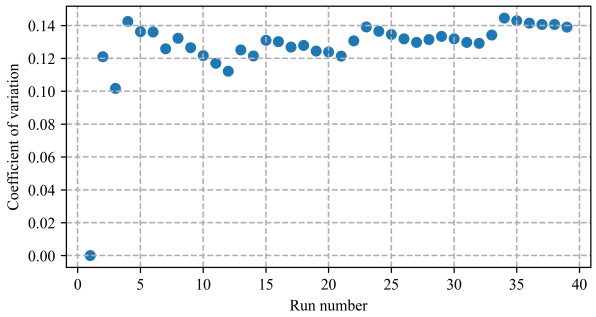


Figure C.12: Evolution of coefficient of variation of the average auction duration. Results shown for TeSSI with the `SUM-DIST` bidding rule in Table 2 of the paper. The case shown is with the highest demand (180 orders).

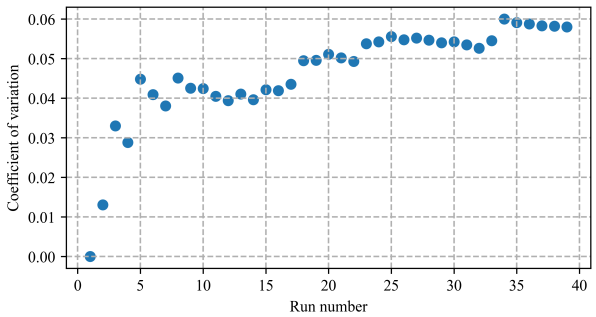


Figure C.13: Evolution of coefficient of variation of the distance per delivery. Results shown for TeSSI with the `SUM-DIST` bidding rule in Table 2 of the paper. The case shown is with the highest demand (180 orders).

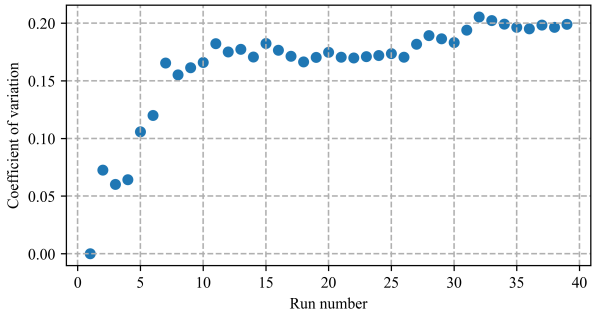


Figure C.14: Evolution of coefficient of variation of the total deliveries. Results shown for pTeSSI in Figure 5 of the paper with the highest contingent edge width and highest demand level (180 orders).

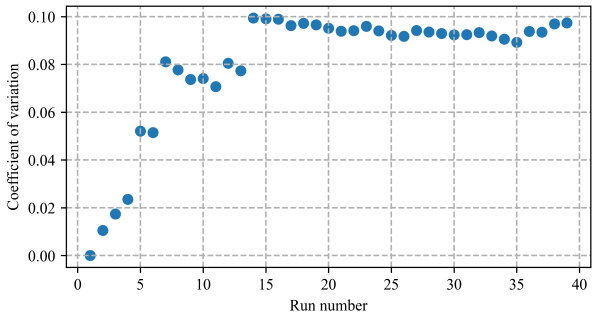


Figure C.15: Evolution of coefficient of variation of the total deliveries. Results shown for TeSSI in Figure 5 of the paper with the highest contingent edge width and highest demand level (180 orders).

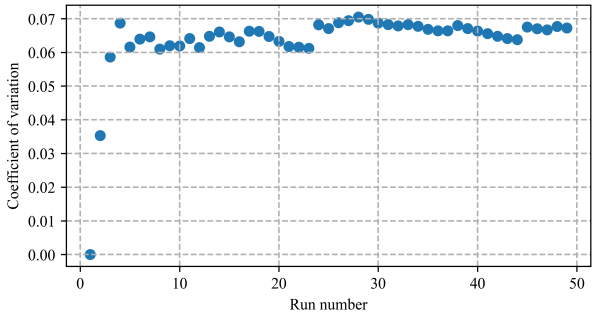


Figure C.16: Evolution of coefficient of variation of the total deliveries. Results shown for pTeSSI-re in Figure 7 of the paper with the lowest reauction threshold and highest demand level (240 orders).

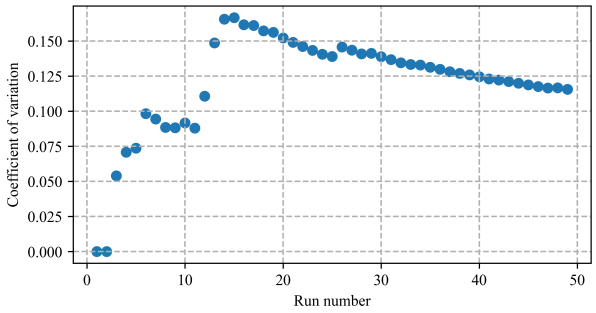


Figure C.17: Evolution of coefficient of variation of the total deliveries. Results shown for pTeSSB-re in Figure 8 of the paper with the lowest distance threshold and highest demand level (240 orders).

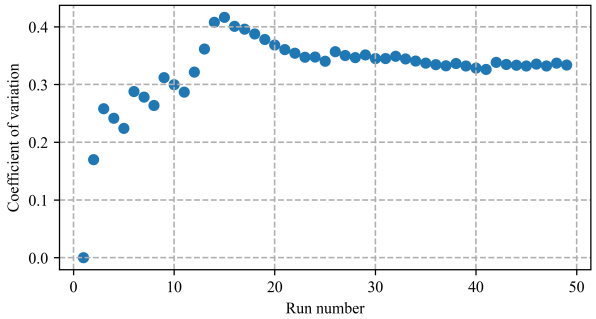


Figure C.18: Evolution of coefficient of variation of the average auction duration. Results shown for pTeSSB-re in Figure 8 of the paper with the lowest distance threshold and highest demand level (240 orders).

Bibliography

[1] (2016). Leading the world to 5G: Evolving cellular technologies for safer drone operation. Qualcomm. URL.

[2] (2017). UberEats Netherlands Guaranteed Earnings. UberEats Netherlands. URL.

[3] (2020a). Amsterdam Climate Data. MeteoBlue. URL.

[4] (2020b). Decentralized Multi-Agent Path Finding for UAV Traffic Management. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems. Preprint*, Auckland, New Zealand. IFAAMAS.

[5] (2020c). Uber Announces Results for Fourth Quarter and Full Year 2019. Uber Technologies, Inc. URL.

[6] Abrache, J., Crainic, T. G., Gendreau, M., and Rekik, M. (2007). Combinatorial auctions. *Annals of Operations Research*, 153(1):131–164.

[7] Agatz, N., Bouman, P., and Schmidt, M. (2015). Optimization Approaches for the Traveling Salesman Problem with Drone. *SSRN Electronic Journal*.

[8] Akmal, S., Ammons, S., Li, H., Gao, M., Popowski, L., and Boerkoel, J. C. (2020). Quantifying controllability in temporal networks with uncertainty. *Artificial Intelligence*, 289.

[9] Amir, O., Sharon, G., and Stern, R. (2015). Multi-Agent Pathfinding as a Combinatorial Auction. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2003–2009, Austin, Texas. AAAI.

[10] Arslan, G., Marden, J. R., and Shamma, J. S. (2007). Autonomous vehicle-target assignment: A game-theoretical formulation. *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, 129(5):584–596.

[11] Barer, M., Sharon, G., Stern, R., and Felner, A. (2014). Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. *Frontiers in Artificial Intelligence and Applications*, 263:961–962.

[12] Bazzan, A. L. and Klügl, F. (2014). A review on agent-based technology for traffic and transportation. *Knowledge Engineering Review*, 29(3):375–403.

[13] Benoit, A., Robert, Y., and Vivien, F. (2013). *A guide to algorithm design : paradigms, methods, and complexity analysis*. Chapman & Hall/CRC.

[14] Berhault, M., Huang, H., Keskinocak, P., Koenig, S., Elmaghraby, W., Griffin, P., and Kleywegt, A. (2003). Robot exploration with combinatorial auctions. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 1957–1962.

[15] Bnaya, Z. and Felner, A. (2014). Conflict-Oriented Windowed Hierarchical Cooperative A*. In *IEEE International Conference on Robotics & Automation (ICRA)*, pages 3743–3748. IEEE.

[16] Bnaya, Z., Stern, R., Felner, A., Zivan, R., and Okamoto, S. (2013). Multi-Agent Path Finding for Self Interested Agents. In *Proceedings of the 6th Annual Symposium on Combinatorial Search (SoCS)*, pages 38–46. AAAI.

[17] Borrajo, D. and Fernández, S. (2019). Efficient approaches for multi-agent planning. *Knowledge and Information Systems*, 58(2):425–479.

[18] Botea, A. and Surynek, P. (2015). Multi-Agent Path Finding on Strongly Biconnected Digraphs. In *29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 2024–2030, Austin, Texas. AAAI.

[19] Boyarski, E., Felner, A., Stern, R., Sharon, G., Tolpin, D., Betzalel, O., and Shimony, E. (2015). ICBS: Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding Introduction and Overview. In *Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 740–746.

[20] Brunet, L., Choi, H.-L., and How, J. (2008). Consensus-Based Auction Approaches for Decentralized Task Assignment. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii.

[21] Bry, A., Richter, C., Bachrach, A., and Roy, N. (2015). Aggressive Flight of Fixed-Wing and Quadrotor Aircraft in Dense Indoor Environments. *International Journal of Robotics Research (IJRR)*, 37(7):969–1002.

[22] Buckman, N., Choi, H.-L., and How, J. P. (2019). Partial Replanning for Decentralized Dynamic Task Allocation. In *AIAA SciTech Forum*, San Diego, California, USA. AIAA.

[23] Burwell, K. (2019). Multi-Agent Path Finding for Unmanned Aerial Vehicles (Master Thesis).

[24] Carson, B. (2019). First Look: Uber Unveils New Design For Uber Eats Delivery Drone. Forbes. URL.

[25] Carsten, J., Ferguson, D., and Stentz, A. (2006). 3D Field D*: Improved path planning and replanning in three dimensions. In *International Conference on Intelligent Robots and Systems*, pages 3381–3386, Beijing, China. IEEE.

[26] Chae, S. and Yang, J.-A. (1988). The Unique Perfect Equilibrium of an N-Person Bargaining Game. *Economics Letters*, 28(3):221–223.

[27] Chae, S. and Yange, J.-A. (1994). An n-person pure bargaining game. *Journal of Economic Theory*, 62:86–102.

[28] Chakrabarty, A., Ippolito, C., Baculi, J., Krishnakumar, K., and Hening, S. (2019). Vehicle to vehicle (V2V) communication for collision avoidance for multi-copters flying in UTM-TCL4. In *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics Inc, AIAA.

[29] Chapman, A., Micillo, R., Kota, R., and Jennings, N. (2009). Decentralised Dynamic Task Allocation: A Practical Game–Theoretic Approach. In *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 915–922, Budapest, Hungary. IFAAMAS.

[30] Cheng, Q., Yin, D., Yang, J., and Shen, L. (2017). An Auction-Based Multiple Constraints Task Allocation Algorithm for Multi-UAV System. In *Proceedings - 2016 International Conference on Cybernetics, Robotics and Control (CRC)*, pages 1–5, Hong Kong, China. Institute of Electrical and Electronics Engineers Inc.

[31] Chouhan, S. S. and Niyogi, R. (2015). DMAPP: A Distributed Multi-Agent Path Planning Algorithm. In *28th Australasian Joint Conference on Artificial Intelligence*, volume 9457. Springer Verlag.

[32] Chouhan, S. S. and Niyogi, R. (2017). DiMPP: a complete distributed algorithm for multi-agent path planning. *Journal of Experimental and Theoretical Artificial Intelligence*, 29(6):1129–1148.

[33] Cohn, P., Green, A., Langstaff, M., and Roller, M. (2017). Commercial drones are here: The future of unmanned aerial systems. McKinsey & Company.

[34] Colaitis, O., Dominguez Puerta, E., Balakrishnan, H., Baumgartner, M., Diaz-Gomez, E., Dombroff, M., Hirschberg, M., Kochendorfer, M., Lamprecht, A., Cto, A. ., Marcus, B., Co-Founder, A. ., Nakamura, C. H., Panchadsaram, R., Perkins, K., Byers, C., and Reuter, T. (2018). Blueprint for the Sky - The Roadmap for the Safe Integration of Autonomous Aircraft. Technical report, Airbus.

[35] Cormen, T. H. and Cormen, T. H. (2001). *Introduction to algorithms*. MIT Press.

[36] D'Andrea, R. (2014). Guest Editorial: Can Drones Deliver? *IEEE Transactions on Automation Science and Engineering*, 11(3):647–648.

[37] de Freitas, J. C. and Penna, P. H. V. (2020). A variable neighborhood search for flying sidekick traveling salesman problem. *International Transactions in Operational Research*, 27(1):267–290.

[38] de Souza, U., Eduardo, P., and Carvalho Chanel, P. (2016). A Game Theoretical Formulation of a Decentralized Cooperative Multi-Agent Surveillance Mission. In *4th Workshop on Distributed and Multi-Agent Planning (DMAP)*, London, UK.

[39] De Wilde, B., ter Mors, A. W., and Witteveen, C. (2014). Push and Rotate: a Complete Multi-agent Pathfinding Algorithm. *Journal of Artificial Intelligence Research*, 51:443–492.

[40] Dechter, R. and Pearl, J. (1985). Generalized Best-First Search Strategies and the Optimality of A. *Journal of the ACM (JACM)*, 32(3):505–536.

[41] Desaraju, V. R. and How, J. P. (2011). Decentralized Path Planning for Multi-Agent Teams in Complex Environments using Rapidly-exploring Random Trees. In *IEEE International Conference on Robotics and Automation*, pages 4956–4961, Shanghai, China. IEEE.

[42] Dias, M. B. and Stentz, A. (2000). A Free Market Architecture for Distributed Control of a Multirobot System. In *Proceedings of the International Conference on Intelligent Autonomous Systems*, pages 115–122.

[43] Dias, M. B., Zlot, R., Kalra, N., and Stentz, A. (2006). Market-Based Multirobot Coordination: A Survey and Analysis. *Proceedings of the IEEE*, 94(7):1257–1270.

[44] Doole, M., Ellerbroek, J., and Hoekstra, J. (2018). Drone Delivery: Urban airspace traffic density estimation. In *8th SESAR Innovation Days*. APA.

[45] Dresner, K. and Stone, P. (2008). A Multiagent Approach to Autonomous Intersection Management. *Journal of Artificial Intelligence Research*, 31:591–656.

[46] Du, J., Zhou, L., Qu, P., Shi, Z., and Lin, Y. (2010). Task allocation in multi-agent systems with swarm intelligence of social insects. In *6th International Conference on Natural Computation (ICNC)*, volume 8, pages 4322–4326.

[47] Dutta, B. and Gevers, L. (1981). On majority rules and perfect equilibrium allocations of a shrinking cake. Technical report, Cahiers de la Faculte des Science Economiques de Namur.

[48] Erdem, E., Kisa, D. G., Oztok, U., Schüller, P., and Schüller, S. (2013). A General Formal Framework for Pathfinding Problems with Multiple Agents. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*. AAAI.

[49] Fatima, S., Kraus, S., and Wooldridge, M. (2015). *Principles of Automated Negotiation*. Cambridge University Press.

[50] Fatima, S., Wooldridge, M., and Jennings, N. R. (2006a). On Efficient Procedures for Multi-Issue Negotiation. In *Proceedings of the Eighth International Workshop on Agent Mediated Electronic Commerce (AMEC)*, page 71–85, Hakodate, Japan.

[51] Fatima, S., Wooldridge Michael, and Jennings, N. R. (2006b). Multi-Issue Negotiation with Deadlines. *Journal of Artificial Intelligence Research*, 27:381–417.

[52] Federal Aviation Administration (FAA) (2020). Unmanned Aircraft System (UAS) Traffic Management (UTM) - Concept of Operations. Technical report, Washington, DC.

[53] Felner, A., Stern, R., Ben-Yair, A., Kraus, S., and Netanyahu, N. (2004). PHA*: Finding the Shortest Path with A* in An Unknown Physical Environment. *Journal of Artiicial Intelligence Research*, 21:631–670.

[54] Felner, A., Stern, R., Shimony, S. E., Boyarski, E., Goldenberg, I. M., Sharon, G., Sturtevant, N., Wagner, G., and Surynek, P. (2017). Search-based Optimal Solvers for the Multi-agent Pathfinding Problem: Summary and Challenges. In *Tenth Annual Symposium on Combinatorial Search*. AAAI.

[55] Ferner, C., Wagner, G., and Choset, H. (2013). ODrM* Optimal Multirobot Path Planning in Low Dimensional Search Spaces. In *International Conference on Robotics and Automation (ICRA)*, pages 3839–3844. IEEE.

[56] Fershtman, C. (2000). A Note on Multi-Issue Two-Sided Bargaining: Bilateral Procedures. *Games and Economic Behavior*, 30:216–227.

[57] FIshburn, P. C. (1988). Normative thoeries of decision making under risk and uncertainty. In Bell, D. E., Raiffa, H., and Tversky, A., editors, *Decision making: Descriptive, normative, and prescriptive interactions*. Cambridge University Press.

[58] Gerkey, B. P. and Mataric, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9):939–954.

[59] Goldenberg, M., Felner, A., Stern, R., Sharon, G., and Schaeffer, J. (2012). A* Variants for Optimal Multi-Agent Pathfinding. In *Proceedings of the 5th Annual Symposium on Combinatorial Search (SoCS)*. AAAI.

[60] Goldenberg, M., Felner, A., Stern, R., Sharon, G., Sturtevant, N., Holte, R., and Schaeffer, J. (2014). Enhanced Partial Expansion A*. *The Journal of Artificial Intelligence Research (JAIR)*, 50.

[61] Goldie-Scot, L. (2019). A Behind the Scenes Take on Lithium-ion Battery Prices. BloombergNEF. URL.

[62] Ha, Q. M., Deville, Y., Pham, Q. D., and Hà, M. H. (2020). A Hybrid Genetic Algorithm for the Traveling Salesman Problem with Drone. *Journal of Heuristics*, 26:219–247.

[63] Harsanyi, J. C. (1967). Games of incomplete information played by Bayesian players. *Management Science*, 14(3):159–182.

[64] Harsanyi, J. C. and Selten, R. (1972). A Generalized Nash Solution for Two-Person Bargaining Games with Incomplete Information. *Management Science*, 18(5-part-2):80–106.

[65] Hart, P. E., Nilsson, N. J., and Bertram, R. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions Of Systems Science And Cybernetics*, 19(2):874–890.

[66] Hawkins, A. J. (2019a). Uber unveils a new look for its food delivery drones. The Verge. URL.

[67] Hawkins, A. J. (2019b). Wing's delivery drones take flight for the first time in Virginia. The Verge. URL.

[68] Heap, B. (2013). *Sequential Single-Cluster Auctions for Multi-Robot Task Allocation*. PhD thesis, University of New South Wales (UNSW).

[69] Heap, B. and Pagnucco, M. (2013). Repeated Sequential Single-Cluster Auctions with Dynamic Tasks for Multi-Robot Task Allocation with Pickup and Delivery. In Klusch, M., Thimm, M., and Paprzycki, M., editors, *German Conference on Multiagent System Technologies*, pages 87–100, Koblenz, Germany. Springer.

[70] Hernández, E., Barrientos, A., and Cerro, J. D. (2014). Selective Smooth Fictitious Play: An approach based on game theory for patrolling infrastructures with a multi-robot system. *Expert Systems with Applications*, 41(6):2897–2913.

[71] Ho, F., Geraldes, R., Goncalves, A., Rigault, B., Oosedo, A., Cavazza, M., and Prendinger, H. (2019a). Pre-Flight Conflict Detection and Resolution for UAV Integration in Shared Airspace: Sendai 2030 Model Case. *IEEE Access*, 7:170226–170237.

[72] Ho, F., Salta, A., Geraldes, R., Goncalves, A., Cavazza, M., and Prendinger, H. (2019b). Multi-Agent Path Finding for UAV Traffic Management. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 131–139, Montreal, Canada.

[73] Jacobs, K., Warner, S., Rietra, M., Mazza, L., Buvat, J., Khadikar, A., Cherian, S., and Khemka, Y. (2019). The last-mile delivery challenge - Giving retail and consumer product customers a superior delivery experience without impacting profitability The last-mile delivery challenge: Giving retail and consumer product customers a superior delivery experience without impacting profitability. Technical report, Capgemini Research Institute.

[74] Jansen, M. and Sturtevant, N. (2008a). A new approach to cooperative pathfinding. In *7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 3, pages 1401–1404, Estoril, Portugal. IFAAMAS.

[75] Jansen, M. R. and Sturtevant, N. R. (2008b). Direction Maps for Cooperative Pathfinding. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 185–190.

[76] Jiang, X., Zhou, Q., and Ye, Y. (2017). Method of task assignment for UAV based on particle swarm optimization in logistics. In *ACM International Conference Proceeding Series*, pages 113–117. Association for Computing Machinery.

[77] Johnson, L., Choi, H. L., Ponda, S., and How, J. P. (2012). Allowing non-submodular score functions in distributed task allocation. In *Proceedings of the IEEE Conference on Decision and Control*, pages 4702–4708.

[78] Johnson, L., Ponda, S., and Choi, H.-L. (2020). Consensus-Based Bundle Algorithm (CBBA). MIT Aerospace Controls Laboratory (ACL). URL.

[79] Johnson, M., Prevot, T., Jung, J., Rios, J., Mercer, J., Homola, J., Mulfinger, D., and Kopardekar, P. (2017). Flight Test Evaluation of an Unmanned Aircraft System Traffic Management (UTM) Concept for Multiple Beyond-Visual-Line-of-Sight Operations. In *Twelfth USA/Europe Air Traffic Management Research and Development Seminar*.

[80] Jun, B. (1987). *A structural consideration on 3-person bargaining*. PhD thesis, University of Pennsylvania.

[81] Keeney, R. and Raiffa, H. (1991). *Structuring and analyzing values for multi-issue negotiation analysis*. University of Michigan Press, Ann Arbor, MI.

[82] Khorshid, M. M., Holte, R. C., and Sturtevant, N. (2011). A Polynomial-Time Algorithm for Non-Optimal Multi-Agent Pathfinding. In *Proceedings of the Fourth Annual Symposium on Combinatorial Search (SOCS)*, pages 76–83. AAAI.

[83] Khoufi, I., Laouiti, A., and Adjih, C. (2019). A Survey of Recent Extended Variants of the Traveling Salesman and Vehicle Routing Problems for Unmanned Aerial Vehicles. *Drones*, 3(3).

[84] Koenig, S., Keskinocak, P., and Tovey, C. (2010). Progress on Agent Coordination with Cooperative Auctions. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, pages 1713–1717, Atlanta, Georgia, USA. AAAI.

[85] Koenig, S., Tovey, C., Zheng, X., and Sungur, I. (2007). Sequential Bundle-Bid Single-Sale Auction Algorithms for Decentralized Control. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1359–1365, Hyderabad, India.

[86] Koenig, S., Zheng, X., Tovey, C., Borie, R., Kilby, P., Markakis, V., and Keskinocak, P. (2008). Agent Coordination with Regret Clearing. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 101–107. AAAI.

[87] Kopardekar, P. (2016). Unmanned Aerial System Traffic Management System. Talks at Google (YouTube channel). URL.

[88] Kuhn, H. W. (1995). The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, pages 83–97.

[89] Lagoudakis, M. G., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A., Koenig, S., Tovey, C., Meyerson, A., and Jain, S. (2005). Auction-Based Multi-Robot Routing. In *Proceedings of the International Conference on Robotics: Science and Systems*, pages 343–350.

[90] Lardinois, F. (2019). A first look at Amazon's new delivery drone. TechCrunch. URL.

[91] Lavalle, S. M. and Hutchinson, S. A. (1998). Optimal Motion Planning for Multiple Robots Having Independent Goals. *IEEE Transactions On Robotics And Automation*, 14(6).

[92] Lewis, N. (2020). A tech company engineered drones to deliver vital COVID-19 medical supplies to rural Ghana and Rwanda in minutes. Business Insider. URL.

[93] Ligterink, N. E. (2016). On-road determination of average Dutch driving behaviour for vehicle emissions. Technical report, TNO, Utrecht, Netherlands.

[94] Liu, L. and Shell, D. (2010). Assessing Optimal Assignment under Uncertainty: An Interval-based Algorithm. *The International Journal of Robotics Research*, 30.

[95] Locascio, D., Levy, M., Ravikumar, K., German, B., Briceno, S. I., and Mavris, D. N. (2016). Evaluation of Concepts of Operations for sUAS Package Delivery. In *16th AIAA Aviation Technology, Integration, and Operations Conference*, Washington, D.C., USA. AIAA.

[96] Luders, B., Levine, D., Ponda, S., and How, J. P. (2011). Information-rich Task Allocation and Motion Planning for Heterogeneous Sensor Platforms. In *Infotech@Aerospace*, St Louis, Missouri. Massachusetts Institute of Technology (MIT), AIAA.

[97] Luna, R. and Bekris, K. E. (2011). Efficient and Complete Centralized Multi-Robot Path Planning. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems.*, pages 3268–3275. IEEE.

[98] Luo, H., Liang, Z., Zhu, M., Hu, X., and Wang, G. (2018). Integrated optimization of unmanned aerial vehicle task allocation and path planning under steady wind. *PLoS ONE*, 13(3).

[99] Ma, H., Li, J., Kumar, T. K. S., and Koenig, S. (2017). Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Sao Paulo, Brazil. IFAAMAS.

[100] Ma, H., Wagner, G., Felner, A., Li, J., Satish Kumar, T. K., and Koenig, S. (2018). Multi-Agent Path Finding with Deadlines. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pages 417–423, Stockholm, Sweden.

[101] Máhr, T. (2011). *Vehicle routing under uncertainty*. PhD thesis, Delft University of Technology.

[102] Masschler, M., Solan, E., and Zamir, S. (2013). *Game Theory*. Cambridge University Press.

[103] Mei, W. and Zhang, R. (2019). Uplink Cooperative NOMA for Cellular-Connected UAV. *IEEE Journal of Selected Topics in Signal Processing*, 13(3):644–656.

[104] Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *International Conference on Robotics and Automation (ICRA)*, pages 2520–2525, Shanghai, China. IEEE.

[105] Melvin, J., Keskinocak, P., Koenig, S., Tovey, C., and Yuksel Ozkaya, B. (2007). Multi-Robot Routing with Rewards and Disjoint Time Windows. In *Conference on Intelligent Robots and Systems*, pages 2332–2337, San Diego, USA. IEEE.

[106] Mes, M., van der Heijden, M., and Schuur, P. (2013). Interaction between intelligent agent strategies for real-time transportation planning. *Central European Journal of Operations Research*, 21:337–358.

[107] Mills-Tettey, A. G., Stent, A., and Dias, M. (2007). The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs. Technical report, Carnegie Mellon University (CMU) Robotics Institute.

[108] Nair, R., Ito, T., Tambe, M., and Marsella, S. (2002). Task allocation in the rescue simulation domain: A short note. In *RoboCup-2001: Robot Soccer World Cup V*. Springer.

[109] Nanjanath, M., Gini, M., and Nunes, E. (2012). Auctioning robotic tasks with overlapping time windows. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, Valencia, Spain. IFAAMAS.

[110] Nilsson, M., Kvarnström, J., and Doherty, P. (2014). Classical Dynamic Controllability Revisited - A Tighter Bound on the Classical Algorithm. In *Proc. of 6th International Conference on Agents and Artificial Intelligence (ICAART-14)*, pages 130–141, Angers, France.

[111] Nixon, A. (2020). How Much Does a Drone Insurance Cost? BestDroneforTheJob.com.

[112] Nunes, E. and Gini, M. (2015). Multi-Robot Auctions for Allocation of Tasks with Temporal Constraints. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2110–2116, Austin, Texas, USA. AAAI.

[113] Olfati-Saber, R., Fax, J. A., and Murray, R. M. (2007). Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233.

[114] Osborne, M. J. and Rubinstein, A. (1990). *Bargaining and Markets*. Academic Press, London.

[115] Oyola, J., Arntzen, H., and Woodruff D L (2017). The stochastic vehicle routing problem, a literature review, part II: solution methods. *EURO Journal on Transportation and Logistics*, 6(4):349–388.

[116] Oyola, J., Arntzen, H., and Woodruff D L (2018). The stochastic vehicle routing problem, a literature review, part I: models. *EURO Journal on Transportation and Logistics*, 7(3):193–221.

[117] Parkes, D. C. and Ungar, L. H. (2000). Iterative Combinatorial Auctions: Theory and Practice. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pages 74–81, Austin, Texas, USA. AAAI.

[118] Paul, G. (2019). Verizon wants to become the first carrier to use its 5G network to connect 1 million drone flights. Business Insider. URL.

[119] Peterson, K. (2020). UPS Flight Forward, CVS To Launch Residential Drone Delivery Service In Florida Retirement Community To Assist In Coronavirus Response. UPS. URL.

[120] Ponda, S. (2012). *Robust Distributed Planning Strategies for Autonomous Multi-Agent Teams*. PhD thesis, Massachussets Institute of Technology (MIT).

[121] Ponsati, C. and Watson, J. (1997). Multiple-Issue Bargaining and Axiomatic Solutions. *International Journal of Game Theory*, 26:501–524.

[122] Porter, J. (2020). Alphabet's nascent drone delivery service is booming. The Verge. URL.

[123] Prehofer, C. and Bettstetter, C. (2005). Self-organization in communication networks: Principles and design paradigms. *IEEE Communications Magazine*, 43(7):78–85.

[124] Prevot, T., Rios, J., Kopardekar, P., Robinson III, J. E., Johnson, M., and Jung, J. (2016). Unmanned Aircraft System Traffic Management (UTM) Concept of Operations. In *16th AIAA Aviation Technology, Integration, and Operations Conference*, Washington, D.C., USA. AIAA.

[125] Pritchett, A. R. and Genton, A. (2018). Negotiated Decentralized Aircraft Conflict Resolution. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):81–91.

[126] Raddalgoda, M. (2020). How can we deploy drones in healthcare to save lives? Ericsson. URL.

[127] Ramchurn, S. D., Farinelli, A., MacArthur, K. S., and Jennings, N. R. (2009). Decentralized coordination in RoboCup Rescue. *The Computer Journal*, 53(9):1447–1461.

[128] Ramchurn, S. D., Polukarov, M., Farinelli, A., Truong, C., and Jennings, N. R. (2010). Coalition Formation with Spatial and Temporal Constraints. In *9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1181–1188, Toronto, Canada. IFAAMAS.

[129] Raptopoulos, A. (2013). No roads? There's a drone for that (TED Talk). TED. URL.

[130] Rasmusen, E. (1989). *Games and Information*. Blackwell, Cambridge.

[131] Rattanagraikanakorn, B., Gransden, D. I., Schuurman, M., De Wagter, C., Happee, R., Sharpanskykh, A., and Blom, H. A. P. (2019). Multibody system modelling of unmanned aircraft system collisions with the human head. *International Journal of Crashworthiness*, pages 1–19.

[132] Reeves, D. M., Wellman, M. P., MacKie-Mason, J. K., and Osepayshvili, A. (2005). Exploring bidding strategies for market-based scheduling. *Decision Support Systems*, 39(1):67–85.

[133] Rehkopf, T. (2019). DHL launches its first regular fully-automated and intelligent urban drone delivery service. Deutsche Post DHL Group. URL.

[134] Richter, C., Bry, A., and Roy, N. (2016). Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments. In Inaba, M. and Corke, P., editors, *Robotics Research. Springer Tracts in Advanced Robotics*. Springer, 114 edition.

[135] Rios, J. (2018). National Aeronautics and Space Administration UAS Traffic Management (UTM) Project Strategic Deconfliction: System Requirements Final Report. Technical report, NASA.

[136] Röger, G. and Helmert, M. (2010). The More, the Merrier: Combining Heuristic Estimators for Satisficing Planning. In *Twentieth International Conference on Automated Planning and Scheduling (ICAPS)*, pages 246–249, Toronto, Canada. AAAI.

[137] Roldan, J. J., del Cerro, J., and Barrientos, A. (2018). Should we Compete or Should we Cooperate? Applying Game Theory to Task Allocation in Drone Swarms. In *International Conference on Intelligent Robots and Systems : Towards a Robotic Society*, pages 5366–5371, Madrid. IEEE/RSJ.

[138] Roldán, J. J., Lansac, B., del Cerro, J., and Barrientos, A. (2016). A proposal of multi-UAV mission coordination and control architecture. In *Advances in Intelligent Systems and Computing*, volume 417, pages 597–608. Springer Verlag.

[139] Rossi, C., Aldama, L., and Barrientos, A. (2015). Simultaneous task subdivision and allocation using negotiations in multi-robot systems. *International Journal of Advanced Robotic Systems*, 12.

[140] Rubinstein, A. (1982). Perfect Equilibrium in a Bargaining Model. *Econometrica*, 50(1):97.

[141] Ryan, M. R. K. (2008). Exploiting Subgraph Structure in Multi-Robot Path Planning. *Journal of Artificial Intelligence Research*, 31:497–542.

[142] Sajid, Q., Luna, R., and Bekris, K. E. (2012). Multi-Agent Pathfinding with Simultaneous Execution of Single-Agent Primitives. In *Proceedings of the Fifth Annual Symposium on Combinatorial Search (SOCS)*, pages 88–96, Niagara Falls, Ontario, Canada. AAAI.

[143] Santana, P., Vaquero, T., Toledo, C., Wang, A., Fang, C., and Williams, B. (2016). PARIS: a Polynomial-Time, Risk-Sensitive Scheduling Algorithm for Probabilistic Simple Temporal Networks with Uncertainty. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

[144] Schelling, T. C. (1956). An Essay on Bargaining. *The American Economic Review*, 46(3):281–306.

[145] Schermer, D., Moeini, M., and Wendt, O. (2018). Algorithms for Solving the Vehicle Routing Problem with Drones. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10751 LNAI, pages 352–361. Springer Verlag.

[146] Schneider, O., Kern, S., Knabe, F., Gerdes, I., Delahaye, D., Vidosavljevic, A., van Leeuwen, P., Niewenhuisen, D., Sunil, E., Hoekstra, J., and Ellerbroek, J. (2014). METROPOLIS - Urban Airspace Design - Concept Design. Technical report, TU Delft, DLR, NLR, ENAC.

[147] Shaked, A. and Sutton, J. (1984). Involuntary unemployment as a perfect equilibrium in a bargaining model. *Econometrica*, 52:1351–1364.

[148] Sharon, G., Stern, R., Felner, A., and Sturtevant, N. (2012). Meta-agent Conflict-Based Search For Optimal Multi-Agent Path Finding. In *Proceedings of the Fifth Annual Symposium on Combinatorial Search (SoCS)*, Niagara Falls, Ontario, Canada. AAAI.

[149] Sharon, G., Stern, R., Felner, A., and Sturtevant, N. R. (2015). Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66.

[150] Sharon, G., Stern, R., Goldenberg, M., and Felner, A. (2013). The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*, 195:470–495.

[151] Shehory, O. and Kraus, S. (1998). Artificial Intelligence Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101:165–200.

[152] Shoham, Y. and Leyton-Brown, K. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. URL.

[153] Silver, D. (2005). Cooperative Pathfinding. In *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 117–122, Marina del Rey, California, USA. AAAI.

[154] Skiena, S. S. (2008). *The Algorithm Design Manual*. Springer, second edition.

[155] Smith, R. G. (1980). The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, 29(12):1104–113.

[156] Srinivasan, A., Kam, T., Malik, S., and Brayton, R. K. (1990). Algorithms for Discrete Function Manipulation. In *1990 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*, pages 92–95. IEEE.

[157] Srour, F. J., Agatz, N., and Oppen, J. (2018). Strategies for handling temporal uncertainty in pickup and delivery problems with time windows. *Transportation Science*, 52(1):3–19.

[158] Stahl, I. (1972). Bargaining Theory. Technical report, Economics Research Institute, Stockholm School of Economics, Stockholm.

[159] Standley, T. (2010). Finding Optimal Solutions to Cooperative Pathfinding Problems. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 173–178, Atlanta, Georgia, USA. AAAI.

[160] Standley, T. and Korf, R. (2011). Complete Algorithms for Cooperative Pathfinding Problems. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 668–673, Barcelona, Catalonia, Spain.

[161] Sturtevant, N. R. (2012). Benchmarks for grid-based pathfinding. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):144–148.

[162] Sun, D. and Wu, J. (2009). Multi-agent coordination based on contract net protocol. In *International Symposium on Intelligent Ubiquitous Computing and Education, IUCE*, pages 353–357.

[163] Sunil, E., Hoekstra, J., Ellerbroek, J., Bussink, F., Nieuwenhuisen, D., Vidosavljevic, A., and Kern, S. (2015). Metropolis: Relating Airspace Structure and Capacity for Extreme Traffic Densities. In *Eleventh USA/Europe Air Traffic Management Research and Development Seminar*, Lisbon, Portugal.

[164] Surynek, P. (2009a). A novel approach to path planning for multiple robots in bi-connected graphs. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3613–3619.

[165] Surynek, P. (2009b). Towards Shorter Solutions for Problems of Path Planning for Multiple Robots in T-like Environments. In *Proceedings of the Twenty-Second International Florida Artificial Intelligence Research Society (FLAIRS) Conference*, pages 207–212, Sanibel Island, Florida, USA.

[166] Surynek, P. (2012a). *On Propositional Encodings of Cooperative Path-Finding*, volume 1.

[167] Surynek, P. (2012b). Towards Optimal Cooperative Path Planning in Hard Setups through Satisfiability Solving. In Anthony, P., Ishizuka, M., and Lukose, D., editors, *Proceedings of the 12th Pacific Rim international conference on Trends in Artificial Intelligence*, pages 564–576. Springer-Verlag.

[168] Surynek, P., Felner, A., Stern, R., and Boyarski, E. (2016a). An Empirical Comparison of the Hardness of Multi-Agent Path Finding under the Makespan and the Sum of Costs Objectives. In *Ninth Annual Symposium on Combinatorial Search (SoCS)*, New York, USA. AAAI.

[169] Surynek, P., Felner, A., Stern, R., and Boyarski, E. (2016b). Efficient SAT Approach to Multi-Agent Path Finding under the Sum of Costs Objective. In *22nd European Conference on Artificial Intelligence (ECAI)*, The Hague, Netherlands.

[170] Sutton, J. (1986). Non-Cooperative Bargaining Theory: An Introduction. *The Review of Economic Studies*, 53(5):709–724.

[171] Sutton, R. S. and Barto, A. G. (2015). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachussetts, 2 edition.

[172] Thayer, J. T., Dionne, A., and Ruml, W. (2011). Learning Inadmissible Heuristics During Search. In *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling*, pages 250–257, Freiburg, Germany.

[173] Tovey, C., Lagoudakis, M. G., Jain, S., and Koenig, S. (2005). The Generation of Bidding Rules for Auction-Based Robot Coordination. In Parker, L., Schneider, F., and Schultz, A., editors, *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, pages 3–14. Springer, Dordrecht.

[174] Troudi, A., Addouche, S. A., Dellagi, S., and El Mhamedi, A. (2018). Sizing of the drone delivery fleet considering energy autonomy. *Sustainability (Switzerland)*, 10(9).

[175] Udluft, H. (2017). *Decentralization in Air Transportation*. PhD thesis, TU Delft.

[176] Ulmer, M. W. and Thomas, B. W. (2018). Same-day delivery with heterogeneous fleets of drones and vehicles. *Networks*, 72(4):475–505.

[177] Vente, S., Kimmig, A., Preece, A., and Cerutti, F. (2020). Increasing negotiation performance at the edge of the network. In *7th International Conference on Agreement Technologies (AT). Preprint.*, Thessalonikki, Greece.

[178] Wagner, G. and Choset, H. (2011). M*: A Complete Multirobot Path Planning Algorithm with Performance Bounds. In *Proceedings of (IROS) IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, USA. IEEE.

[179] Wang, D. (2015). The Economics of Drone Delivery. Flexport. URL.

[180] Wang, K.-H. C. and Botea, A. (2008). Fast and Memory-Efficient Multi-Agent Pathfinding. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS)*, pages 380–387, Sydney, Australia. AAAI.

[181] Wang, K.-H. C. and Botea, A. (2011). MAPP: a Scalable Multi-Agent Path Planning Algorithm with Tractability and Completeness Guarantees. *Journal of Artificial Intelligence Research*, 42:55–90.

[182] Whitbrook, A., Meng, Q., and Chung, P. W. (2018). Reliable, Distributed Scheduling and Rescheduling for Time-Critical, Multiagent Systems. *IEEE Transactions on Automation Science and Engineering*, 15(2):732–747.

[183] Whitbrook, A., Meng, Q., and Chung, P. W. (2019). Addressing robustness in time-critical, distributed, task allocation algorithms. *Applied Intelligence*, 49(1).

[184] Wu, M., De Weerdt, M., La Poutré, H., Yadati, C., Zhang, Y., and Witteveen, C. (2010). Multi-player multi-issue negotiation with complete information. *Studies in Computational Intelligence*, 319:147–159.

[185] Yoshizumi, T., Miura, T., and Ishida, T. (2000). A* with Partial Expansion for large branching factor problems. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, page 923–929, Austin, Texas, USA. AAAI.

[186] Yu, J. and LaValle, S. M. (2013). Planning Optimal Paths for Multiple Robots on Graphs. In *International Conference on Robotics and Automation (ICRA)*, pages 3612–3617, Karlsruhe, Germany. IEEE.

[187] Yu, J. and Lavalle, S. M. (2013). Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 1443–1449, Bellevue, Washington, USA. AAAI.

[188] Zeng, Y., Wu, Q., and Zhang, R. (2019). Accessing from the Sky: A Tutorial on UAV Communications for 5G and beyond. In *Proceedings of the IEEE*, volume 107, pages 2327–2375. IEEE.

[189] Zhang, J., Wang, G., and Song, Y. (2019). Task assignment of the improved contract net protocol under a multi-agent system. *Algorithms*, 12(4).

[190] Zheng, X. and Koenig, S. (2009). K-Swaps: Cooperative Negotiation for Solving Task-Allocation Problems. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pages 373–378, Pasadena, California, USA. AAAI.

[191] Zheng, X., Koenig, S., and Tovey, C. (2006). Improving Sequential Single-Item Auctions. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2238–2244, Beijing, China. IEEE.