Proceedings of the 35th European Safety and Reliability & the 33rd Society for Risk Analysis Europe Conference
Edited by Eirik Bjorheim Abrahamsen, Terje Aven, Frederic Bouder, Roger Flage, Marja Ylönen
 Published by Research Publishing, Singapore.
doi: 10.3850/978-981-94-3281-3_ESREL-SRA-E2025-P1412-cd

Reliability of Information Gathering under Dynamic and Stochastic Utility

Sridhar Radhakrishnan
School of Computer Science, University of Oklahoma. E-mail: sridhar@ou.edu

Kash Barker
School of Industrial and Systems Engineering, University of Oklahoma. E-mail: kashbarker@ou.edu

Scheduling tasks across multiple resources is a critical challenge in many domains, including the process of information gathering via drones. The complexity increases when tasks (i.e., reconnaissance operations) have dynamic utility functions that represent the value or benefit provided by completing the task at any given time. These utility functions can change over time due to external factors, and the goal is to maximize the total or expected utility while minimizing the costs associated with switching between tasks. The stochastic job scheduling problem addresses scenarios in which tasks are dynamically assigned to resources with utility values that may be probabilistic. Each task's utility varies over time, and switching tasks incurs costs such as time, energy, or other penalties, adding complexity to optimizing resource schedules. We demonstrate that the deterministic version (utility values are fixed) of the problem, involving a single information gathering agent (i.e., drone) and multiple monitoring regions, is NP-Hard. To address this, we develop an integer linear programming (ILP) model with constraints that ensure feasible solutions. By introducing uncertainty and a decay function, we make the utility values stochastic and dynamic and use the ILP model to solve the problem effectively. Additionally, we propose an iterative algorithm to determine a nonzero switching cost matrix that maximizes total utility, providing a practical approach to optimizing resource scheduling in dynamic environments.

Keywords: Defense, Security risks, Socio-technical systems, Crisis management, Disaster management

1. Introduction

In real-world systems, scheduling tasks across multiple resources is a critical problem in various domains. The scheduling process becomes even more complex when we account for tasks with dynamic utility functions—representing the value or utility of completing a task at any given time—and allow for preemptive switching between tasks. These utility functions may vary over time due to external factors, and the system's goal is to maximize the total or expected utility while minimizing the overhead incurred by switching between tasks.

In this context, the stochastic job scheduling problem models scenarios where tasks can be dynamically assigned to resources, with utility values that may be probabilistic or subject to external influences. Each task has a utility function that varies over time, and switching between tasks or resources incurs a cost (e.g., time, energy, or some other penalty). This leads to the challenge of optimally scheduling tasks across available resources, accounting for utility fluctuations, and minimizing the negative impact of switching.

This problem has many practical applications across various domains, where optimizing resource utilization and maximizing overall utility or performance are crucial. From drones performing reconnaissance in disaster-affected areas to autonomous vehicle fleets delivering goods, the general principles of this scheduling problem can be adapted to numerous use cases. Each application dynamically allocates resources to tasks (or jobs) to maximize utility, minimize downtime, and handle the costs of switching between tasks.

This paper addresses the challenge of scheduling drones for reconnaissance missions in disaster areas with the aim of saving as many people as possible. In this context, each task represents a specific location that a drone must visit. The effectiveness of visiting a location is determined by a utility function that reflects the number of people rescued at a point in time. These utility values are not constant; they fluctuate dynamically due

to factors such as evacuations, changing weather conditions, or accessibility challenges. This variability introduces uncertainty and complexity into the scheduling process, requiring drones to adapt to real-time changes.

Note that the utility functions that change over time are assumed to be known *a priori* in the drone application, as it is assumed that they can be estimated using predictive models and historical data from similar disaster scenarios. For example, population movement patterns, expected evacuation rates, and accessibility challenges in different regions can often be modeled based on prior knowledge. In addition, geographic and environmental data can help anticipate how the utility of visiting certain locations evolves over time.

Drones serve as resources in this problem. Each drone can visit only one location at a time, but can move from one location to the other. In addition, multiple drones can operate simultaneously, each covering a different location. The operation time for each drone is limited and these time limits can vary between drones. Therefore, the scheduling process must ensure that the available time for each drone is used efficiently.

A key challenge in this problem is the presence of switching costs. When a drone moves from one location (e.g., geographical search area) to another, it incurs a cost that could represent the time or energy spent during the transition. These switching costs reduce the time available for actual reconnaissance and must be taken into account in the schedule to maximize overall effectiveness.

Another layer of complexity comes from the stochastic nature of utility functions. The utility of each location is uncertain and is drawn from a probability distribution. This means that the number of people a drone can rescue at a given location varies, making it necessary to balance the expected benefits of visiting a location against the associated switching costs and uncertainty.

The ultimate objective is to assign drones to locations in a way that maximizes the total expected number of people rescued within the available time. This involves carefully planning the schedule to make the best use of each drone's time, minimize switching costs, and account for the unpredictable nature of rescue outcomes at each location. In essence, the problem is about optimizing drone missions to save as many lives as possible despite time constraints, travel costs, and uncertainty.

1.1. *Related Work*

In this review of previous work, we focus on research involving (i) switching costs and the scheduling of resources to tasks and (ii) drone scheduling. Ho and Sum (2017) described a scheduling problem that incorporates switching costs. They highlight that makespan, total completion time, and due date assignment problems become NP-Hard for specific cases of switching costs. However, they also demonstrate that the problem can be solved in polynomial time for certain scenarios.

Fu et al. (2024) discuss the multitasking scheduling problem with the objectives of minimizing the makespan and the total completion time. They explore a parallel machine environment where "work stealing" is used to address tasks not initially allocated.

Bender et al. (2013) introduced a framework known as reallocation problems, where a collection of unit-length tasks must be processed on m machines. Each task has a specific time window, defined by an arrival time and a deadline, during which it must be completed. Tasks are dynamically added and removed, and the main objective is to maintain a feasible schedule at all times. Farach-Colton et al. (2021) presented more recent advances in reallocation problems.

Letsios et al. (2021) considered scheduling under uncertainty, where machines may fail, work may be canceled, or other unpredictable events may occur. Their goal was to minimize the makespan while accounting for these uncertainties. Similarly, Song and Leus (2022) addressed the problem of scheduling parallel machines under uncertainty, focusing on robust solutions.

Recent work also includes research on drone delivery systems and associated scheduling challenges. Attenni et al. (2023) provided a survey on scheduling problems related to urban deliv-

ery services. Pasha et al. (2022) presented another survey that addresses various problems, including general drone scheduling, drone delivery, monitoring, and recharging considerations. Montemanni and Dell'Amico (2023) studied parallel drone scheduling with the objective of minimizing the total time required to serve all customers.

1.2. *General Problem Definition*

The *stochastic job scheduling problem* involves scheduling a set of tasks across one or more resources over a finite period of time. Each task is defined by a *utility function* that represents the value or performance of the task as it progresses over time. The objective is to maximize the *total expected utility* obtained from all tasks while accounting for the costs associated with switching between tasks.

In this context, the key elements of the problem are as follows.

- **Tasks**: A set of tasks $J = \{j_1, j_2, \ldots, j_n\}$, each associated with a utility function $u_i(t)$. The utility function defines the value or utility that the tasks provides at each point in time t.
- **Time Horizon for Each Resource**: Each resource r_i has a specific time horizon D_i, which represents the total time available for scheduling tasks on resource r_i. This suggests that the time available for scheduling tasks on each resource need not be uniform.
- **Resources**: Let R be the set of resources $R = \{r_1, r_2, \ldots, r_k\}$, where each resource can handle only one task at any given time. Multiple tasks can run in parallel on different resources.
- **Switching Costs**: A switching cost d_{ij} applies when switching from task j_i to task j_j on a given resource. This cost is typically time-based, but involves other factors, such as energy or overhead.
- **Stochastic Utility Function**: The utility function $u_i(t)$ for each task defines the value of completing task j_i at time t. In the stochastic case, $u_i(t)$ is drawn from a probability distribution, $u_i(t) \sim \mathcal{P}_i(t)$, where $\mathcal{P}_i(t)$ represents the distribution governing the utility of task j_i at time t.
- **Objective**: The goal is to find a schedule $S(t)$ that assigns tasks to resources over time so that total (expected) utility is maximized. The schedule must also account for switching costs between tasks.

Problem Formulation:

$$\text{Maximize } \mathbb{E}\left[\sum_{r\in R}\sum_{t=0}^{D_r-\sum_{(i,j)\in S_r} d_{ij}} u_{S_r(t)}(t)\right]$$

where:

- $R = \{r_1, r_2, \ldots, r_k\}$ is the set of resources,
- D_r is the time horizon for resource r,
- $S_r(t)$ represents the task scheduled at time t on resource r,
- $u_{S_r(t)}(t)$ is the stochastic utility provided by the task assigned to resource r at time t, drawn from $\mathcal{P}_{S_r(t)}(t)$,
- d_{ij} is the switching cost incurred when switching from task j_i to task j_j on resource r,
- $\mathbb{E}[\cdot]$ denotes the expectation operator, reflecting the stochastic nature of the utility.

The formulation ensures that the expected total utility is maximized over the adjusted time horizon D_r, which accounts for the cumulative switching costs $\sum_{(i,j)\in S_r} d_{ij}$. Only one task can be scheduled on a resource at any given time, and the randomness in the utility functions is addressed through the expectation operator.

1.3. *Differences from Traditional Job Scheduling or Multiprocessor Scheduling*

The scheduling problems discussed here differ significantly from traditional job scheduling or multiprocessor scheduling in several key ways.

First, the proposed problems incorporate stochastic utility functions, where the utility of a task at any given time is uncertain and derived from a probability distribution. Furthermore, we have incorporated a utility decaying function over time to account for the dynamic utility values. This contrasts with traditional scheduling, where tasks have fixed execution times or defined priorities. The probabilistic nature of utility adds com-

plexity, as scheduling decisions must account for the varying and uncertain benefits of scheduling tasks over time.

Second, the proposed problems introduce switching costs, which are often negligible in traditional scheduling. Here, transitioning from one task to another incurs penalties, such as time delays or energy consumption, as switching implies changing search regions in the information gathering context. These costs make frequent switching undesirable and force schedulers to balance search task transitions carefully to optimize overall performance.

Third, the objective differs fundamentally. Instead of minimizing time-based metrics, such as makespan or waiting time, the goal here is to maximize the expected utility of scheduling tasks over a fixed period. This shift in focus demands more sophisticated scheduling strategies that prioritize long-term benefits over traditional efficiency metrics.

2. Deterministic Drone Scheduling: Single-Drone, Multiple-Task Case

In this section, we will take a simple version of the problem in which a single drone (resource) is available and we are given a number of locations to perform reconnaissance (tasks). The goal is to determine the optimal schedule for a single drone. At each time step t, the drone is assigned to a particular location so that the overall utility is maximized, taking into account the utility functions and the cost of switching between locations.

More formally, the utility function for each task is deterministic, which means that the utility provided by each task at any given time is known and does not vary. The problem considers scheduling tasks on a *single resource* over a fixed time horizon. The objective is to maximize total utility over the adjusted time horizon while accounting for the switching costs incurred between tasks.

- **Tasks**: A set of tasks $J = \{j_1, j_2, \ldots, j_n\}$, each associated with a deterministic utility function $u_i(t)$, which defines the utility provided by task j_i at time t.
- **Resource**: A single resource r with a fixed time horizon D, during which the tasks must be scheduled. The resource can execute only one task at a time.
- **Switching Costs**: A switching cost d_{ij} is incurred when switching from task j_i to task j_j. The cumulative switching cost reduces the total available time for task execution on the resource.
- **Objective**: Maximize the total utility over the adjusted time horizon D for the single resource, accounting for the time lost due to switching costs.

Problem Formulation:

$$\text{Maximize} \quad \sum_{t=0}^{D-\sum_{(i,j)\in S} d_{ij}} u_{S(t)}(t)$$

where:

- $S(t)$ is the task scheduled at time t on the single resource r,
- $u_{S(t)}(t)$ is the deterministic utility provided by the scheduled task at time t,
- D is the total time horizon for the single resource, adjusted for switching costs $\sum_{(i,j)\in S} d_{ij}$,
- d_{ij} is the switching cost incurred when switching from task j_i to task j_j,
- S is the set of all scheduled task transitions during the horizon D.

2.1. *NP-Hardness Proof*

In this subsection, we show that the deterministic job scheduling for a single drone with multiple tasks is NP-Hard by reducing the Knapsack problem instance to our problem. Note that the 0/1 knapsack problem involves selecting items to maximize total utility while remaining within the restricted fixed capacity. Each item i is characterized by two attributes: a value, v_i, representing the utility or benefit of choosing the item, and a weight, w_i, representing resource consumption. The problem is constrained by a total capacity W, which represents the maximum available resource (e.g., time). The objective is to choose a subset of items to maximize the total value without exceeding the capacity.

The items in the knapsack map directly to the tasks or locations j_i that the drone must visit. The value of each knapsack item v_i corresponds to the utility $u_i(t)$ of being at the location j_i for a specific time t. For simplicity, the utility can be assumed to be constant over the time spent at a location. The weight w_i of an item translates to the amount of time the drone spends at a location. Furthermore, switching between locations introduces a fixed cost d_{ij}, which represents the time or energy consumed when moving from location j_i to j_j. This switching cost is not part of the standard Knapsack problem, but adds an important layer of complexity to the scheduling problem. The total capacity W of the knapsack is mapped to the total operational time of the drone, including both the time spent at the locations and the time spent transitioning between them.

Given a time horizon D, the scheduling problem can be framed as follows: the drone must visit a subset of locations j_i, where each selected location has an associated time t_i that the drone spends there. The utility earned at a location is proportional to $u_i(t) \cdot t_i$, combining the utility rate and the duration of the visit. The transition between locations j_i and j_j incurs a switching cost d_{ij}, which could be measured in terms of time, energy, or other relevant metrics. The total time spent in locations and transitions must not exceed the available time horizon D. The goal is to determine an optimal schedule that maximizes total utility while respecting time constraints and accounting for switching costs.

To demonstrate that the deterministic task scheduling problem is NP-hard, we reduce the knapsack problem to this scheduling problem. The reduction is as follows:

Each item in the knapsack instance is assigned to a task j_i in the scheduling problem. The utility v_i of the item corresponds to the utility $u_i(t)$ of the task at a location for a specific time t. Similarly, the weight w_i of the item translates to the time t_i the drone spends at the location j_i. To align this reduction with the pure knapsack setup, we introduce a trivial switching cost $d_{ij} = 0$, ensuring that transitions between tasks do not incur any additional penalties in this simplified setup.

The job scheduling problem is then solved by selecting a subset of locations j_i that maximizes the total utility $\sum u_i(t) \cdot t_i$, subject to the constraint $\sum t_i \leq T$, where T is the total available time. This setup mirrors the objective of the Knapsack problem: selecting a subset of items that maximizes total value without exceeding the capacity.

The solution to the job scheduling problem directly corresponds to a solution to the Knapsack problem. The selected tasks in the scheduling problem are mapped to the selected items in the knapsack problem, and the total utility achieved matches the total value in the Knapsack solution.

In our formulation, we have included non-zero switching costs d_{ij}, and the scheduling problem becomes a generalization of the knapsack problem. This added complexity, while more representative of real-world scenarios, does not change the fundamental computational hardness of the problem. Since the Knapsack problem is NP-hard, the deterministic job scheduling problem is also NP-hard because of this polynomial-time reduction.

2.2. *Integer Linear Programming Formulation*

The ILP formulation of the deterministic task scheduling problem with preemptive scheduling on a single source has the following objective function to be maximized:

$$\text{Maximize} \sum_{i=1}^{n}\sum_{t=1}^{D} u_i(t)\cdot x_{i,t} - \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=2}^{D} d_{ij}\cdot z_{i,j,t}$$

Recall the following, along with additional information on the binary variables $x_{i,t}$ and $z_{i,j,t}$.

- $J = \{j_1, j_2, \ldots, j_n\}$: Set of tasks.
- $u_i(t)$: Deterministic utility provided by the task j_i at time t.
- D: Total time horizon for a single resource.
- d_{ij}: Switching cost incurred when transitioning from task j_i to task j_j.
- $x_{i,t} \in \{0, 1\}$: Binary variable indicating whether task j_i is scheduled at time t.
- $z_{i,j,t} \in \{0, 1\}$: Binary variable indicating whether the resource switches from task j_i to task j_j at time t.

We formulate the set of constraints that address when (i) only one task is executed at each time step, (ii) we switch tasks, then track and penalize the total time available by the switching cost d_{ij}, (iii) a task does not switch itself and if it does, we do not penalize that switching cost, and (iv) the total available time does not exceed D. Given this, our constraints are formulated as follows:

- **Single Task per Time Step**:

$$\sum_{i=1}^{n} x_{i,t} = 1, \quad \forall t \in 1, \ldots, D.$$

- **Switching Indicator**: The switching indicator $z_{i,j,t}$ is designed to trigger only when $x_{i,t-1} = 1$ (i.e., the task was active at $t-1$) and $x_{j,t} = 1$ (i.e., a new task is active at t). If a switch occurs, $z_{i,j,t} = 1$; otherwise, $z_{i,j,t} = 0$. This is given as $z_{i,j,t} \geq x_{i,t-1} + x_{j,t} - 1, \quad \forall i, j \in J,\ t \in \{2, \ldots, D\}$.
- **Time Horizon Adjusted for Switching Costs**: Note that we start with $t = 2$ below, as the switch does not occur at time $t = 1$. The first part of the equation addresses the amount of time spent doing some task, and the second part addresses the time spent switching. The total time D is fully utilized by performing a task with $x_{i,t} = 1$ or switching with $z_{i,i,t} = 1$.

$$\sum_{t=1}^{D} \sum_{i=1}^{n} x_{i,t} + \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{t=2}^{D} d_{ij} \cdot z_{i,j,t} \leq D.$$

- **Additional Notes**: To explicitly incorporate shifting constraints, two additional conditions can be introduced. The delayed start constraint ensures that if a switch occurs at time t, task j cannot begin until the switching delay $t + d_{ij}$ is completed. This is represented as $x_{j,t+d_{ij}} \leq z_{i,j,t}, \quad \forall i, j \in J,\ t + d_{ij} \leq D$.

  Next we account for any utility contribution from task j during the switching delay period. This is enforced by ensuring $x_{j,k} = 0, \quad \forall k \in [t, t + d_{ij} - 1],\ \forall i, j \in J,\ t + d_{ij} \leq D$.

  Instead of explicitly setting $x_{j,k} = 0$ for all $k \in [t, t + d_{ij} - 1]$, which may result in infeasibility, the delayed start constraint can be used indirectly. Here the utility contribution from j is delayed until $t + d_{ij}$. Since $x_{j,t+d_{ij}}$ depends on $z_{i,j,t}$, the solver avoids assigning utility during the switching period as long as this condition is strictly enforced. This approach is simpler and avoids adding explicit constraints that may conflict with other scheduling conditions, leading to infeasibility.

2.3. *Working Example*

We consider three tasks $n = 3$ and $D = 20$ which is our time horizon. Consider the following utility functions for each of the three tasks.

The utility functions for each task over time are as follows:

- **Task 1**: $\{10, 8, 6, 4, 2, 1, 5, 7, 9, 11, 13, 10, 8, 6, 4, 2, 1, 5, 7, 9\}$
- **Task 2**: $\{5, 7, 9, 11, 13, 12, 11, 10, 9, 8, 7, 9, 11, 13, 15, 10, 5, 3, 6, 8\}$
- **Task 3**: $\{3, 4, 5, 6, 7, 6, 10, 12, 14, 13, 11, 9, 8, 7, 6, 5, 9, 11, 12, 8\}$

The switching costs between tasks are as follows: $d = [0, 2, 3; 2, 0, 1; 3, 1, 0]$. The Gurobi optimizer provided the optimal solution is less than 0.01 seconds on a CPU model: Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz with 8 physical cores, 16 logical processors, using up to 16 threads. The value of the optimal solution was 162. Fig. 1. provides an illustration of the tasks that are executed along with the when a switch is performed.

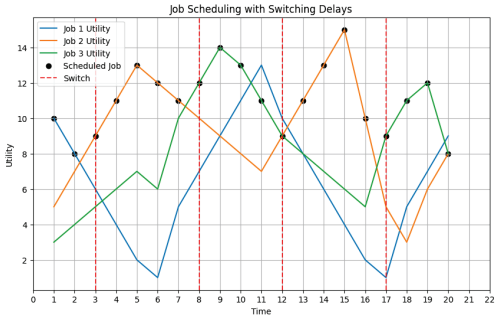

Fig. 1. Scheduling and utility functions over time with switching costs.

3. Stochastic and Dynamic Drone Scheduling: Single-Drone, Multiple-Task Case

In this section, we extend the deterministic case in Section 2 to the stochastic case. A single drone (resource) is still available, and the goal remains to determine the optimal schedule for reconnaissance at multiple locations (tasks). However, unlike the deterministic case, the utility functions are stochastic, meaning that the utility provided by each task at a given time is not fixed, but instead drawn from a probability distribution. This introduces uncertainty into the problem, requiring decisions to be based on the expected utility.

Differences from the Deterministic Case:

- **Utility Function**: Each task j_i is associated with a stochastic utility function $u_i(t)$, where $u_i(t)$ is drawn from a probability distribution $u_i(t) \sim \mathcal{P}_i(t)$, where $\mathcal{P}_i(t)$ defines the distribution governing the utility of task j_i at time t. The expectation of this utility is used in the objective function.
- **Objective**: The goal is to maximize the *expected total utility* over the adjusted time horizon D, adjusting for uncertainty in utility values and switching costs between tasks.
- **Problem Formulation**:

$$\text{Maximize } \mathbb{E}\left[\sum_{t=0}^{D-\sum_{(i,j)\in S} d_{ij}} u_{S(t)}(t)\right],$$

where $\mathbb{E}[\cdot]$ denotes the expectation operator.

The stochastic nature of the utility functions introduces additional complexity into the scheduling problem. Decisions must be made under uncertainty, balancing expected utility gains against penalties for switching between tasks.

The ILP formulation is provided below for the above problem.

$$\text{Maximize } \mathbb{E}\left[\sum_{t=1}^{D}\sum_{i=1}^{n} u_i(t)x_{i,t} - \sum_{t=2}^{D}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}z_{i,j,t}\right]$$

The utility $u_i(t)$ is modeled as a random variable, and its expected value $\mathbb{E}[u_i(t)]$ is used in optimization:

$$\mathbb{E}[u_i(t)] = \int u_i(t) \cdot f_i(t)\,\mathrm{d}u$$

where $f_i(t)$ is the probability density function of $\mathcal{P}_i(t)$. Note that for normal distribution $\mathbb{E}[u_i(t)] = u_i(t)$. We use Monte Carlo sampling because our distribution could be complex or unknown. In the working example given in Section 2.3, the expected utilities for each task and time step were calculated using Monte Carlo sampling. A specified number of samples (1,000) were generated from a normal distribution with the given mean (μ, where each initial value in Section 2.3 was treated as the mean) and standard deviation ($\sigma = \{2, 1.5, 1\}$ for the three tasks, respectively). The expected utility at each time step was then estimated as the mean of these samples.

These expected values were used in the ILP model to obtain the optimal schedule. For the working example in Section 2.3, and with the above, we see that our optimizer produces an optimal value of 107.

Fig. 2. provides an illustration of the tasks that are executed along with the when a switch is performed for the stochastic case.

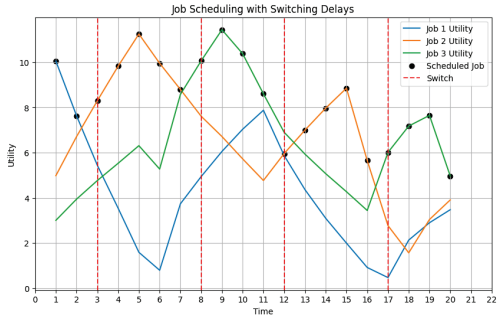

Fig. 2. Scheduling and utility functions over time with switching costs for the stochastic utility case with a decaying function.

4. Sensitivity Analysis of Switching Cost and Optimal Model

In the working example in Section 2.3, setting all switching costs to zero resulted in a total maximum utility of 226, compared to 162 when non-

zero switching costs were applied. This observation raises the question of identifying specific non-zero switching costs that maximize the total utility.

In this paper, our objective is to determine the switching cost matrix d using an iterative approach. We begin with a randomly initialized matrix d and solve the scheduling problem using the ILP model. Once a solution is obtained, we adjust the values in the matrix d based on the results of the scheduling problem to improve the total utility in general. Specifically, in gradient-based optimization, the values of d_{ij} are updated iteratively by estimating the gradient of the total utility with respect to d_{ij}. This method allows the algorithm to move in the direction that increases utility, refines the switching costs step by step for better performance (**Algorithm 1**).

Algorithm 1 Iterative/Heuristic Approach for Optimizing Switching Costs

Input: Utility values $u_i(t)$, time horizon D, initial switching costs $d_{ij}^{(0)}$, tolerance ϵ, maximum iterations K
Output: Optimized switching costs d_{ij}^{*}, optimal schedule S^{*}, total utility U^{*}
Initialize $d_{ij}^{(0)} \leftarrow \text{Uniform}(0, d_{\max})$
Set iteration counter $k \leftarrow 0$
Set convergence flag $\Delta U \leftarrow \infty$
$U^{*} \leftarrow 0$
while $\Delta U > \epsilon$ and $k < K$ **do**
    Solve the scheduling problem using $d_{ij}^{(k)}$
    Compute the optimal schedule $S^{(k)}$ and total utility $U^{(k)}$
    Update switching costs $d_{ij}^{(k+1)}$ based on $U^{(k)}$, $d_{ij}^{(k+1)} \leftarrow max(d_{min}, d_{ij}^{(k)} + \eta \cdot \frac{\partial U(d_{ij}^{(k)})}{\partial d_{ij}})$
    Update convergence flag: $\Delta U \leftarrow |U^{(k+1)} - U^{(k)}|$
    Increment iteration counter $k \leftarrow k + 1$
end while
Return: Optimized switching costs d_{ij}^{*}, schedule S^{*}, and total utility U^{*}

5. Conclusion

We have presented a single-drone multiple-task scheduling problem with switching costs, aimed at maximizing total utility. Our approach incorporates an expected utility maximization formulation that accounts for uncertainties and time-dependent decaying effects on utility, reflecting realistic operational scenarios.

References

Attenni, G., V. Arrigoni, N. Bartolini, and G. Maselli (2023). Drone-based delivery systems: A survey on route planning. *IEEE Access 11*, 123476–123504.

Bender, M. A., M. Farach-Colton, S. Fekete, J. T. Fineman, and S. Gilbert (2013). Reallocation problems in scheduling. In *Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures*, pp. 271–279.

Farach-Colton, M., K. Leal, M. A. Mosteiro, and C. T. Caro (2021). Dynamic windows scheduling with reallocation. *Journal of Experimental Algorithmics (JEA) 26*, 1–19.

Fu, B., Y. Huo, and H. Zhao (2024). Multitasking scheduling with shared processing. *Naval Research Logistics (NRL) 71*(4), 595–608.

Ho, K. I.-J. and J. Sum (2017). Scheduling jobs with multitasking and asymmetric switching costs. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2927–2932.

Letsios, D., M. Mistry, and R. Misener (2021). Exact lexicographic scheduling and approximate rescheduling. *European Journal of Operational Research 290*(2), 469–478.

Montemanni, R. and M. Dell'Amico (2023). Solving the parallel drone scheduling traveling salesman problem via constraint programming. *Algorithms 16*(1), 40.

Pasha, J., Z. Elmi, S. Purkayastha, A. M. Fathollahi-Fard, Y.-E. Ge, Y.-Y. Lau, and M. A. Dulebenets (2022). The drone scheduling problem: A systematic state-of-the-art review. *IEEE Transactions on Intelligent Transportation Systems 23*(9), 14224–14247.

Song, G. and R. Leus (2022). Parallel machine scheduling under uncertainty: Models and exact algorithms. *INFORMS Journal on Computing 34*(6), 3059–3079.