Master thesis in Mathematics-Economics: Schedule Robustness of UAV Operations in Indoor Environments

Mads Kammer Christensen
Master Thesis in Mathematics-Economics
Department of Mathematical Sciences
Aalborg University, Aalborg, Denmark
Supervised by Peter Nielsen

February 5, 2018

Abstract

Producing robust plans for any industrial task has been of import since the dawn of modern industry, where people are dependent on each other for making ends meet. When employing UAVs for various indoor tasks, the need for robustness is not any less. In this paper the effect of introducing time buffers to UAV task plans to increase stability is examined under conditions of varying stochastic influences. A problem configuration with varying stochastic influences is presented, and an algorithm for simulating the resulting delays is presented. The presented algorithm and related planning heuristic allows for the implementation of time buffers, and the effect of these on expected production delay under pre-set stochastic conditions is examined.

1 Introduction

The use of Unmanned Aerial Vehicles (UAVs) is becoming increasingly popular in many modern industrial endeavours, both outdoors and inside production plants. Uses in fields relating to search and rescue missions, crop monitoring at large agricultural expanses, and widespread military applications are some of the outdoor uses for UAV technology (Kristiansen et al., 2012). Indoor applications include, but are not limited to, production supervision, quality inspection, and transport of light production components or tools.

While reasonably accurate positioning for outdoor UAV operations is available through GPS, a number of issues arise when considering using UAVs in indoor environments. Among the sources of uncertainty in indoor environments are: speed of the positioning signal, unforeseen path obstructions, UAV propulsion performance, and disturbances in air currents. In situations with a large number of UAVs operating simultaneously in a confined space, the issue of planning for stochastic flight times is prudent to consider.

Various articles concerning dealing with these issues using an engineering are available, e.g., (Bachrach et al., 2009; Oh et al., 2011). In the absence of sufficient mitigation by technology, publications considering mitigating the uncertainties relating to indoor aerial conveyance by modifying an underlying task plan also exist (Dadkhah and Mettler, 2011). Scheduling UAVs when considering stochastic flight times, gives rise to the question of when to expect delay, and how large of a time buffer is necessary to reduce nervousness to an acceptable level. Inclusion of time buffers is shown to have at mitigating effect on the nervousness of schedules (Elshaer and Yamamoto, 2012). The equilibrium between nervousness of the schedule and improvement of makespan, i.e., the time difference between commencing the first task in a sequence and finishing the last, is one of the major issues to consider when managing many different kinds of production schedules, and becomes ever more relevant in the absence of the ad hoc decision-making capabilities associated with direct human interaction.

The placement of aforementioned equilibrium is largely an arbitrary matter, dependent on external considerations. The more slack allowed for in a UAV production schedule, the more stable it becomes, thus, allowing for more precise planning of adjoining activities, potentially including the hand-off of the finished product.

A mathematical model incorporating the stochastic nature of the conveyance times found in UAV operations, is desirable in order to, with even a slight degree of certainty, control the equilibrium between stability and make-span. This paper concerns the development of such a novel mathematical model, which accounts for the cascade effect of delays throughout a drone network, thereby providing a useful addition to (Khosiawan and Nielsen, 2016).

To estimate the total delay throughout a set of interdependent tasks performed by UAVs it is relevant to consider how these delays stack, i.e., how arriving late at one task propagates through the entire task schedule. To this end we are interested in answering the following problem statement:

How does separate UAV transport delays aggregate throughout a set of tasks, and what is the effect of introducing a mitigating time buffer?

2 Problem definition

We consider a manufacturing environment with U UAVs available to perform T tasks at N different location nodes. Our interest is in determining the influence of stochastic flight times between locations, and how the resulting nervousness in the schedule realisation can be allayed by introducing a fixed time buffer B between tasks. In this paper we consider a manufacturing floor layout which allows for UAV movement between nodes, while considering any delays from intersecting flight paths covered by the stochastic delays. The task times differ for each task, but are considered to be identical among the UAVs in question. Multiple tasks may be located at the same location node, and that node is blocked while a task is in progress. Tasks are subject to precedence constraints defining which tasks must be completed before others as illustrated in Figure 1, allowing for various types of assignments to be considered.

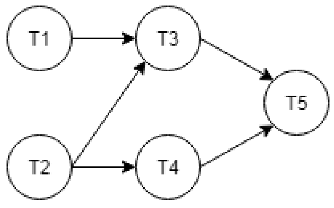

Figure 1: An illustration of the concept of precedence constraints.

The location nodes have a predetermined flight time between them, with stochastic delays being applied to these. In order to allow for a better adaptation of our model to the variations in surroundings inside a production hall, we introduce the concept of *stochasticity zones* to describe different areas of a production hall, with differing stochastic influences. The stochastic delays of the flight between nodes are dependent on which stochasticity zones the UAV is travelling through, Figure 2 provides an example of how different transport delay types $D*$ are applicable between different nodes $N*$, dependent on between which stochasticity zones $Z*$ the UAV is moving.

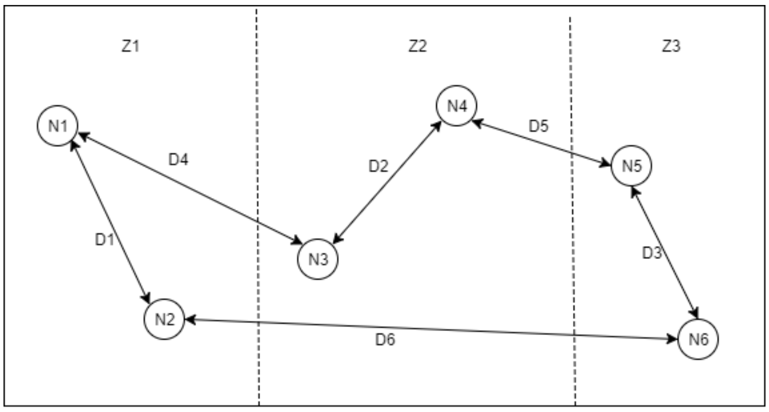

Figure 2: The placement of the task nodes for the small numerical example, including the three delay zones and the corresponding types of delay for flight between nodes.

UAVs are considered identical in terms of flight characteristics and stochastic influences. Battery life considerations are outside the scope of this paper, as initial route planning is assumed to include recharging requirements.

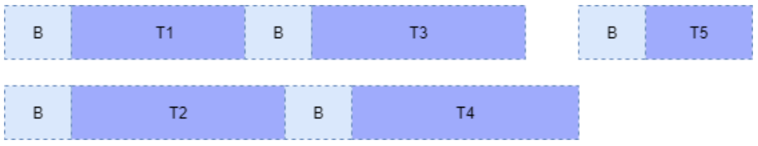

Figure 3: An illustration demonstrating the use of time buffers B placed prior to commencing tasks $T1, \ldots, T5$.

The assumption is that, time buffers can be added to the start time of each task in order to mitigate the negative impact of the unforeseen delays which may otherwise cascade through the task plan. In Figure 3 the concept of time buffers is illustrated by the light blue blocks labelled B being inserted to absorb any late arrival to the location of the following task. The use of time buffers is a well known method to absorb variance in process, or travel times and is widely used in a large variety of planning and scheduling approaches (Jacobs et al., 2011).In this paper we seek a description of the relation between time buffer size and unforeseen schedule timespan increases, in order to make a qualified estimation of the optimal time buffer size.

2.1 Assumptions

To model the problem we rely on a number of assumptions aimed at mimicking actual conditions, while still leaving the problem manageable. Assumptions **A.1**, **A.2**, **A.3**, **A.4**, and **A.5** are similar to those presented in (Khosiawan and Nielsen, 2016), and (Park et al., 2015), while Assumptions **A.6**, **A.7**, **A.8**, and **A.9** are implemented to accommodate the circumstances relating specifically to the problem presented in this paper.

A.1 The tasks cannot be subdivided, and have precedence constraints as listed in Table 1.

A.2 Task completion time is deterministic.

A.3 Multiple tasks can be performed at each location node.

A.4 A task must be completed before the next task at the same node can commence execution.

A.5 Flight times between nodes are deterministic and identical for all UAVs, stochastic behaviour is implemented as delays afterwards.

A.6 Any stochastic delay is fully described by the pdfs associated with the stochastic zones demonstrated in Figure 2.

A.7 UAVs cannot overtake each other, the task plan fully describes the task sequence.

A.8 UAVs are never ahead of their schedule, only delays occur.

A.9 The time buffer is identical for all nodes.

A.1 is a basic assumption allowing for the task planning, as the management of partially completed tasks is beyond the scope of this paper. **A.2** is based on a set up of identical UAVs performing well defined tasks with high accuracy. **A.3** provides a means of considering the physical location of a UAV, in order to account for delays introduced by multiple UAVs working in the same environment simultaneously. **A.4** prevents UAVs from crashing while performing their tasks. **A.6** renders the problem manageable, while conditioning the validity of the estimated delay on the estimation of delay zones. **A.7** leaves the task plan as a valid process description, and maintains a semblance of order in our world. **A.8** relies on the conditions that flight times between nodes are based on UAV top speed, or that the UAV automatically slows down to avoid arriving early. **A.9** is implemented to facilitate the presentation of system delays as a function of time buffer size. **A.5** is the prerequisite for constructing a deterministic task schedule, while still allowing for subsequent stochastic influences. The *pdf*s associated with the different route types seen in Figure 2 are listed in Table 4, and will be used for all examples throughout this paper. If a configuration with more or different stochasticity zones is prudent for other problems, addition and exchange of *pdf*s to the delay simulation is a trivial matter provided that they are in accordance with the above assumptions.

3 Methodology

The UAVs are scheduled in a manner similar to the scheduling procedure of (Christensen et al., 2017), with feasibility restrictions modified to consider transport times, node occupancy, and time buffers, while only considering single task execution. This scheduling procedure is chosen for its ease of implementation and modification, alternate scheduling procedures are readily found in the literature, e.g., (Khosiawan and Nielsen, 2016), but beyond the scope of this paper, as any planning method consistent with previous assumptions is applicable to the delaying algorithm presented. With a task plan set, we focus on determining the delays caused by the stochastic influence of the flight times, and what effects the introduction of a time buffer has. The route planning is performed using integer steps to limit computational cost, while the delays are generated by continuous distributions to allow for greater realism. As the time buffer is incorporated in the route planning, this too is in integer steps.

3.1 Stochastic delays

Stochasticity of the process is handled by assigning different probability distributions to different flight paths, depending on which stochasticity zones the UAVs move in. An example of flight paths with different probability distributions is given in Figure 2, and the delays resulting from considering these stochasticity zones are elaborated on in Section 4.

3.2 Mathematical Formulation

Indices	
t	Index of assignable tasks $t = 1, 2, \ldots, T$.
u	Index of UAVs $u = 1, 2, \ldots, U$.
n_t	Index of task nodes $n = 1, 2, \ldots, N$, each associated with a task t.
ut_u	Tasks t assigned to a specific UAV $ut = 1, 2, \ldots, UT_u \forall u$.
tn_t	Node at which task t is performed $tn_t \in N, t = 1, 2, \ldots, T$.
tu_t	UAV u to which task t is assigned $tu_t \in U, t = 1, 2, \ldots, T$.
nt_n	Task performed at node n.
Parameters	
Prc	$T \times T$ binary matrix of precedence relations between tasks.
$tr_{s,t}$	Transport time between location nodes for task s and t.
tt_t	Time needed to carry out task t.
st_t	Start time of task t.
ft_t	Finish time of task t.
$del_{ut_u - 1, ut_u}$	Stochastic delay between two consecutive tasks assigned to UAV u.
B	Time buffer.
Decision variables	
$x_{t,u}$	1 if task t is assigned to UAV u.
$ud_{u,t}$	Delay of UAV u at time t.
$nd_{n,t}$	Delay of node n at time t.

Equations for the planning algorithm shown in Algorithm 2:

Objective:

$$\min C = \sum_{t=1}^{T} \sum_{u=1}^{U} x_{t,u} \cdot (tr_{s,t} + tt_t + B) \tag{1}$$

Subject to:

$$\sum_{u=1}^{U} x_{s,u} - \sum_{u=1}^{U} x_{t,u} \leq 0 \qquad \forall s \in Prc(t), \tag{2}$$

$$\sum_{u=1}^{U} x_{t,u} = 1, \qquad \forall t \in T, \tag{3}$$

$$x_{t,u} \in \{0,1\} \tag{4}$$

(1) concerns the minimization of cycle time, i.e., the time required for each task in T to be executed once, by considering transport time between tasks, task execution time, and a fixed time buffer. (2) describes the precedence constraints for each task. (3) ensures that each task is only assigned once. (4) defies the entry types in the assignment matrix.

Equations for the delay algorithm shown in Algorithm 3:

Objective:

$$\text{Total delay} = \max_{u \in U} \left(ud_u \right), \tag{5}$$

Subject to:

$$ud_u = \sum_{ut_u=1}^{UT_u} \left(\max \left(ft_{ut_u-1} + tr_{ut_u-1,ut_u} + del_{ut_u-1,ut_u} - st_{ut_u} - B, nd_{n_{nt-1}} \right) \right), \tag{6}$$

$$nd_{tn_t} = \max \left(nd_{n_{nt-1}}, ud_u \right), \tag{7}$$

Here (5) equates the total delay with the delay of the most delayed UAV. (6) concerns the delay of each UAV, with the delay of each task in the UAV task sequence being weighed against its corresponding node delay to determine which is the defining element. (7) manages the delay of the nodes by checking if the node is already too delayed to be affected by the delayed UAV. The various indices in Table 3.2 are intermingled in order to track which criteria which item is selected from.

3.3 The algorithm

The total delay is coded according to Algorithm 1, with Algorithm 2 supplying the task plan to be delayed, and Algorithm 3 outputting the relevant stochastic delay, as seen in Figure 4.

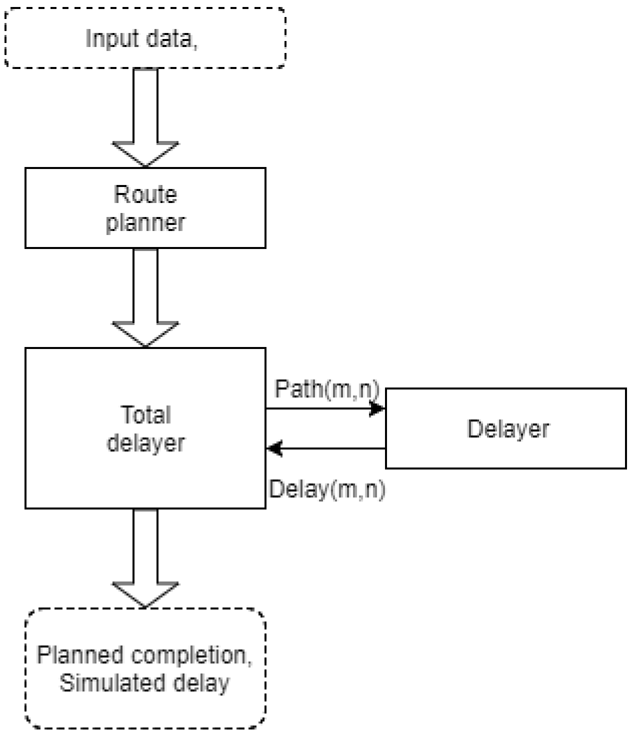

Figure 4: The overall procedure of simulating the total delay of a set of UAV tasks.

The algorithmic procedure outlined in Figure 4, functions by taking as input; task times and at which location they are each performed, transport times between the location nodes, the delay types associated with the transport paths, the precedence constraints for scheduling the tasks, the size of the time buffer, and the number of UAVs used for performing the tasks. Initially a task plan is generated by the TaskPlanner function, this task plan is then passed to the TotalDelayer function, which handles the accumulation of individual delays of flight between nodes. The individual delays are simulated by the Delayer function which is called for each task, and returns a delay based on which route it is called for, these routes are shown in Figure 2. The output of the TotalDelayer function is the total planned time, and the simulated accumulated delay when considering propagation through the tasks. The algorithms are shown and described in the proceeding part.

Algorithm 1 UAV delay simulator

```
function TotalDelayer(TransportTimes, TaskTimes, TaskLocations, DelayTypes, PrecedenceConstraints, Buffer, UAVs)
    Import route plan from *RoutePlanner* function
    Construct sequence of tasks for each UAV from route plan
    for First UAV task index,..., last UAV task index do
        for First UAV,..., last UAV do
            Update which UAV task is ”current”
            Update which UAV task is ”previous”
            Import delay of the current path from *Delayer* function
            if UAV is more delayed than task node delay then
                Update UAV delay with time exceeding planned start time
                if UAV delay is less than or equal to Buffer then
                    Set UAV delay to 0
                end if
                if UAV delay exceeds Buffer then
                    Subtract Buffer from UAV delay
                end if
                Set task node delay to UAV delay
            end if
            if Task node delay exceeds UAV delay then
                if Task node delay is less than or equal to Buffer then
                    Set task node delay to 0
                end if
                if Task node delay exceeds Buffer then
                    Subtract Buffer from task node delay
                end if
                Set UAV delay to task node delay
            end if
        end for
    end for
return Planned total time and Total delay
end function
```

Algorithm 1 *TotalDelayer* is the main function supplying the outputs for Section 4.1, taking as input; transport times, task times, task locations, delay types, precedence constraints, time buffer size, and number of UAVs. The algorithm makes use of *RoutePlanner* to construct a route plan for the delays to apply to, a fully external route plan of the correct format could be imported here instead. Afterwards the task plan is divided into lists of tasks for each UAV, in order to track UAV movement. The algorithm then cycles through all tasks, for all UAVs, in order to determine how the delays stack. For each UAV for each task a stochastic delay, dependent on which route the UAV traverses, is returned by *Delayer* and added to any pre existing UAV delays. If the UAV is then more delayed than the node, the delay of the node is set to be that of the UAV, and the time buffer is subtracted from the delay of both, while still requiring all delays to be non negative to avoid violating Assimption **A.8**. If the node is more delayed than the UAV, the UAV delay is set to be that of the node, and the time buffer is subtracted, as the UAV delay is now dominated by the pre existing queue at the node. After the delays of all tasks of all UAVs have been addressed, and the UAV- and node delays have been updated accordingly, the total delay is chosen to be that of the most delayed UAV, as the set of tasks is not finished till the last UAV is done. This, along with the planned finish time, becomes the output of the function, in order to allow for later representation of total completion time.

Algorithm 2 Route planning heuristic

```
function RoutePlanner(Transport times, Task times, Task locations, Buffer)
    while All tasks are not assigned do
        Increment time
        for First task,..., last task do
            for First UAV,..., last UAV do
                if Precedence criteria met then
                    if Both task node and UAV can finish task within time then
                        Update task plan with UAV and time of completion
                        Update UAV occupancy
                        Update node occupancy
                        Update UAV task sequence
                        Update task assignment counter
                    end if
                end if
            end for
        end for
    end while
return Task plan
end function
```

Algorithm 2 *RoutePlanner* provides the deterministic route plan as input to *TotalDelayer*, by gradually incrementing a time counter, and assigning an unassigned task at an unoccupied node to an unoccupied UAV, provided the task, including transportation, execution, and buffer time, can be completed within the time counter. The task plan and UAV task sequences are returned from the function.

Algorithm 3 Stochastic delay generator

```
function DELAYER(Task plan, Task locations, UAV task sequences, Delay types, Current task, Current UAV)
    if Current task is assigned to Current UAV then
        if Current task is first task for the UAV then return Delay for the path from UAV origin to Current task node
        end if
        if Current task is not first task for the UAV then return Delay for the path from previous to current task for the UAV
        end if
    end if
    return 0
end function
```

Algorithm 3 *Delayer* is the source of the stochastic delays for the system. By reading the entry of the delay type matrix with the [previous, current] tasks as coordinates, the delay distribution for the current path is chosen, and from this a delay is returned. The delay distributions used in this paper are listed in
The algorithms 1, 2, and 3 are implemented in C++, and the related .cpp files are attached.

4 Numerical examples

To illustrate the validity and usefulness of Algorithm 1, we first present a small example constructed using performance data as presented in (Park et al., 2015), and reiterated in Tables 1, and 2. We additionally make use of a matrix representation of the different delay types shown in Figure 2, which are also displayed in Table 3. The delay type distributions from which the delays are attained are listed in Table 4. The lognormal distribution type is chosen for the non-negativity property, and for not having the bulk of mass at zero, which would make the effects of delay stacking hard to discern.

Task	Duration	Node	Precedence
1	2	6	-
2	2	3	-
3	6	4	-
4	5	5	1
5	2	6	2
6	2	4	2
7	4	1	4
8	3	2	4,5
9	3	5	7
10	3	3	8

Table 1: Data for the tasks in the small numerical example.

Node	1	2	3	4	5	6
1	0	4	4	5	6	6
2	4	0	4	5	6	6
3	4	4	0	4	5	5
4	5	5	4	0	4	4
5	6	6	5	4	0	4
6	6	6	5	4	4	0

Table 2: Flight time between nodes in the small numerical example.

Node	1	2	3	4	5	6
1	0	1	4	4	6	6
2	1	0	4	4	6	6
3	4	4	0	2	5	5
4	4	4	2	0	5	5
5	6	6	4	4	0	3
6	6	6	5	5	3	0

Table 3: Delay types between positions in the small numerical example.

Path	Distribution
0	0
1	$\text{Lognormal}(0, 0.1)$
2	$\text{Lognormal}(0, 0.3)$
3	$\text{Lognormal}(0, 0.2)$
4	$\text{Lognormal}(0, 0.4)$
5	$\text{Lognormal}(0, 0.5)$
6	$\text{Lognormal}(0, 0.6)$

Table 4: Probability distributions with varying variance, for delays throughout this paper.

The parameters used for the examples containing more than ten tasks have been generated automatically by using the statistical programming language R, the code for these generations is found in Appendix A.

4.1 Results

The numerical results of applying the algorithms presented in Section 3.3 to a system with the above parameters, are attained by executing a C++application through *Rstudio*. The effects of buffer size on planned completion time, expected delay, and total production time, are presented in Figure 5.

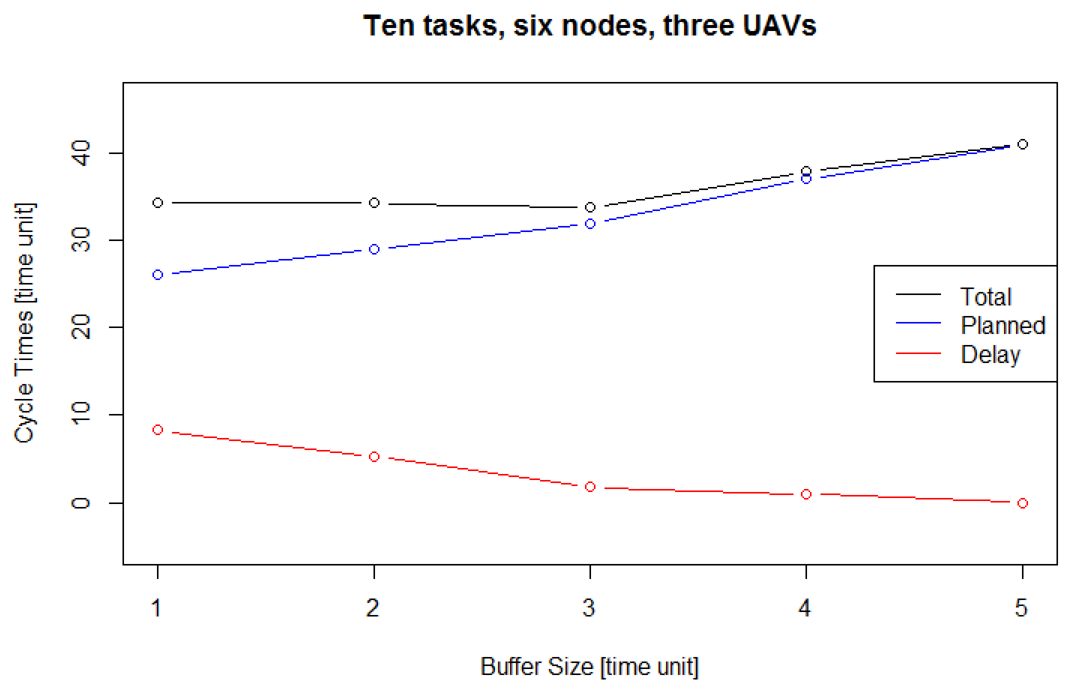

Figure 5: Ten tasks at six nodes performed by three UAVs.

The buffer size is clearly seen to have an obvious, but gradually diminishing, effect on the average delay of the system. The benefits of this delay reduction is eventually offset by the increase in planned cycle time, which leads to the establishment of an assumed optimal time buffer size of three time units. With a time buffer large enough to absorb any transport delays, the total and planned times become identical. To demonstrate the broader applicability of the algorithms, a broader set of different production parameters are randomly generated in Rstudio, and the C++program is applied. By varying the number of tasks, nodes, and UAVs, the general tendencies of increasing the time buffer, regardless of problem size are demonstrated. The resulting times are seen in figures 6, 7, 8, and 9.

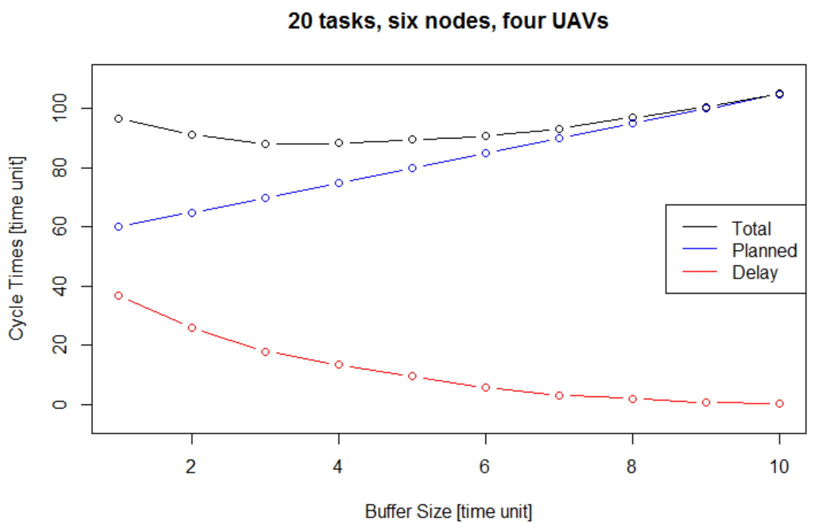

Figure 6: 20 tasks at six nodes performed by four UAVs.

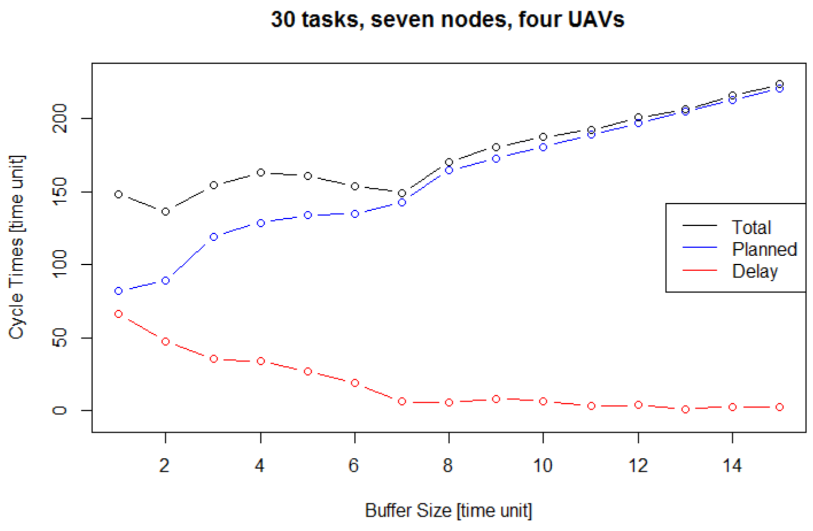

Figure 7: 30 tasks at seven nodes performed by four UAVs.

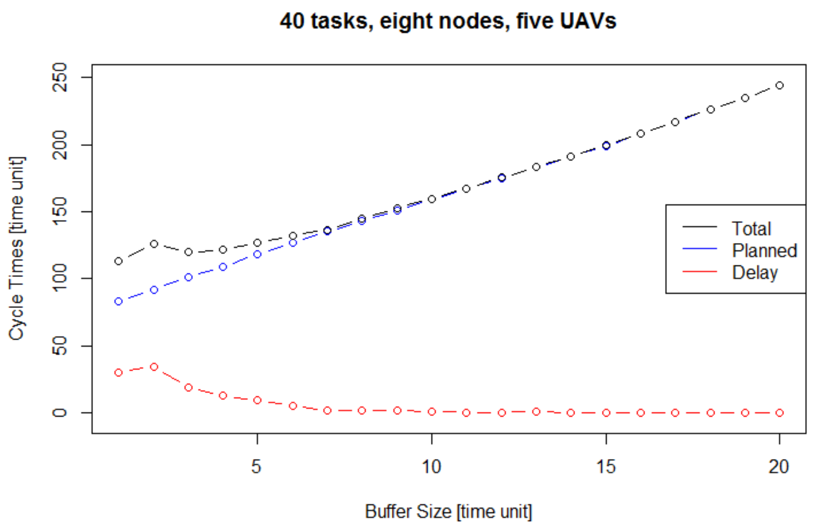

Figure 8: Forty tasks at eight nodes performed by five UAVs.

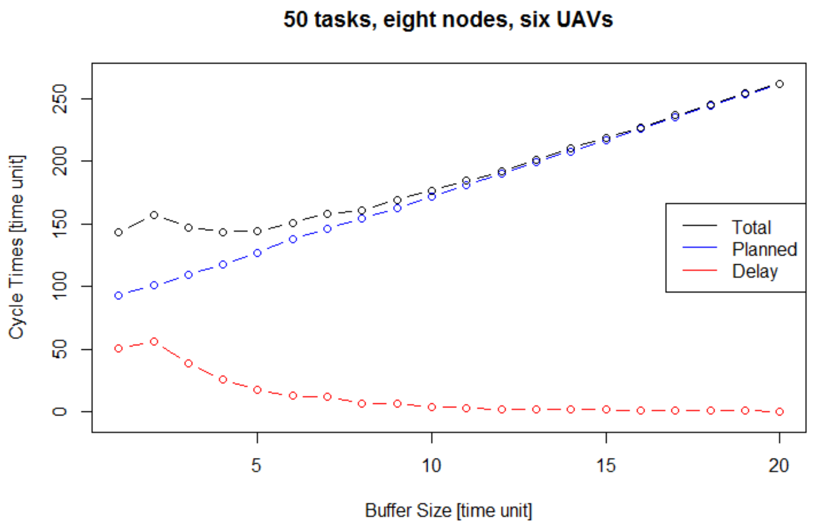

Figure 9: 50 tasks at eight nodes performed by six UAVs.

These figures all display the same tendencies as seen in Figure 5, with some slight irregularities at certain buffer sizes. Irregularities such as these are expected from the stochastic nature of the delays. Furthermore, the convergence of the delay to zero as the time buffer increases, is clearly evident in all figures, but the nature of the *pdf*s, as listed in Table 4, excludes complete certainty of no delays throughout the system, regardless of buffer size.

5 Conclusion

The algorithms presented in this paper provide a stable approach to assess the effects of implementing a time buffer in a production environment modelled with different stochasticity zones. The possibility of easily changing the type, number, and segmentation of the stochasticity zones considered, makes the algorithms in Section 3.3 applicable to a wide range of problems related to stochastic behaviour of conveyance in production environments. Assigning different types of stochastic influences to different sections of an area of operations for a UAV, and tracking how these stochastic influences affect each other, makes for a versatile way to mirror the conditions of UAV application in an indoor production environment, and thus ought to be of use to production planners working in such. An advantage of our delay simulating algorithm is the compatibility with different route planning approaches, which allows for almost seamless integration into pre-existing planning systems. Application of the algorithms in this paper adds an additional step to any planning procedure, but may in many cases result in greater planning stability.

References

Bachrach, A., R. He, and N. Roy (2009). Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles 1*, 217–228.

Christensen, M. K., M. N. Janardhanan, and P. Nielsen (2017). Heuristics for solving a multi-model robotic assembly line balancing problem. *Production & Manufacturing Research 5*, 410–424.

Dadkhah, N. and B. Mettler (2011). Survey of motion planning literature in the presence of uncertainty: Considerations for uav guidance. *Journal of Intelligent & Robotic Systems 65*, 233–246.

Elshaer, R. and H. Yamamoto (2012). New proactive time buffer heuristics for robust project scheduling. *Journal of Advanced Mechanical Design, Systems, and Manufacturing 6*, 559–571.

Jacobs, F. R., W. L. Berry, D. C. Whybark, and T. E. Vollmann (2011). *Manufacturing Planning and Control for Supply Chain Management.* McGraw-Hill Education.

Khosiawan, Y. and I. Nielsen (2016). A system of uav application in indoor environment. *Production & Manufacturing Research 4*, 2–22.

Kristiansen, R., E. Oland, and D. Narayanachar (2012). Operational concepts in uav formation monitoring of industrial emissions. In *3rd IEEE International Conference on Cognitive Infocommunications.*

Oh, H., D.-Y. Won, S.-S. Huh, D. H. Shim, M.-J. Tahk, and A. Tsourdos (2011). Indoor uav control using multi-camera visual feedback. *Journal of Intelligent & Robotic Systems 61*, 57–84.

Park, Y., Y. Khosiawan, I. Moon, M. N. Janardhanan, and I. Nielsen (2015). Scheduling system for multiple unmanned aerial vehicles in indoor environments using the csp approach.

A Example parameter generating R-code

The R-code used to generate parameters fot the examples in Section 4.

```
Delays <- matrix(rep(0, BufferRange*Trials), nrow = BufferRange, ncol =
    Trials)
CycleTimes <- vector()
TotalTime <- vector()
MeanDelay <- vector()
Precedence20 <- rbinom(Tasks,c(0,1),c(0.9,0.5))
Matrix20 <- matrix(Precedence20, nrow = Tasks*Tasks, ncol = Tasks)
for (i in 1:Tasks){
  Matrix20[i,1] <- 0
  Matrix20[i,i] <- 0
}
Precedence <- as.vector(Matrix20)
TaskTime <- sample(x = 2:8, size = Tasks, replace = T)
TaskLocation <- sample(x = 0:(Nodes-1), size = Tasks, replace = T)
TransportVec6 <- sample(x = 5:15, size = Nodes*Nodes, replace = T)
Transport6 <- matrix(TransportVec6, nrow = Nodes, ncol = Nodes)
Transport6[upper.tri(Transport6)] = t(Transport6)[upper.tri(Transport6)
    ]
diag(Transport6) = 0
TransportTimes = as.vector(Transport6)
DelayVec6 <- sample(x = 1:6, size = Nodes*Nodes, replace = T)
Delay6 <- matrix(DelayVec6, nrow = Nodes, ncol = Nodes)
Delay6[upper.tri(Delay6)] = t(Delay6)[upper.tri(Delay6)]
diag(Delay6) = 0
DelayType = as.vector(Delay6)
for(i in 1:BufferRange)
{
  for(j in 1:Trials)
  {
    Delays[i, j] <- TotalDelayer(TransportTimes, TaskTime, TaskLocation
        , DelayType, Precedence, Nodes, Buffer = i-1, Tasks, UAVs)[2]
  }
  MeanDelay[i] <- sum(Delays[i,])/Trials
  CycleTimes[i] <- TotalDelayer(TransportTimes, TaskTime, TaskLocation,
      DelayType, Precedence, Nodes, Buffer = i, Tasks, UAVs)[1]
  TotalTime[i] <- CycleTimes[i] + MeanDelay[i]
}
```