Improving Fruit Harvesting Speed with Multi-Armed Robots

By

NATALIE C. PUEYO SVOBODA
DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Biological Systems Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

---

Stavros G. Vougioukas, Chair

---

Zhaodan Kong

---

Brian Bailey

Committee in Charge

2024

Dedicated to ...

I dedicate this dissertation work to my mom, Monika Svoboda, my sister, Nicole Pueyo, and Abhinav Sinha who supported and encouraged me throughout the whole process. This work would not exist without them. I should also dedicate this to my dog, Nova. She kept me moving even during the darkest of times.

Also, a thank you to all my friends, lab mates, and relatives who encouraged, supported, and inspired me.

A special thank you to my mentor, Stavros Vougioukas, whose support was instrumental during my journey. He gave tremendous support, encouragement, and assistance, as well as the freedom I needed, not only to grow, but to flourish during my graduate career.

# Contents

# 1 Abstract

Successfully automating the harvest of fresh-market apples requires that robots match or exceed the apple picking throughput (FPT) of humans, at 0.83 fruits/s, while achieving a high fruit picking efficiency (FPE, percent of harvested fruits). To this end, robotic harvesters are built with multiple arms, though this introduces challenges. Fruit distributions are non-uniform, requiring workload balancing between arms. This can be accomplished through strategies such as flexible arm workspaces, having the harvester in constant motion, and choosing the right scheduling algorithm.

To address the need for workload balancing, we simulated a constantly moving harvester with multiple 3-degree of freedom arms sharing column workspaces. The arm's movements were constrained by the column's frame and software-defined vertical limits which prevented collisions between the arms. Initially, the harvester scheduled 3.5 m sections of orchard rows. It knew the location of all fruits, so could optimize its vertical arm limits, choose a "best," fixed harvester speed, and compute the schedule for that orchard row segment. Two scheduling algorithms were evaluated, First Come First Served (FCFS) and a Mixed Integer Linear Programming formulation based on Goal Programming (GP). Later, harvesting was extended to the whole orchard rows using a new, semi-dynamic strategy, the Sliding Planning Window (SPW). The orchard row was solved as a series of individual, short, sequential, and overlapping planning windows which maximized the cumulative Orchard Row-FPT (OR-FPT) and OR-FPE.

Simulation experiments with real fruit distribution data validated our approach, resulting in combined throughput and efficiency gains for multiple harvester configurations. Increasing the number of arms increased the combined FPT and FPE. Using FCFS and the best harvester speed for nine arms, partitioning the columns to make rows of arms containing the same number of fruits resulted in a mean of 1.374 fruits/s compared to 1.049 fruits/s when the

rows of arms were all the same height. Both achieved at least 95% FPE. Scheduling with GP improved the mean FPT to 2.0 fruits/s. When harvesting whole orchard rows, SPW always achieved at least 95% OR-FPE, however the amount of overlap between planning windows affected OR-FPT. No overlap resulted in a mean OR-FPT of 1.0 fruits/s. Overlapping half the previous planning window achieved 1.86 fruits/s. Further increasing the overlap decreased the OR-FPT. These results show that workload balancing could be used to increase the throughput of multi-armed harvesters and introduces a way to do this semi-dynamically.

# 2 Introduction

## 2.1 Motivation

In the $14 to $18 billion U.S. fresh-market fruit industry, labor takes up between 30% to 60% of total operational costs Q. Zhang and Karkee, 2016. In April 2020 alone, around 688 thousand farmworkers were hired in the US *Farm Labor (May 2020)*, 2020. Because of this, the industry has been heavily impacted by recent, country-wide labor shortages Blanco, 2016.

In 2019, 56% of surveyed farmers in California alone reported labor shortages, especially for crops that are labor-intensive *Adapting to Farm Worker Scarcity Survey 2019*, 2019, with no end in sight. Real farm wages across the U.S. have been on the rise for decades Zahniser et al., 2018, with the last five years seeing a record increase of 2.8% per year due to labor shortages "USDA ERS - Farm Labor," n.d. During the same time-frame, there has been a rise in demand for H2-A workers, a very expensive option for employers Maria and Salassi, 2019; "USDA ERS - Farm Labor," n.d. The shortage has been exacerbated by improving conditions in Mexico and a growing focus on stronger enforcement at the US-Mexico border, Haspel, 2017. With the lack of labor, farmers face losing their crops. In California, 18% of growers that responded to a 2017 survey by the California Farm Bureau either downsized or did not harvest their crops *Agricultural Labor Availability Survey Results*, 2017. In 2019, an updated survey found that 31% of surveyed farmers have switched away from high labor acreage like fruit orchards to low maintenance crops like nut tree orchards over the last five years, and 56% of respondents said that they are turning increasingly towards technology to help them decrease their reliance on labor *Adapting to Farm Worker Scarcity Survey 2019*, 2019. This same trend towards mechanization is being seen throughout the agricultural industry in the U.S. Blanco, 2016.

## 2.2 The challenge of high-speed robotic fruit harvesting

Two technologies are being explored to automate fruit harvesting. The first is shake-and-catch, where the trees are shaken to dislodge the apples, which are caught by the system as they fall down. Although very fast, shake-and-catch systems do not work for fresh-market apples because it causes bruising (Calvin et al., 2022; Z. Zhang et al., 2016). The second technology is a selective harvester, a robot that can harvest apples one-by-one without causing damage. This technology has been in development for over 50 years with the goal of improving two major performance metrics: Fruit Picking Efficiency (FPE) and Fruit Picking Throughput (FPT). The first, FPE, measures the percent of harvestable fruits harvested by the system. The second metric, FPT, measures the speed of harvest in fruits per second. A robotic harvester must have both high FPE and FPT to be successfully commercialized.

Early robotic harvesters lacked adequate fruit identification technology and had difficulties working within the three-dimensional canopies (Q. Zhang and Karkee, 2016; Q. Zhang and Pierce, 2016). Leaves, branches, and other fruit impacted both FPE and FPT through occlusion of harvestable fruits. Robotic harvesters were deemed important enough, however, to spur changes in tree fruit breeding and orchard management. Fruit trees were bred and trained into trellis structures that flattened out the canopies, making it easier to see and reach the fruits. These changes decreased occlusion and eased the reachability constraints. Furthermore, detection rates dramatically improved through the use of machine learning, as shown in Bac et al., 2014; Koostra et al., 2021. For example, in Tang et al., 2020, the use of neural networks gave a recognition accuracy of 95.35% for citrus identified in its natural environment. These developments have once again spurred interest in robotic harvesters, both in academia and in industry. Systems can now target a close-to 100% FPE, since a large majority of harvestable fruits can be identified and reached. This is important because a crucial economic study into automated fruit harvesting by Harrell, 1987 shows that FPE has a higher sensitivity than FPT. However, this comes with a challenge: FPE and FPT are

conflicting metrics.

Harvesting fruits takes time. Z. Zhang et al., 2016 measured people harvesting speeds at 0.83 fruits/s while robots harvested 0.06 fruits/s. This difference in FPT is problematic because growers require both speed and efficiency. As shown in the 2014 and 2021 agricultural robot reviews Bac et al., 2014; Koostra et al., 2021, since the 1970s, the main focus of robotic harvesting research to improve FPT has been motion planning. Most research during that time focused on increasing speeds for arms with six degrees-of-freedom (DOF) which are expensive to run and complicated to control, but have the range of motion needed to move within complicated environments like tree canopies and between leaves. Results of motion control research, however, did not have significant enough effects on harvesting speeds to offset the cost of the robots Bac et al., 2014; Koostra et al., 2021. With this in mind, both academia and companies, such as FFRobotics and Advanced.Farm, pivoted towards harvester concepts that use multiple arms with only three degrees of freedom (DOF), decreasing motion control complexity and cost, while increasing overall harvesting speeds. Such "simple" arms perform very well in flat systems such as the trellised orchards; in fact, Edan and Miles, 1993 showed that Cartesian robots harvesting melons had better reach, and were easier to control than a more complicated system. Another example is the multi-armed harvester described in Li et al., 2023 which harvested a trellised orchard row, section-by-section, with an average FPE of 79.31% and an average FPT of 0.17 fruits/s for all sections. On the commercial side, Advanced.Farm described being able to harvest an average of 0.17 fruits/s with the goal of increasing it to 0.27 fruits/s in their grant proposal "2023 Technology Research Review," 2022. Results such as these show that multi-armed harvester can increase FPT while keeping FPE high. However, it is important to note that to do this, the arms have to work together and be allocated to the fruits correctly (Edan and Miles, 1993; Recce et al., 1996). This task requires scheduling many arms in real-time while taking into account continuously changing fruit distributions that are non-uniform. It is challenging because even a single arm scheduling

problem falls under known $\mathcal{NP}$-hard problems, while the multi-armed problem is strongly $\mathcal{NP}$-hard (Garey and Johnson, 1978; Gerkey and Matarić, 2004); the problem space for task assignment increases exponentially with the number of tasks and arms. Furthermore, complicates arise from the clustering and non-uniform distribution of fruits and the dynamic nature of orchards, where fruit locations can change while harvesting takes place. To keep things simple, existing research into scheduling for multi-armed harvesters, such as Li et al., 2023; Mann et al., 2016; Recce et al., 1996; Scarfe, 2012, mainly focus on developing scheduling that maximizes the FPE. Adding the second optimization objective, FPT, increases the complexity of the problem; However, as previously noted, robotic harvesters will require high FPE and FPT to be successful. Thus, there is a need for a method to schedule many arms in real-time which can adapt to changes in the incoming fruit distribution and changes to the fruits being immediately harvested.

## 2.3 Objective statement

The main goal of this dissertation is to improve the Fruit Picking Throughput (FPT) of multi-armed, fruit-harvesting robots while keeping the Fruit Picking Efficiency (FPE) high, through the development of a scheduling framework for arm-to-fruit assignment. Such a framework will be able to:

1. Model the fruit picking process with arms working in series and in parallel with flexible workspaces,

2. Introduce workload balancing strategies to improve scheduling results,

3. Compute the best schedule and harvester speed combination which maximize FPT while meeting a minimum FPE threshold,

4. Use the sliding planning window to optimize the schedule, harvester speed, and arm row limits dynamically along a whole orchard row,

5. Determine how FPT changes as the number of arms increases.

The dissertation will describe the algorithmic and experimental tools enabling the design, modeling, simulation, and optimization of the multi-armed robotic harvester scheduling framework.

## 2.4 Contribution of this thesis

The contributions of this thesis include:

1. Proposing a realistic way to make arm workspaces flexible in one dimension while removing the need for collision avoidance,

2. Evaluating dual-optimization strategies for constantly moving harvesters which can compute the best schedule and harvester speed combination to maximize FPT while meeting a minimum FPE threshold,

3. Presenting a novel way to harvest whole orchard rows dynamically and near-optimally while only knowing the location of fruits in range of the harvester's cameras.

# 3 Literature Review

## 3.1 Defining the general problem of arm-to-fruit scheduling for multi-armed harvesters

Multi-armed, fruit harvester scheduling falls under the optimization problems known as multiagent task scheduling. This is a field which attempts to determine a schedule that pairs agents to tasks in an effort to optimize an objective; for example, creating a schedule to finish all tasks as quickly as possible or finish the most tasks with the least agents. For multi-armed harvesters, scheduling means to create schedules that determine which arms should harvest fruit and when. Scheduling can be used to optimize both the number of fruits harvested and the speed of that harvest; however, only single objective optimization was used for multi-armed harvesters until Yang et al., 2022 introduced the use of dual-objective optimization for a mushroom harvester.

Many optimization problems are $\mathcal{NP}$-hard problems which take a lot of computation power and time to reach a solution. Formally, the multiarmed harvester fits in the general problem described by the oft-cited Gerkey and Matarić, 2004 as multiple robots that can perform a single task at a time and each task require at most a single robot to complete, with time extended assignment (ST-SR-TA). An arm can harvest a single apple at a time, and each apple only requires a single arm. The last descriptor, time extended assignment, is used when the system has existing knowledge of multiple jobs. Harvesters commonly have knowledge of either all the fruits in the row or all the fruit in the section they are working in. Understanding the general problem is important because, in optimization, related problems can be solved in the same, or a similar, fashion. However, complications arise because this particular problem type is a well-known, strongly $\mathcal{NP}$-hard problem which cannot be solved optimally except for very small instances. Not only does the amount of time it takes to find an optimal schedule grow exponentially as the number of fruits and arms increase, but, for these larger problem

spaces, there is no way to prove if a solution is the optimal one in real time. Thus, most research into scheduling for multi-armed harvester focuses on ways to reduce the complexity, through strategies such as the use of heuristics.

## 3.2 The evolution of scheduling solutions for multi-armed harvesters

The first paper which mention scheduling for multi-armed harvesters is Edan and Miles, 1993. It used the Traveling Salesman Problem (TSP) to schedule the harvest of melons using multiple arms working in parallel across the melon row. The objective was to harvest all fruits, using multiple arms to speed up the overall speed. Edan and Miles, 1993 was the first to show that as the number of arms working in parallel increased, the average amount of time to harvest a fruit decreased. However, the largest decrease in average time to harvest was with the addition of the first arm; after three arms, the drop in average time to harvest was less than 1% due to the non-uniform nature of the fruit distribution.

Scheduling for multi-armed kiwi harvesters appeared next, in Scarfe, 2012; Williams et al., 2019. Their research focused primarily on cluster dependencies and collision avoidance because picking the incorrect kiwi in a cluster can cause other fruits to fall or lead to arm collisions. In Scarfe, 2012, this was done by locating all fruits and clusters in the area above the robot and dividing the fruits approximately equally among the arms with complete clusters assigned to individual arms. This was a naive way of scheduling which did not attempt to optimize the schedule, but it could be performed in real time. However, in-field tests resulted in an FPT per arm of 0.45 fruits/second/arm, half the desired speed. This system was expanded in Williams et al., 2019 which used safety distances between arms to avoid collisions, rather than dividing the fruits into groups for each arm. Based on the reported seconds per fruit, the four-armed harvester could pick 0.182 fruit/s on average, and harvested close to 50% of the fruits. The largest time sink was the detection system which added 3 s/image at each

detection step.

Zion et al., 2014 was the first to formulate the problem as a non-TSP optimization problem. The paper examined two different combinatorial algorithms, the *k-colorable subgraph problem* and the *taxi dispatcher problem* to schedule melon harvesting for robots with multiple 2-DOF, Cartesian arms. Both combinatorial-based algorithms produced near-optimal to optimal results for the 2-DOF arms and solved in seconds for whole rows of melons. However, the team hypothesized that 3-DOF arms, which could not use these heuristics, would produce better results. Work by the same team, Mann et al., 2016, expanded the previous research to 3-DOF arms by formulating the melon harvesting problem as a time dependent team orienteering problem with time windows (TDTOPTW). The TDTOPTW was approached as a combinatorial graph-coloring problem which could work with higher fruit densities, tested up to 5 fruits/$m^2$, and solved much faster than TSP or the branch-and-prune methods. In fact, solutions took polynomial solving time, and it was proved optimal when solving for maximum FPE. However, this formulation had some limits: the combinatorial approach required that the minimum speed be limited to removed cycles in the schedules which meant that it would have difficulties working with high density crops such as apples. The paper also showed that 3-DOF outperforms 2-DOF in most cases, though, at lower speeds ($< 0.15$ m/s), the resulting FPE for the two systems is very close.

In Barnett, 2018; Barnett et al., 2020, kiwi fruit identification and localization were done offline and separately from the harvesting, but the partitioning for scheduling was done in real time. The system divided the fruit evenly between arms to load balance, with a second stage to reallocate fruits to arms if a fruit had dependencies. In Barnett, 2018 (see Fig. 1), the reallocation was done either based on the fruit's hanging height and $x$-coordinate, or through a greedy TSP approach based on the nearest next fruit. The two algorithms resulted in almost identical results, even though there was a greater Euclidean traveled distance when

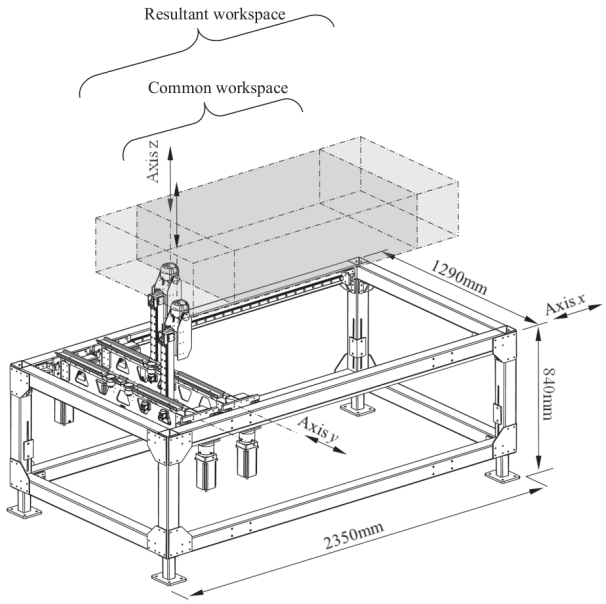



Figure 1: Concept art of a multi-armed kiwi robotic harvester from Barnett et al., 2020 with two arms sharing a workspace along the $x$-axis.

using the x-rank algorithm. On the other hand, the TSP algorithm was plagued by collision issues that the x-rank algorithm did not have. In the x-rank algorithm all the arms traveled in the same direction which physically avoided the problem. The x-rank algorithm was then used in Barnett et al., 2020 with Cartesian arms to test if it performed better than the articulated arms in the presence of clustering. Cartesian arms resulted in more uniform workload distribution between the arms and shorter task completion times in nine out of ten in-field tests.

Arikapudi, 2019; Arikapudi and Vougioukas, 2023 were the first two papers to examine scheduling for multi-armed harvesters in a grid-like configuration with arms working independent of each other in disjoint, rectangular work cells. The matrix structure alongside TSP resulted in high FPE values in orchards with two different tree structures each with more than 2000 fruits. Arikapudi and Vougioukas, 2023 also studied how the number of arms, and the workspace partitioning into cells for these arms, affected FPE and FPT. Similarly to Edan and Miles, 1993, results in Arikapudi and Vougioukas, 2023 showed that increasing

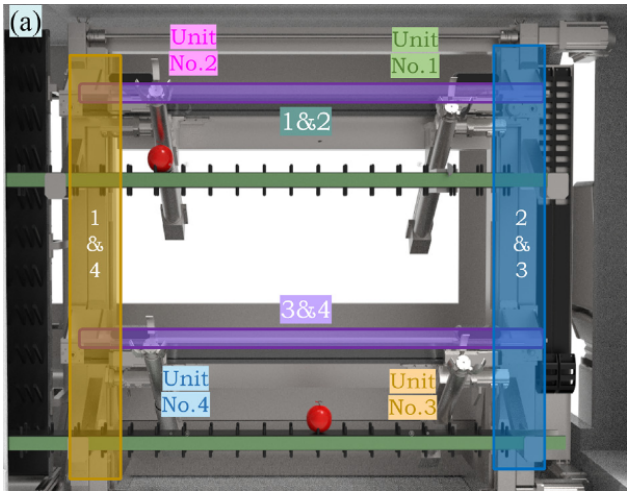



Figure 2: Concept art of a multi-armed kiwi robotic harvester from Li et al., 2023 where horizontal arms are coupled.

the number of arms increased overall harvesting speeds, but after a threshold, provided diminishing returns per-arm. The mean seconds per fruit ($1/FPT$) followed a power law as a function of arms per machine, irrespective of cell or harvester configuration. Dividing the workspace into cells based on equal number of fruits produced better results than dividing it into equally sized cells. This was because the division of cells by equal number of fruits better balanced the workload between the arms when testing on real fruit distributions.

Li et al., 2023 posed the multi-armed harvester scheduling problem as a vehicle routing problem solved using the Markov game framework to devise an efficient picking sequence and multi-agent reinforcement (MARL) as a target allocation scheme focused on cooperation between the arms. The goal of this project was to minimize overall operational time for four robotic arms set in a 2x2 pattern. As shown in Figure 2, the horizontal arms are coupled and must move up and down together. Over all sections, the proposed scheduling strategy had an average FPE of 79.31% with an average FPT of 0.17 fruits/s. When tested against a Genetic Algorithm (GA) heuristic, their MARL strategy took around 1 s to finish compared to around 8 s by GA. Furthermore, GA was unable to handle situations where an arm failed to grasp a fruit leading to more harvesting rounds while the MARL strategy only required

one round because a failed task could be renewed immediately. The authors believe that results could be improved by changing the mechanical design to have decoupled horizontal arms and a way to change the arm's angle of attack to reach fruits that were partly obscured by leaves, branches, or wires.

Multi-objective optimization was used to schedule a multi-armed mushroom harvester in Yang et al., 2022. It combined a Genetic Algorithm decompose a multiple TSP problem into individual TSP problems and Ant Colony Optimization to compute the shortest trajectory of each arm. The goal of the trajectory optimization was to get a both a high FPE and FPT while avoiding collisions between arms. The harvester achieved an FPT of 0.33 fruits/s and 97% FPE, producing schedules that were 21% faster than a two-chromosome genetic algorithm and 15% better a genetic stepwise algorithm. It was noted that the operation time of the algorithm needs to be improved to allow the system to work in real-time.

### 3.2.1 Pick-and-place, an alternate look at the same problem

Task scheduling for conveyor belts shares many of the complications seen in multi-armed fruit harvesting: they are online, have time-windows due to the conveyor belt movement, incoming objects have non-uniform distributions, both throughput and efficiency have to be maximized, etc. They have a strikingly similar problem space based on the ST-SR-TA general form and are already found in commercial settings, making scheduling for pick-and-place conveyor systems a great case study for multi-armed harvesters. Interestingly enough, conveyor scheduling has gone a very different route than research in multi-armed fruit harvesting scheduling, with simpler and less optimal task scheduling being the preferred commercial method. A majority these systems, such as Daoud et al., 2014; Huang et al., 2012, 2015; Humbert, Pham, et al., 2015, focus on determining which combination of dispatching rules—first-in-first-out (FIFO), last-in-first-out (LIFO), shortest processing time (SPT), etc.—should be used for a particular line of arms to obtain the best throughput and efficiency. However, there is more

experimental research that looks to speed up or better optimize scheduling which include: Bozma and Kalalıoğlu, 2012 uses non-cooperative game theory while Choudhury et al., 2022; Tika et al., 2020 focus on multi-layered approaches.

'Part dispatching rules' such as FIFO and SPT have low computation costs and, in single-armed, under loaded systems, can be proved optimal for various cases (Mattone et al., 1998). Simple dispatching algorithms can quickly become overloaded when the average distance between consecutive objects becomes too small and lose many items. Their low computation cost, however, makes up for their problems. Most research into this space (Daoud et al., 2014; Huang et al., 2012, 2015; Humbert, Pham, et al., 2015) focuses on determining the best combination and order of dispatching rules. The results remain suboptimal but can generally be calculated in real time.

More experimental strategies include Bozma and Kalalıoğlu, 2012; Choudhury et al., 2022; Tika et al., 2020. Bozma and Kalalıoğlu, 2012 has arms make decisions independently based on local information, minimizing both communication and complexity overhead. Results show that it picks up the same number of objects as a simple strategy where arms pick up the closest objects, though the workload distribution is more even among the arms. Tika et al., 2020 uses a two-layer optimization-based control policy involving task scheduling in the top layer and path planning, along with the motion constraints, at the bottom one. Unfortunately, results showed that the strategy was limited due to the complexities inherent in the problem; tests in Tika et al., 2020 could only be performed using two non-Cartesian arms and a maximum of 12 objects. A more recent example of the use of multi-layers by Choudhury et al., 2022 describes a hierarchical strategy that can work under uncertainty (what if the arm misses a task) and time constraints. Scale-ability is still an issue, however, with the task allocation as the bottleneck.

## 3.3 Research gaps and challenges

### 3.3.1 Robot model with arms in series and in parallel

Edan and Miles, 1993 shows that there are benefits to multiple arms in series or in parallel; since then, most research based itself on harvesters with many arms in one of those two configurations. A single row or a single column of arms is possible for lower density fruits, such as for melons in Edan and Miles, 1993; Mann et al., 2016, both of which resulted in high FPE values when the harvester had around six arms. For orchards, where fruits are produced at higher densities, the harvester might require a grid-like configuration to handle the non-uniform fruit distributions and to keep the robot's size practical. Both Arikapudi and Vougioukas, 2023; Li et al., 2023 show that the matrix configuration are beneficial, though the workspace configuration of the arms matters a lot. Arikapudi and Vougioukas, 2023 states that the best arm configurations are either having the arms in a single row or in matrix-like configurations, where the total arms are divided into two rows or two columns. Importantly, dividing the workspace into cells based on equal number of fruits produces better results than dividing it into equally sized cells. By doing so, the workload is balanced for all arms. The multi-armed harvester in Li et al., 2023 also uses a matrix-like configuration for four arms, however the arms in each row are coupled and move together in the vertical direction. This coupling was a challenge and the study's conclusion specifically stated that better results could be obtained if the arms were completely independent of each other. Both show that multiple arms with thoughtful work cell partitioning could lead to better FPE and FPT results; the challenge is to do this dynamically while avoiding collisions between arms.

### 3.3.2 Optimizing both FPE and FPT using dual-objective optimization

Although all research into scheduling for multi-armed harvesters reports both FPE and FPT (or 1/FPT), optimization is generally performed for one or the other, not both. For example, in Mann et al., 2016, the heuristic maximizes FPE; changes to FPT are incidental or require

manual changes between tests which are unoptimized and cannot be optimized without changes to their setup. The opposite is true of papers that prioritize FPT, such as in Li et al., 2023, where all identified fruits are scheduled to be harvested, and optimization is performed to minimize the traversal times between fruits. FPE changes in Li et al., 2023 are caused by failed grasping attempts rather than as part of the optimization process. This is problematic because optimizing for a single objective limits the results to either a maximum FPE or FPT, when the "optimal" solution is likely to be a trade-off between the two. In "2023 Technology Research Review," 2022, under the "Automated Apple Harvester" by the company Advanced.Farm, an economic analysis of their current system defines success as harvesting between 30 to 50% of the apples at a speed of 0.5 fruits/s. The harvester in Li et al., 2023, while scheduling for an FPE of 100%, when including the start-and-stop motion of the whole system, achieved FPT results of 0.114 and 0.132 fruits/s on two orchard sections with final FPE values of 71% and 81%. Because FPE and FPT are in opposition, as shown in Edan et al., 1993, it is likely that dropping the hard 100% FPE requirement could lead to a higher FPT that is closer to a commercial system's definition of success. The mushroom harvester in Yang et al., 2022 achieved the best FPT at 0.33 fruits/s with a success rate of 97% by only guaranteeing that their FPE would be above 95%. However, as seen for mushroom harvester, this requires that the problem be treated as a multi-objective problem that look for solutions that optimize both FPE and FPT, rather than a single metric. These minimum values should be determined by the growers and economic analysis for the specific crops. Two types of multi-objective optimization solutions exist: (a) solving for the Pareto front, a set of all solutions that cannot be improved without degrading at least one of the objectives, and (b) solving for a single value in the Pareto front by converting the multi-objective problem into single-objective optimization problem, called scalarization. The first method shows what Pareto optimal FPE and FPT combinations could be possible and the trade-offs that would make these combinations possible, making it useful for analyzing the effects of design parameters on robotic harvesters. Scalarization, on the other hand, is useful when computing

harvesting schedules used by the robot. The design of the robot is set, and the minimum required FPE and FPT values are known by this point, so a set of optimal schedules is superfluous and time-consuming to go through. Instead, with scalarization, a single schedule that meets the desired specifications is computed and followed by the harvester.

### 3.3.3 Dynamic scheduling and vehicle speed selection

Vehicle motion was evaluated for a single arm harvester in Edan et al., 1993. Results showed that continuous vehicle motion decreased the seconds per fruit (1/FPT), with variable vehicle speed outperforming constant vehicle speed. By matching the vehicle speed with incoming fruit density, the system was 3.3 % faster and was able to pick a higher percentage of fruits. This can be compared with results in Li et al., 2023; Yang et al., 2022, which have the harvester advance along the crop row, stop at each new grouping of fruit to harvest them, and moving forward again to the next segment with fruits. The start-and-stop strategy in Li et al., 2023 caused a difference between the mean reported FPT for all sections where the vehicle stopped, and the global FPT, reported as 'Number of fruits harvested per hour,' for two in-orchard tests. The average 'section' FPT was 0.17 fruit/s, while the global FPT for the two tests was 0.13 fruits/s and 0.11 fruits/s. These results indicate that continuous motion along the crop row with variable speed is desirable, but it adds challenges to computing schedules. Edan et al., 1993 changes the speed based on the number of fruits in incoming clusters which is an algorithm that can be cheaply and quickly computed. However, the number of incoming fruits in a cluster as a measure of speed might not work with multi-armed systems which are, as shown by Arikapudi and Vougioukas, 2023, affected by the non-uniform distribution of fruit, not just the density. Treating the vehicle speed as an optimization variable is possible, and confers benefits such as: (a) the speed can be computed automatically alongside the schedule, providing a way to get the best combination of speed and schedule, (b) don't have to worry about determining the parameters of the incoming fruit distribution that affect the optimality of the speed However, this increases the size of an already large problem space,

more so if the speed needs to be dynamic, changing as the incoming fruit distribution changes. It is likely that the problem becomes intractable unless strategies are employed to either reduce the problem space or pose the problem as a solvable heuristic. Potential strategies for this type of problem include multi-layered scheduling, such as in Tika et al., 2020 where TSP and Model Predictive Control are used to calculate the schedule and the trajectory, in that order. Of note, the schedule is recalculated every so often as new items approach the arms on the conveyors. This approach is similar to Edan et al., 1993, which also recalculates the schedule as new information becomes available, providing a semi-dynamically updating schedule.

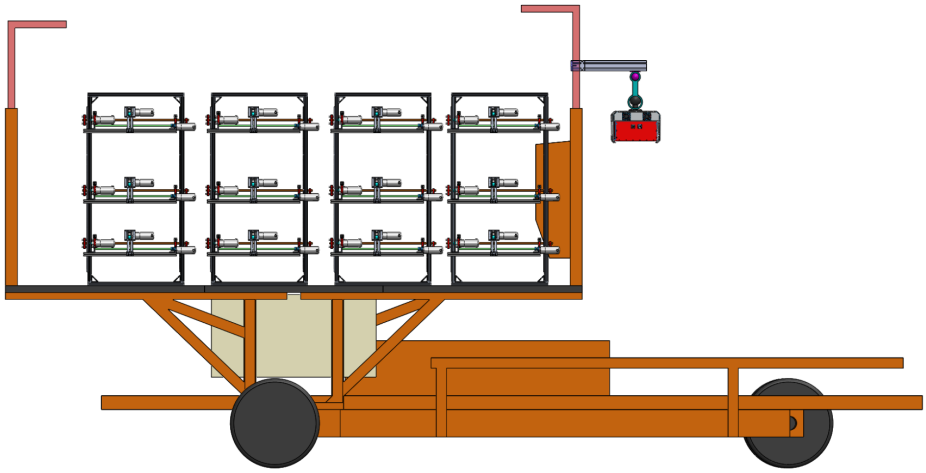


Figure 3: Concept art of the robotic harvester made up of four columns, each with three arms.

# 4 Modeling the fruit harvesting process in an orchard segment

## 4.1 Introduction and approach

This chapter describes a multi-armed robotic harvester model with arms both in series and in parallel, moving at a constant speed along orchard rows. The harvester consists of $C$ number of columns set side-by-side, each with $R$ number of arms (or rows) within each column. The columns of arms will be set on a vehicle that is always in motion. All arms are identical Cartesian arms, with three linear axes which move independently based on trapezoidal velocity profiles. The arms inside a column share the workspace and their vertical positioning is physically constrained, in the vertical direction, only by the arms (or frames) above and below them. Instead of physical limits, software-defined vertical limits will be computed to create rows in which the arms can move freely while avoiding collisions with other arms.

Orchard rows will be broken up into smaller segments to simulate how the harvester may only know the locations of fruits in sight of its cameras. Within a segment, the harvester

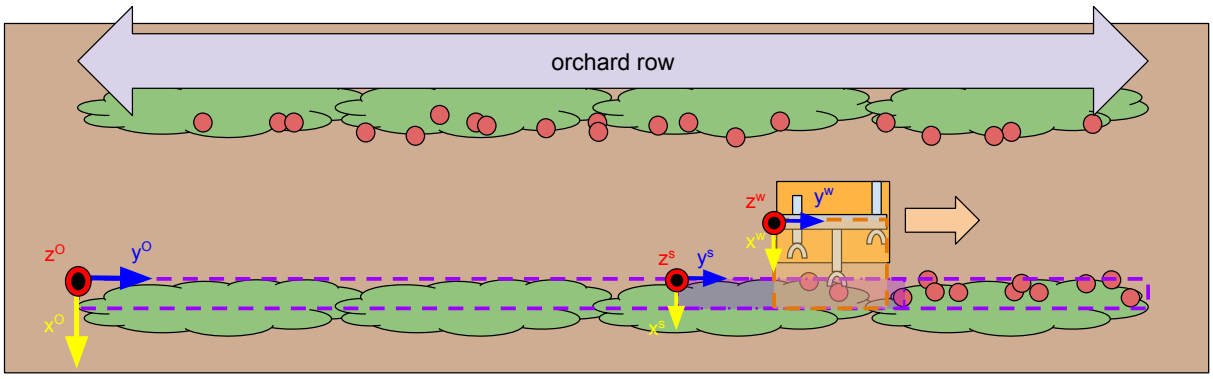



Figure 4: Top-down view of the orchard row frame $O$, segment frame $s$, and harvester's workspace frame $w$ with respect to each other over a harvesting run.

will move with a constant speed, though speeds can vary between segments as needed. The harvester will solve single segments of orchard rows individually. All fruit locations are known and are static within the segment.

## 4.2 Defining the orchard segment and harvester frames

The orchard row's frame will be the global frame, and it will describe the volume of space enclosing all fruits. It is centered at the start of the current orchard row, at the edge of the canopy and on the ground. Fig. 4 shows a top-down view of the frame. The $x$-axis will point into the tree canopies, the $y$-axis will point along the orchard row, and the $z$-axis will point upwards. The orchard frame's bounds in the $x$-axis start at the edge of the canopy and end at the center of the canopy, in the same plane as the tree trunks. In the $y$-axis, the bounds enclose all the fruits between the start and end of the orchard row. Lastly, the $z$-axis bounds will start below the bottom-most fruit and end above the top-most fruit.

It is assumed that the harvester only knows the location of fruits within view of its cameras. Because of this, the orchard frame is broken up into segments which can be scheduled and harvested independently as the system moves forward. Each one will be centered at some $(x^s, y^s, z^s)$ and have the same length $d_s$ along the $y$-axis. The orchard row and the segments

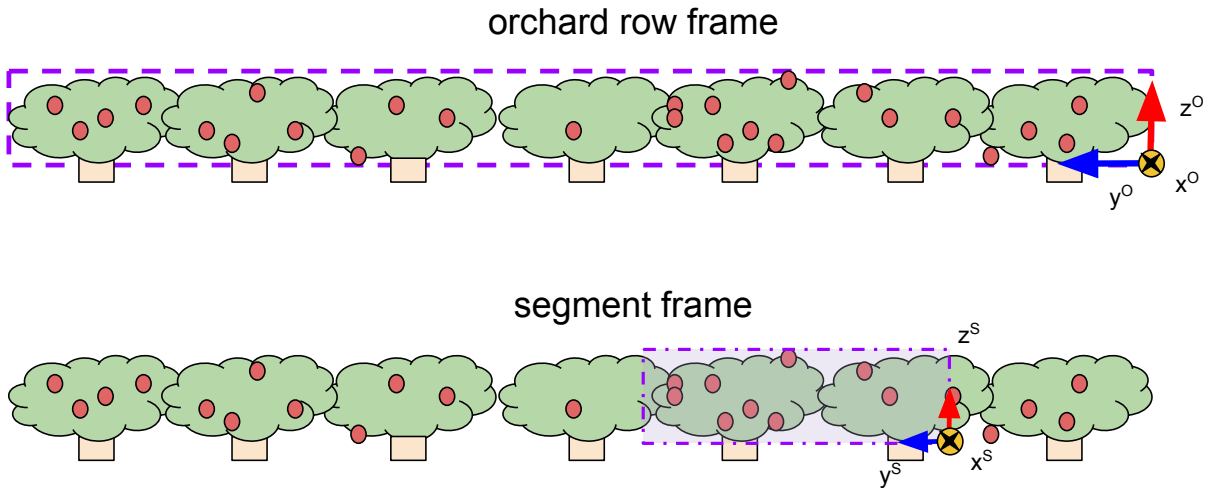



Figure 5: Side view of the orchard row frame and the visualization of what potential segment frames could be chosen within the orchard row frame.

have matching $x$ and $z$-coordinates and bounds.

Because segments are scheduled independent of one another, the harvester's frames are set in reference to the segment frames rather than the orchard row frame. The harvester frame will be the "workspace," or the volume of space with fruits that the system can currently harvest. It is composed of the volume of space in front of the harvester's arms, encompassing the volume of space starting at the fully retracted grippers and ending at the tree trunks, as seen in Figure 6. The $y$ and $z$-coordinates of the frame are set at the back-most column's bottom-back corner (minimum $y$ and $z$-coordinates with respect to the segment frame), and include all the columns, up to the front-most column's front frame. For example, if there is only one column, then the workspace is that one column frame, if there are three columns, the workspace would encompass all three column frames and any space between the columns.

It is assumed that the harvester is far enough from the tree canopies, some distance $w_o$ when retracted, such that any arm will be too far back to hit the tree's branches when moving

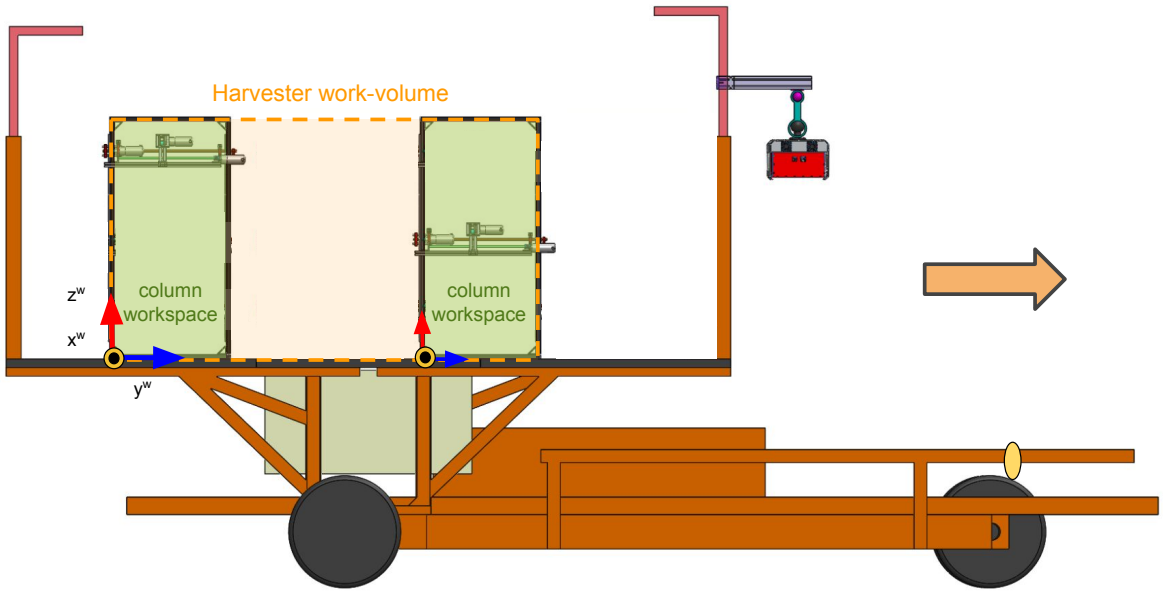



Figure 6: Concept art showing the platform carrying two columns which make up the workspace frame. Each column has a modular arm that can move anywhere within the shaded, green area which indicates the two separate column work areas in the $Y - Z$ plane.

within the $Y - Z$ plane. This means that the $x$-axis of the harvester's frame is centered $w_o$ distance away from the segment frame's center. Fig. 4 gives a visual representation of this. As for the column's $y$-bounds, the system is in constant motion which means that the coordinates of the workspace and column frames change with time along the $y$-axis. As noted, the workspace frame is centered at the bottom, back of the harvester. Assuming that the workspace begins a run at $y$-coordinate $Q$, the workspace frames' $y$-locations can be calculated using

$$y^v(t) = Q + V * t \tag{1}$$

This allows us to set the back-most column's index at 0. Each individual column's frame will, similarly to the harvester, be centered at column's bottom, back corner. To find a column frame's locations at time $t$, we can use

$$y^c_{min}(t) = Q + (c+1)d_c + cd_o + Vt \quad \forall c = 0, 1, ..., C-1 \tag{2}$$

while a column's front frame would be located at

$$y^c_{max}(t) = Q + (c+2)d_c + cd_o + Vt \quad \forall c = 0, 1, ..., C-1 \tag{3}$$

$c$ is the column's index, $d_c$ is the length of a column along the $y$-axis, $V$ is the harvester's speed along the $y$-axis, and $d_o$ is the distance between columns, if any. Lastly, and for ease of calculation, the $z$-bounds of the harvester match the tallest segment's $z$-bounds.

## 4.3 Modeling the harvest of a single fruit with a single arm

Harvesting a single fruit by one arm is modeled as a single Cartesian arm within a single column within a segment, towed by a vehicle with constant speed along the positive $y$-axis. The fruit $i$ is located at ($x_i$, $y_i$, $z_i$) in the segment frame, and is static within this frame. Because the harvester is moving forward with a constant speed, however, the fruit is moving with respect to the workspace frame of the arm. At the start, the arm will be fully retracted with zero initial velocity with reference to the moving vehicle. It will start at the back and center of the column frame.

We assume that, even if the workspace is smaller than the segment frame, the workspace will cover the whole segment frame volume over time, by traveling along the $y$-axis. This means that there exists a time window, $TW_i$, describing when the fruit can be reached and harvested; before and after the time window, the fruit is unreachable. The start and end time for the single fruit is calculated as:

$$TW_i = t_i \in \left[\frac{y_i - (Q + d_c)}{V}, \frac{y_i - Q}{V}\right] \tag{4}$$

$t_i$ is the time of harvest of fruit $i$ and $Q$ is the harvester's starting location on the $y$-axis, and thus the arm's starting $y$-coordinate, with respect to the section frame. The harvester's speed is assumed constant at all points of the run, with no initial acceleration or ending

deceleration period.

The time it takes for an arm to harvest a fruit is based on the distance the arm has to travel to get to the fruit; the farther the fruit's coordinates from the arm's start coordinates, the longer it takes to finish the harvest. In this paper, the time for an arm to harvest a single fruit is named the handling time, or $T_h$, and it is made up of a series of steps, each taking some amount of time based on the distance traveled. An arm takes all steps in the same order when picking any fruit. The steps taken by every arm are:

- "approach": moves within the column frame ($Y - Z$ plane) from the arm's starting coordinates to the fruit's location,
- "extension": extends out from the $Y - Z$ plane to the fruit's coordinate within the canopy,
- "detachment": grabs the fruit after some constant time duration $t_g$,
- "retraction": retracts back to the column frame, mirroring the extension step,
- "transport": (gripper) moves to a drop-off point on one of the edges,
- "drop-off": (gripper) drops off the fruit onto a conveyor over a constant time.

The steps are sequential, each must be completed, and they cannot be stopped and started. While the arm is going through these steps, it is considered "busy" until it has finished the whole process. If the arm has not been assigned to harvest the fruit or has already harvested the fruit, it is considered "idle." As noted in the list, these steps are affected by the subsystem used to harvest the fruit. For example, a mechanical gripper has the added steps where it has to drop off the fruit at a conveyor. A vacuum subsystem would remove the need for the drop-off by having the fruit move directly to a bin using soft ducting.

Depending on the gripper, there are up to four movement steps which are variable in duration

and up to two time steps with a constant duration. Each movement step's time duration values are calculated using a trapezoidal or triangular velocity curve that takes the arm and fruit's $x$-, $y$-, and $z$-coordinates as well as the arm's motor's maximum acceleration $a_{max}$, deceleration $d_{max} \approx a_{max}$, and velocity $v_{max}$ specs. The algorithm is based on Besset and Béarée, 2017's initial steps towards calculating an S-curve velocity trajectory. At each step of the picking cycle, and depending on if the distance traveled is long enough to reach the maximum arm velocity, the optimal velocity profile is calculated as either trapezoidal or triangular, assuming maximum (bang-bang) acceleration step profiles. The minimum distance to reach the maximum arm velocity is

$$\Delta d_{v,max} = \frac{v_{max}^2 - v_0}{2a_{max}} + \frac{v_{max}^2}{2d_{max}}. \tag{5}$$

If $|\Delta d| \geq \Delta d_{v,max}$, the velocity profile is trapezoidal because the arm will reach the maximum velocity during movement in that axis. This determines how the trajectory parameters and variables are calculated. These include the motor's maximum acceleration, $a_r$, velocity $v_r$, and deceleration $d_r$ and the duration of time the motor will accelerate, $T_a$, stay at a constant velocity, $T_v$, and decelerate $T_d$. Defining the sign of the motion as $s = \text{sign}(\Delta d)$, we get the following equations for the trapezoidal profile

$$a_r = s * a_{max};\ d_r = s * d_{max};\ v_r = s * v_{max}; \tag{6}$$

$$T_a = \frac{v_r - v_0}{a_r};\ T_d = \frac{v_r}{d_r};\ T_v = \frac{\Delta d - s * \Delta d_{v,max}}{v_r}. \tag{7}$$

Otherwise, if $|\Delta d| < \Delta d_{v,max}$, the velocity profile becomes triangular which changes the equations to

$$a_r = s * a_{max};\ d_r = s * d_{max};\ v_r = s * \sqrt{\frac{2 * a_{max} * d_{max} * \Delta d - d_{max} * v_0^2}{a_{max} + d_{max}}}; \tag{8}$$

$$T_a = \frac{v_r - v_0}{a_r};\ T_d = \frac{v_r}{d_r};\ T_v = 0. \tag{9}$$

Summing $T_a$, $T_v$, and $T_d$ gives the duration of movement in one axis which can be used to calculate the duration of each movement step.

During the approach phase, the fully retracted arm moves in the $Y$ and $Z$-axes independently, from the arm's starting $y$-coordinate $y_{c,r}$ to the fruit's coordinate at $y_i$, and the starting $z$-coordinate $z_{c,r}$ to the fruit's coordinate $z_i$. The distance traveled by the arm in each axis is:

$$\Delta d_y = |y_i - y_{c,r}| \tag{10}$$

$$\Delta d_z = |z_i - z_{c,r}|. \tag{11}$$

Movement in these two axes happens simultaneously to minimize the approach time. Because this happens, it will take a duration equal to the highest travel time between travel in the $y$-axis or in the $z$-axis, or $T_{yz} = max(T_y, T_z)$. The longer distance in the two axes determines the step's duration. Once the correct $y$ and $z$-coordinates are reached, the arm extends toward the fruit over $T_x$ duration of time. The grasping time, $T_g$ is constant, and is followed by retraction of the arm back to the frame, $T_{x,retract} \approx T_x$. Lastly, if using a gripper, $T_c = T_{yz,conveyor} + T_d$ where the arm moves to the conveyor and takes a constant time to drop the fruit onto the conveyor. The total handling time for a fruit is then calculated using

$$T_{h,gripper} = T_{yz} + 2T_x + T_g + T_c. \tag{12}$$

As noted above, however, a vacuum harvesting subsystem would not require $T_c$, leading to the equation

$$T_{h,vacuum} = T_{yz} + 2T_x + T_g. \tag{13}$$

Thus, if the harvesting process of the fruit starts at a time $t$, it will end at $t + T_h$. As long as

the detachment step ends before $\max(TW)$, the fruit is considered harvested. Otherwise, the fruit will have been missed by the arm.

It is important to note that the gripper's maximum velocity, acceleration and deceleration values are different for each axis, depending on the motors controlling the movement. The motors are chosen so that the specific axis' length over which a motor moves, $max(\Delta d)$, result in $|max(\Delta d)| = \Delta d_{v,max}$, such that the motor never spends much time at the constant velocity section of the trapezoidal velocity profile. It is preferred that the velocity profile always be triangular because the motors are always accelerating or decelerating, which is optimal. The harvester's constant speed will affect the choice in motor specifications for the $y$-axis because the harvester's speed has to be taken into account through the arm's starting speed variable when calculating $\Delta d_{v,max}$.

## 4.4 Modeling the harvest of multiple fruits with a single arm

In this section, the single-arm-to-single-fruit model is expanded to a model describing a single arm harvesting multiple fruits while being towed with constant speed in the positive $y$-direction. $N$ number of harvestable fruits are available to the arm. From the start, the harvester knows the location of all harvestable fruits, and these locations are assumed static, with respect to the segment frame, throughout the run. We need to generate an order in which the arm will harvest these fruits, a sequence $\Sigma$. This sequence has to fit within the duration of time it takes for the system to enter the segment frame, traverse the length between the start and end $y$-coordinates of the segment, and fully exit the segment frame. Fruits can be missed if there are too many for one arm to handle or if the combinations of time windows and mean handling time make it impossible for the arm to harvest all fruits within the allotted time. The total harvesting time can be computed using the single-arm-to-single-fruit method described in the previous section and applied to each successive fruit in $\Sigma$, with one caveat: the approach step's duration is variable based on what fruit was previously harvested by the

arm.

When entering an orchard row or an individual segment, the single column starts a column's length from the start of the canopy. All fruits are unreachable at first. Once the harvester starts moving, fruits enter the column's workspace and the arm can start harvesting. Harvesting stops once the back of the harvester passes the end of the segment, such that all fruits have been passed by the column.

A fruit can only be harvested by the arm if the harvester has moved far enough that the fruit has entered the arm's workspace. We calculate this time using Equation 4 for every fruit. Although the arm can begin the steps to harvest a fruit before the fruit enters its workspace, the detachment step (when the arm comes in physical contact with the fruit), must happen before the fruit passes beyond the arm's reach. The time of harvest must happen at or before $\max(TW_i)$. Furthermore, the arm can only harvest one fruit at a time, during which it is considered "busy." Once a fruit $i$ is harvested, it is not available to be harvested anymore. Furthermore, fruit $i$ cannot be harvested if the arm is still busy with a previous fruit $j$,

$$\max(TW_i) > t_j + T_h^j. \tag{14}$$

The sequence of fruits that the arm follows while it harvests will be the result of an optimization process that maximizes user-defined criteria such as FPE or FPT. Generating a usable $\Sigma$ is highly dependent both on the time window and handling time values of every fruit. $\Sigma$ must be able to describe both the order at which the fruits are harvested and the times at which the fruits are harvested. Therefore, if $P$ fruits were harvested in the run, let $\Sigma = \{\sigma(1), \sigma(2), ..., \sigma(P)\}$ be the fruit picking sequence; for example, if $\sigma(4) = [12, 2.7]$, the fruit with index 12 is the fourth fruit to be harvested at time 2.7 s. We compute FPE for $P$

harvested fruit in the whole segment with $N$ total fruits as:

$$FPE = \frac{P}{N}. \tag{15}$$

FPT is calculated by first computing the overall travel time, $T$, across the segment, from the harvester's starting point at $Q$ to the stopping point at $Q + D$.

$$T = \frac{D}{V}, \tag{16}$$

which leads to

$$FPT = \frac{P}{T}. \tag{17}$$

We calculate the mean handling time as the sum of the individual handling times of picked fruits, divided by the number of picked fruits. Assuming that the fruits harvested have indexes ($k = 1, 2, ..., P$),

$$\overline{T_h} = \sum_{k=1}^{P} T_h^k. \tag{18}$$

## 4.5 Modeling the harvest of multiple fruits with multiple arms

This section presents a model for harvesting a single segment of an orchard row, with $N$ total fruits, using multiple linear arms, and computing the model's associated FPT and FPE. Like in the single arm sections, coordinates are expressed in the segment's frame. This frame will contain $N$ number of fruits, all assumed harvestable. Every fruit $i$ is positioned at its coordinates ($x_i$, $y_i$, $z_i$), with $i = 0, 1, ..., N - 1$ ordered by their $y$-coordiante such that the first fruit encountered by the harvester is $i = 0$, and the last fruit is $i = N - 1$. The segment is harvested as the harvester moves along it with constant speed $V$ over the travel length $D$. However, unlike the single arm sections, the harvester may be made up of multiple columns of arms, requiring multiple column frames contained within the workspace frame. The harvester starts at the $y$-coordinate $Q$, already at the constant speed $V$; the harvester

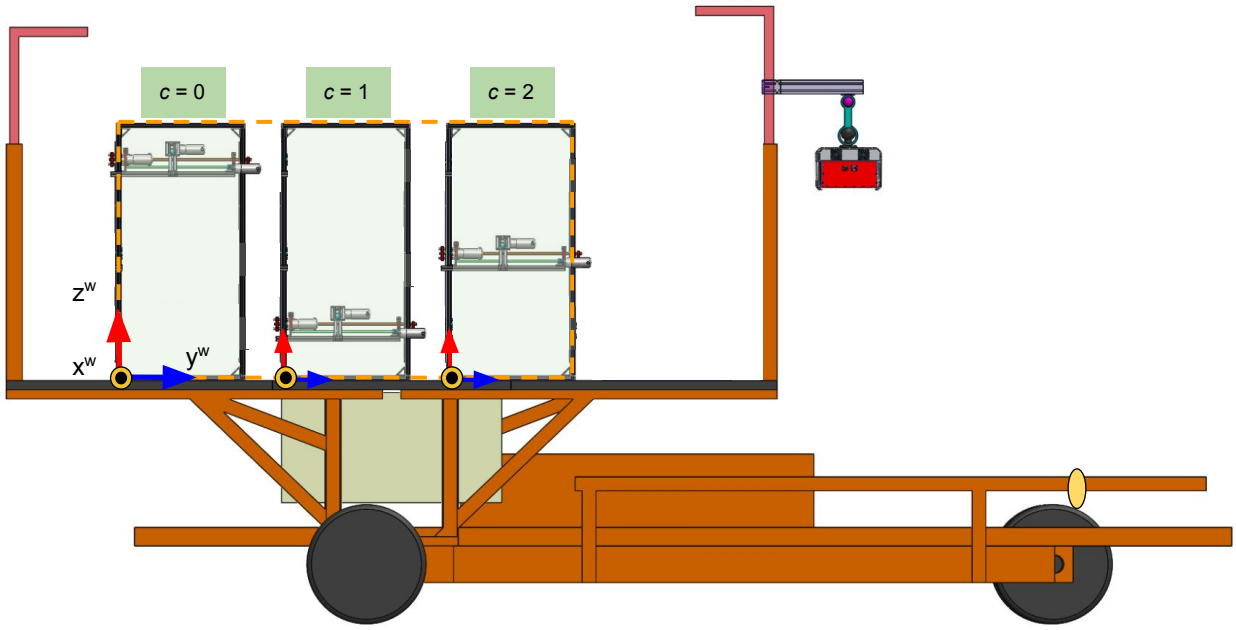



Figure 7: Concept art of a multi-armed robotic harvester made up of three columns, each with one arm, set in a row such that the arms work in series with each other.

will not accelerate or decelerate over a run. A run ends when the back of the harvester reaches the end of the travel length. Within the workspace frame, the grippers may only move in the $Y - Z$ plane within their respective column frame. All columns and all grippers within the columns are identical, and the frames for both columns and grippers are stationary with respect to the moving platform and to each other. Note that when there are multiple arms in a column, the arms are limited to moving in the vertical direction by software-defined $z$-coordinate limits. $c = 0$ is set as the back-most column frame and $c = C - 1$ the front-most column frame, $r = 0$ is set as the bottom row and $r = R - 1$ is set as the top row.

### 4.5.1 Arms working only in series

We modeled the process of harvesting a segment of an orchard row with several linear arms working in series along a single row as follows; the robot is made up of $C$ number of rectangular, upright column frames, each with one arm, set side-by-side on top of a harvesting platform. Since there is only one row, grippers will be bound in the $z$-axis by their respective column frame. The grippers start fully retracted, centered vertically in the column and all

the way next to the back.

An ordered sequence of harvest, $\Sigma^c$, is generated for every arm $c$ at the beginning of a run based on the fruits within the segment's frame. The arms will harvest the fruits as determined by their respective harvesting sequence, in the order and at the times given for each fruit. We let $\Sigma^c = \{\sigma(1), \sigma(2), ..., \sigma(P^c)\}$ be the harvesting sequence for arm $c$, with the order of $\sigma(k)$ indicating the order of harvest, and the values within $\sigma(k)$ indicating the fruit index and the time of harvest. A fruit $i$ can only ever be in one sequence and, once added, will remain within that sequence for the rest of the run. The overall order of harvest, $\Sigma$, is obtained by combining and sorting every column's harvesting sequence, based first on the time of harvest, followed by the arm number; for example, if fruit $i$ and fruit $j$ were harvested at the same time by $c = 2$ and $c = 0$, respectively, then $\Sigma = \sigma(k_j), \sigma(k_i)$, where $\sigma(k_j) = [j, t_j, 0]$, $\sigma(k_i) = [i, t_i, 2]$, and $t_j = t_i$. It should be noted that the $\sigma(k)$s in overall harvesting sequence, $\Sigma$, include the harvesting arm index, the fruit index, and time of harvest.

Because there are multiple columns, the time window calculations have to be expanded; every column has a time window for when each fruit enters and exits the column's workspace. These are calculated by extending Equation 4 to take into account the distance between each column's starting $y$-coordinate—based on its index, $c$, and the cell width, $d_c$—and the fruit's $y_i$ within the section frame.

$$TW_i^c = t_i^{c,r} \in \left[ \frac{y_i - (Q + d_c(c+1) + d_o c)}{V}, \frac{y_i - (Q + d_c c + d_o c)}{V} \right]. \tag{19}$$

Although the arms in the separate columns are independent of each others and not affected by which fruits are in its $\Sigma^c$, the FPE and FPT are strongly affected by the final $\Sigma$. Better combined FPE and FPT results are obtained when the harvest jobs are spread out amongst all the arms, rather than when fewer arms do more work (Edan et al., 1993), indicating that

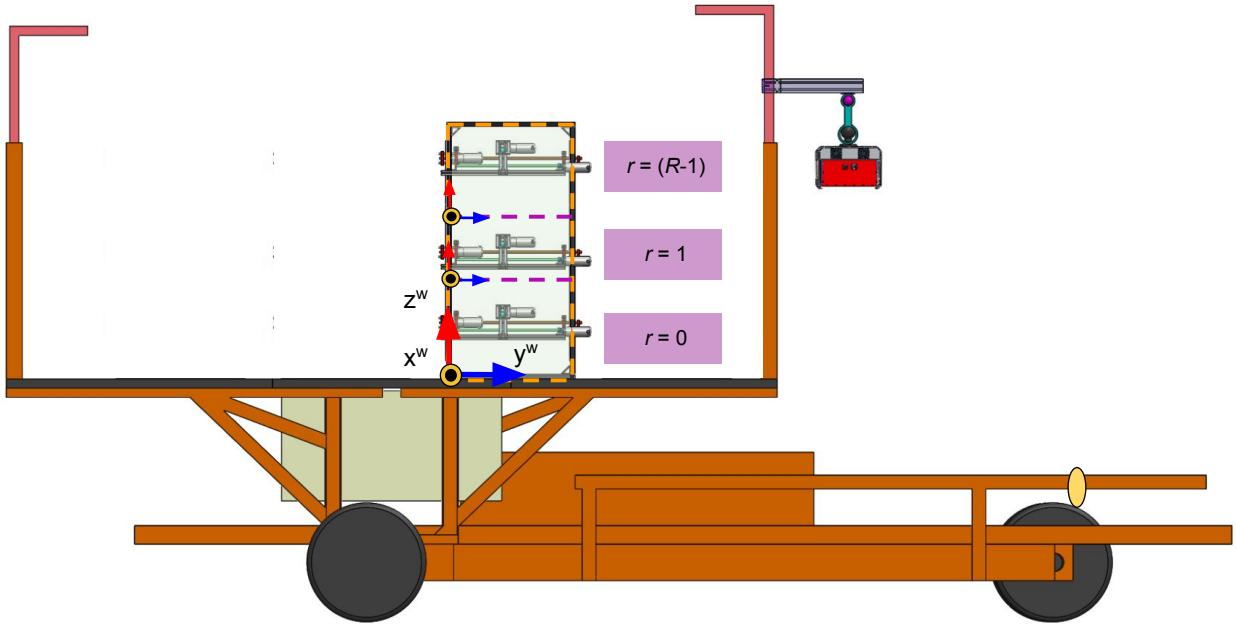



Figure 8: Concept art of a multi-armed robotic harvester made up of one column with three software-defined (purple lines) cells set one-on-top of the other. The arms are limited to their software-determined row indicated by the purple, dashed line.

load balancing is important to get the best results.

We can compute the overall FPE using Equation 15 with the overall number of harvested fruits, $P$, calculated by:

$$P = \sum_{c=0}^{C-1} P^c. \tag{20}$$

For arms working in series, the FPT and handling time are computed with $P$ using Equations 17 and 18.

### 4.5.2 Arms working only in parallel

A harvester with $R$ number of arms working in parallel within one column is modelled as a single upright column with multiple identical Cartesian grippers placed one on top of the other within the column. Because there is only one column, the column frame is the workspace frame. The arms all share the column and are physically limited in their vertical travel only by each other and the top and bottom frame members. Software will be used to

divide the column into $R$ total rectangular rows, each row containing one arm. Dividing the column into rows is important to avoid collisions without resorting to collision avoidance. This division of the column is done once, before starting any harvesting operations in a segment. Because an arm can only be assigned fruits that are within its row boundaries, it cannot generate paths that could lead to collisions with the arms above or below it. However, the software-defined nature of these rows provides flexibility to deal with changes in fruit distributions between segments.

Some important considerations when calculating each arm's vertical limits are as follows:

- all vertical limits in the columns are computed once, at the start of a run, and they stay static for that run,
- the column's bottom is row 0's bottom vertical limit, and the column's top is row $R-1$ top vertical limit,
- grippers take up vertical space within the columns.

This last point is important. If not taken into account when calculating the software-defined $z$-limits, there are likely to be collisions between arms. This height will require that there be some 'dead space' equal to the gripper's height, $h_g$, between arms' $z$-limits to take into account the gripper's geometry. Any fruit with a $z$-coordinate within the column's dead space cannot be harvested by any arm.

Like for the multiple arms in one row, every arm in the column is provided its own ordered harvesting sequence, $\Sigma^r$, which it will follow to harvest fruits in the segment, and which when combined with all other $\Sigma^r$ leads to the overall ordered harvest sequence $\Sigma$. A fruit can only be added to $\Sigma^r$ if the fruit's $z$-coordinate is within an arm's vertical limits. This means that each fruit can only be harvested by one arm during the time window calculated by Equation 4. FPE, FPT, and handling time calculations are equivalent to the calculations

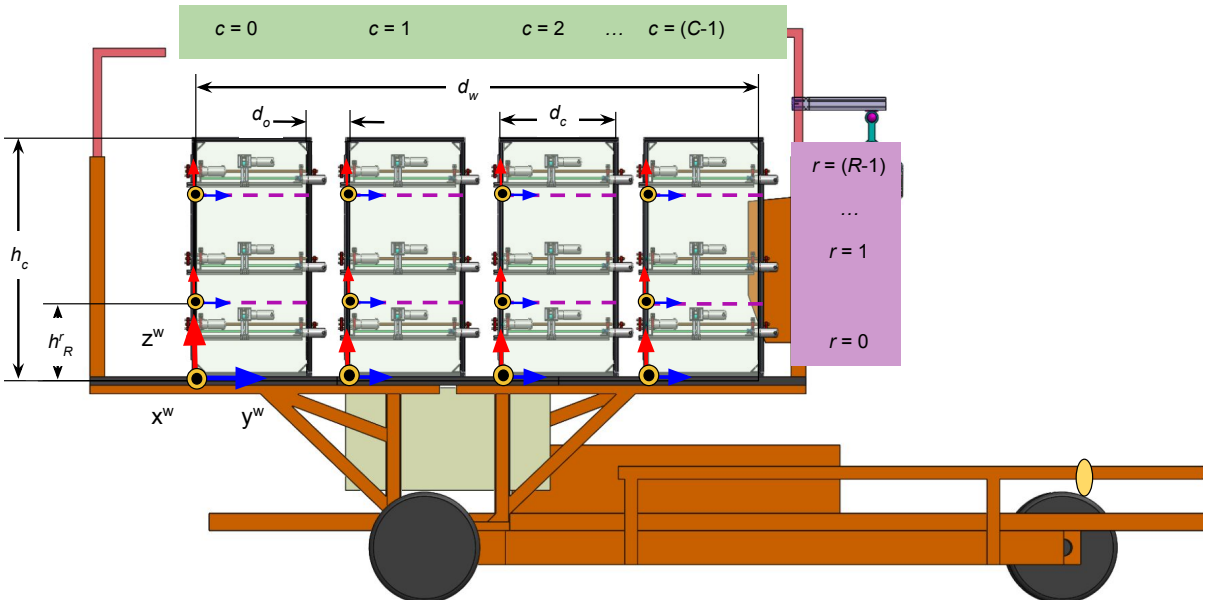



Figure 9: Concept art of the robotic harvester made up of four columns, each with a width $d_c$ and three arms. The distance between the back frame of $c = 0$ to the front frame of $c = (C - 1)$ make up the workspace length $d_w$. The three arms share the whole column as a workspace, with a total height of $h_c$ and software defined row heights $h_r$, increasing the system's flexibility. Each column will have its own frame of reference which is centered at the bottom, back-most point within the frame's work area (shaded in green).

in the single-row Section, except that the sum is over all $P^r$ values.

### 4.5.3 Arms working in series and in parallel

This section discusses the unification of the multi-armed single-column and single-row models into a model where the harvester has multiple columns set side-by-side, each with multiple linear arms (see Fig. 9) mounted within them. Arms in different columns are independent of each other, but within a column the arms share the area enclosed by the frame which requires the use of software $z$-limits to prevent collisions. A harvesting sequence will be generated for each arm, $\Sigma^{c,r} = \{\sigma(1), \sigma(2), ..., \sigma(P^{c,r}), \}$. $\sigma(k)$ contains the fruit index, time of harvest, arm column number, and arm row number, such that $\sigma(k) = [i, t_i, c, r]$. When combining all $\Sigma^{c,r}$ into the unified sequence $\Sigma$, the order is based first on time of harvest, followed by column number, and finally row number. Like in previous multi-armed sections, the total

number of harvested fruits is calculated as the sum of fruits harvested by each arm,

$$P = \sum_{c=0}^{C-1} \sum_{r=0}^{R-1} P^{c,r}. \tag{21}$$

Like in the single column model, the columns have dead space bands between rows which grippers cannot enter to avoid collisions. However, the addition of more columns allows us to change the dead space locations from column to column such that no two dead space bands overlap. The dead space of each additional column will be placed immediately above or below the existing lowest or highest dead space, in an alternating pattern. The resulting pattern means that every fruit can be reached by at least one column of arms. Assuming the index of columns starts at 0, even numbered columns will be offset $-\frac{c}{2}h_g$ while odd columns will be offset $\frac{c+1}{2}h_g$, as visualized in Figure 10. We determine if a column is even or odd if their $c$ index is even or odd. We treat the second column ($c = 1$) as an odd numbered column, but

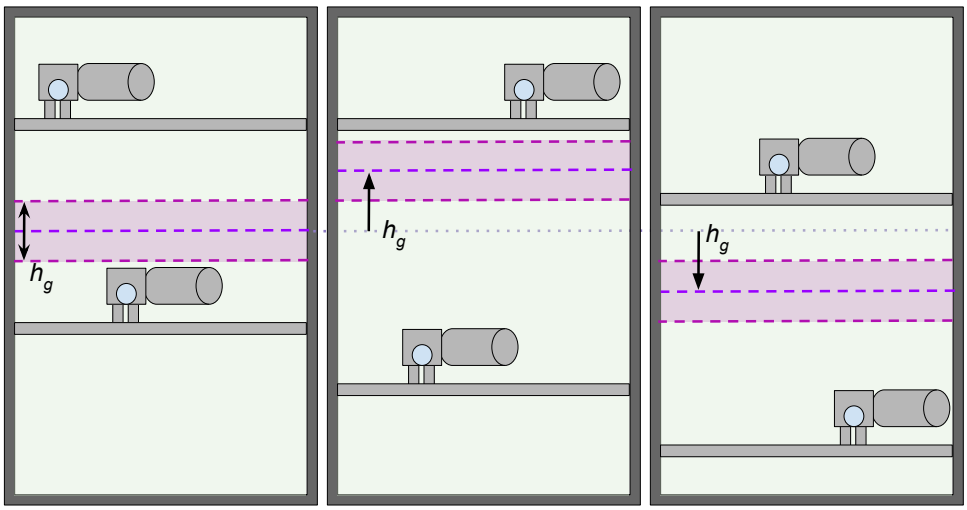



Figure 10: Visualization of the dead space offset when there are three columns.

the offset is calculated from column $c = 0$. The result is that

$$z_{max}^{c_e,r} = z_{min}^{c_e-2,r} - h_g, \ \ \forall c_e = 2, 4, ..., C-1 \tag{22}$$

and

$$z_{min}^{c_o,r} = z_{max}^{c_o-2,r} + h_g, \ \ \forall c_o = 3, 5, ..., C-1. \tag{23}$$

## 4.6 Model considerations due to constant forward speed

For the multi-armed harvester, each fruit is a job that has to be assigned to an agent, or arm. Each fruit has a non-constant handling time that is 'time dependent;' the handling time changes depending on when the arm begins the process of harvesting the fruit. The closer subsequently harvested fruits are, the shorter the handling time for the second fruit because the arm doesn't have to move as far. Furthermore, an arm can only harvest a fruit within its work volume which is determined by the arm's extension capabilities, the front and back column frame locations, the software and frame imposed limits above and below the arm, and the harvester speed. The constant forward speed means that the fruits "move" within this volume in the negative $y$-axis direction—towards the back of the system—as time passes. This means that fruits have a time window during which they can be harvested by a specific arm in a specific column. Equation 19 calculates the time window, $TW_i^c$, or duration during which fruit $i$ is located within column $c$'s work volume

$$TW_i^c = t_i^{c,r} \in \left[ \frac{y_i - (Q + d_c(c+1) + d_o c)}{V}, \frac{y_i - (Q + d_c c + d_o c)}{V} \right]$$

where $y_i$ is fruit $i$'s $y$-coordinate, $Q$ is the workspace's current location in the world frame, $d_c$ is the column's width (along the $y$-axis) and $d_o$ the space between cells, $c$ is the column's index, $V$ is the harvester speed in m/s, and $t_i^{c,r}$ is the time when fruit $i$ is harvested by the arm in column $c$ and row $r$. Time $t_i^{c,r}$ is the end of the "detachment" step of fruit harvesting, the third out of four steps for a vacuum based harvester. The $i$-th fruit's $TW_i^c$ values helps

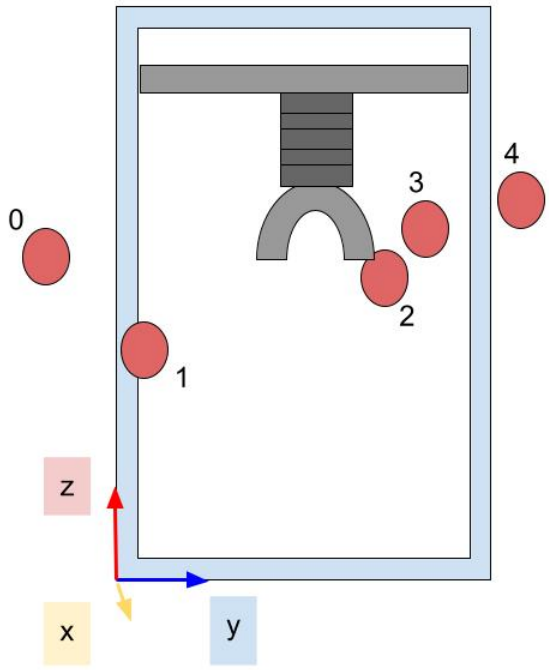



Figure 11: Arm-to-fruit assignment is dependent on each fruits' $y$-coordinates and the handling time for each fruit, since the arm will be busy until the handling time is finished. In this image only fruits number 2, 3, and 4 might be harvestable by the arm. Fruit 0 is past the frame and fruit 1 cannot be reached before it passes the frame. Fruit 3 might be too close to fruit 2 and, if $V$ is very fast, could pass the frame before the arm has finished harvesting fruit 2.

determine when the fruit is within the work volume and so, when fruit $i$ can be physically grabbed or vacuumed by an arm. If $max(TW_i^c) < 0$, then the fruit cannot be harvested by the arm in column $c$ because it has already passed beyond the arm's work volume by passing the back frame. Such a fruit is represented in Fig. 11 as fruit 0. If $min(TW_i^c) < 0$ but $max(TW_i^c) > 0$, the fruit is already within $c$'s work volume. As long as $max(TW_i^c) > 0$, the fruit is still potentially harvestable. In Fig. 11, fruit 1, 2, and 3 all fall in this category, though fruit 1 is likely unpickable because the arm is too close to the back edge and too far away from the arm to be reached in time. In a case where $min(TW_i^c) < 0$ but $max(TW_i^c) > 0$, $min(TW_i^c)$ should be set to zero, since negative time does not make sense. Once an arm $c$ is assigned a fruit $i$ after harvesting fruit $j$, and it begins the process, it must go through all harvesting steps between fruits $j$ and $i$, before the job is considered finished. In Fig. 11,

fruits 2 and 3 are an interesting edge case. If the harvester speed is too fast, then fruit 3 may not be harvestable because the minimum time between $j$ and $i$'s harvest might have fruit 3 passing the back frame column before it can be harvested. Extending Equation 14 for a vacuum gripper, $i$ can only be harvested after $j$ if:

$$\max\left(TW_i^c\right) \geq t_j^c + T_x^j + T_{yz}^{ji} + T_x^i + T_g^i \tag{24}$$

After $t_j^c$, the arm must retract, move to $i$ in the $Y-Z$ plane, extend to $i$, and take time to grab $i$ before $\max\left(TW_i^c\right)$ time is reached. This is further complicated because the handling time encompasses multiple steps, some which can start before $min(TW_i^c)$, or take place beyond $max(TW_i^c)$—movement in the $Y-Z$ plane and the extension in $X$ can be started before the fruit comes into the work volume, while retraction and gripper drop-off movements can be performed even after the fruit would have passed the back frame.

# 5 Multi-objective scheduling maximizing FPT while meeting a minimum FPE

## 5.1 Introduction and approach

This chapter focuses on the development of multi-objective scheduling to obtain the maximum possible FPT while meeting an FPE lower bound when working with multi-armed harvesters. To do this, the harvester first performs load balancing across rows through software-defined row boundaries and then computes the best harvester speed and schedule combination. This process is then extended to whole orchard rows by solving for the best harvester settings and schedule for small segments of the orchard, one at a time.

To load balance the workload between rows, we divide the rows so that every row has approximately the same number of fruits. Based on these rows, the scheduler will find a best combined speed and schedule. None of this is solved dynamically, e.g., in real time. Instead, the system computes a single set of row limits, schedule, and harvester speed for the segment of the orchard row that it can observe. Together, the harvester settings and schedule maximize FPT while meeting a minimum FPE value. Although we prioritize FPE due to its higher sensitivity (see Harrell, 1987), it is important that FPT be as high as possible for the harvester to be useful.

Obtaining the best combination of speed and schedule is complicated because FPE and FPT are at odds with each other, as described in Edan et al., 1993. We tested two scheduling strategies to get the best harvester speed and schedule combination. The first is a greedy, naive strategy where the scheduler loops through a set of discrete speeds, solves the scheduling problem for each velocity, and chooses the schedule and speed combination that meet the requirements. In contrast, the second strategy poses the optimization problem as a dual-objective optimization problem, with the harvester speed as one of the optimization variables.

Finally, we use dynamic planning over a Sliding Planning Window to extend harvesting from a single segment to the whole orchard row. This is necessary for three reasons: the harvester can only "see" fruits in front of it, optimization problems can fail if the problem space is too large, and optimizing the settings and schedule once, for the whole orchard row, would be suboptimal. Not only can the harvester only see a small section of the orchard at any one time, but a whole orchard row has too big of a problem space to solve using MILP, especially as the number of arms increases. Thus, we break up the orchard row into a series of sequential segments, set one after the other, and each segment is solved individually. Segments can overlap one another, such that fruits that were missed or which moved can be re-visited, and horizons are used to provide future knowledge for better results. This has the added benefit that the best row height, vehicle speed, and schedule combination is obtained for each segment. Such a strategy provides even better results than solving for a schedule for the whole orchard row in a single pass.

## 5.2 Using software-defined row limits for load balancing between rows

The flexibility of software-defined $z$-axis limits for the rows is important because fruit distributions are not uniform along the $z$-axis. By defining the limits through software instead of physical frames, the harvester can determine the best location for these limits and react to the distribution while still avoiding the need for collision avoidance. When harvesting an orchard segment, the harvester will calculate the $z$-limits once, at the very start, and keep that configuration for the whole run. One way of doing this is by calculating the $z$-axis limits of rows so that each row contains the same number of fruits as the other rows.

If we assume that all fruits take around the same amount of time to harvest, partitioning the columns into rows with the same number of fruits (by fruits) would lead to workload balancing across the rows. All the rows will take the same amount of time to harvest; if the

workload was unbalanced, instead, the row with the most fruits would require more time to finish harvesting fruits while the row with the fewest fruits would require less time. With unbalanced workloads, any harvester speed becomes suboptimal. FPE suffers if the system goes faster because more fruits are left behind in the slower rows, while FPT decreases if the slower speeds are chosen and many arms in the rows with fewer fruits are left idling. The more extreme the non-uniformity of the fruit distribution, the more important it becomes that the system be able to react to it.

To show the effects of using $z$-axis limits for workload balancing across rows, we compare it setting all rows to have equal heights. The calculations for the naive, equal heights' method are as follows. The bottom limits are calculated using

$$z^r_{top} = \frac{h_c}{R} r + \frac{1}{2} h_g \quad \forall r = 1, 2, ..., R-1 \tag{25}$$

while the top limits are calculated using

$$z^r_{bot} = \frac{h_c}{R} (r+1) - \frac{1}{2} h_g \quad \forall r = 1, 2, ..., R-1. \tag{26}$$

These equations take into account the dead-space caused by the gripper's physical height. Although not shown here, the limits for a cell in column $c$ and row $r$, for both this method and the next, will be slightly different across columns due to offsets calculated to minimize dead-space. This alternating dead-space pattern is described in Chapter 3.

The method to calculate $z$-axis limits so that every row has around the same number of fruits is as follows; the fruits are ordered, based on their $z$-coordinate, into a sequence $M$. The total number of fruits are then divided by the number of rows such that $n_r = floor(N/R)$. Fruits at indexes

$$m(r) = \{(r+1) * n_r\} \quad \forall r = 0, 1, 2, ..., R-1 \tag{27}$$

are the fruits located at the highest point in row $r$, and their $z$-coordinates,

$$M(m(r)) = \{z_{m(0)}, z_{m(1)}, ..., z_{m(R-1)}\}, \tag{28}$$

can be used to determine the top $z$-limits of their respective row, except for the top-most row. However, the row $z$-limits should not be placed exactly on the fruits' coordinates to avoid putting the fruit as the boundary. Instead, the $z$-coordinate used for the $z$-limit will be the average coordinate between $z_{m(0)}$ and $z_{m(0)+1}$. Mathematically, the $z$-limits are calculated using

$$z^r_{top} = \frac{(M(m(r)) + M(m(r)+1))}{2} \quad \forall r = 0, 1, ..., R-2 \tag{29}$$

and

$$z^r_{bot} = z^{r-1}_{top} + h_g \quad \forall r = 1, 2, ..., R-1 \tag{30}$$

## 5.3 Combining harvester speed selection and scheduling to maximize results

Determining the harvester speed is very important because it has a large effect on both FPE and FPT. A good speed helps distribute the fruits among the columns by constraining how many fruits a single arm can harvest during a run; too slow, and the front arms will be busy harvesting all the fruits while the back arms will remain idle. When the harvester's speed is too fast compared to the amount of time it takes an arm to go through all the steps in the picking cycle, the arms will not have enough time to harvest fruits. The "best" speed will be one that provides just enough time that all arms remain busy over the majority of the run, while still having time to harvest most fruits. However, it is too complicated to implement dynamic speed selection that matches changes in fruit density in real-time while providing near-optimal dual-objective scheduling results. Instead, speed selection will be implemented alongside the scheduler. The two will work together to find the best combined harvester speed and schedule to obtain a maximum FPT given a minimum FPE is met.

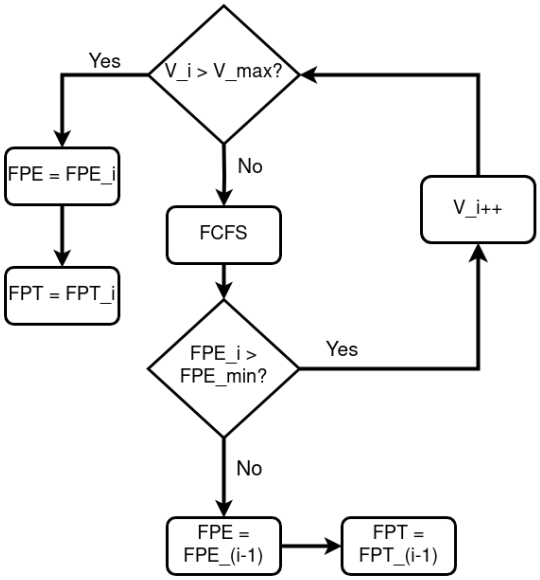



Figure 12: Diagram of the steps taken when using the exhaustive speed search strategy.

Two different strategies are used to find the best speed along with the arm-to-fruit schedule. The first is used as a baseline and is an Exhaustive Speed Search (ESS), a loop that contains the scheduler and which loops through discrete harvester speed values in ascending order until an FPE value of less than 95% is reached, choosing the previous run. Figure 12 shows a diagram of the ESS. This strategy works because as the harvester's speed increases, FPE is a monotonically non-increasing function and FPT is a monotonically non-decreasing function. In other words, FPE will only ever go down while FPT will ever only go up over the range of speeds used in this thesis. Once FPE drops below 95%, no further increase can be seen for FPE, so the previous harvester speed will have the highest possible FPT for an FPE above the required threshold. This is a very easy strategy but greedy and suboptimal. A second way to choose the harvester speed is to include it as a decision variable within an optimization problem. This is seemingly less complicated to implement; however, the optimization's problem-space grows a lot. Although the MILP method can theoretically provide better results, the growing complexity increases the chance of making the problem intractable and

unsolvable with current technologies.

## 5.4 Formulating First Come First Served as the benchmark scheduling algorithm

The FIFO scheduling algorithm known as First Come First Served (FCFS) is used as the benchmark scheduling algorithm. Although uninformed and suboptimal, it is commonly used in multi-armed, pick-and-place industrial applications, see Daoud et al., 2014; Huang et al., 2012; Humbert, Pham, et al., 2015, because it is easy to implement and a schedule can be calculated in real time. In a FCFS algorithm, jobs are sorted by some desired value—arrival time, location, etc.—and then assigned to an agent one-by-one, in ascending index order of the sorted list or array, to be completed by the agent before a new job is assigned. To better fit the robotic harvesting system, however, the base FCFS algorithm requires extensions such as the addition of time windows, multiple agents, and dynamic handling time.

In the multi-armed harvester FCFS algorithm, fruits are sorted by their $y$-coordinate, and the algorithm assigns arms to fruits by the fruit's distance to the column frames. Once the fruits are sorted by their $y$-coordinate, the TW for every fruit is calculated based on a given $V$ value. The arm (or arms) are set as "idle" and assigned to the first fruit that has not been harvested nor assigned to a different arm. Once an arm is assigned to a fruit, it is set as "busy" until it has finished harvesting the fruit and retracted back to $x^V_{min}$; this assumes that every fruit assigned to an arm will be harvested. Harvesting a fruit $i$, assuming a vacuum gripper, is based on Equation 13, with four steps: move within the $Y-Z$ plane to $y_i$ and $z_i$, extension to $x_i$, grabbing, and retraction back to $x^V_{min}$. Let $i$ be the fruit to be harvested, $j$ be the previously harvested fruit, and $c, r$ be the arm doing the harvesting. If there are multiple rows per columns, only fruits with $z$-coordinates between the row's $z$-limits can be assigned to arms in row $r$. FCFS starts by checking if $j+1$ is available to be harvested and in the correct row. If so, then $i = j+1$. Otherwise, the code will loop through the queue of

sorted fruits until it identifies one that is available and in the correct row and that will be $i$. The algorithm then checks if it is possible to harvest $i$ after $j$ by checking if $c, r$ will be become "idle" in time to move to and grab $i$ while it's in $c, r$'s workspace. This is checked by seeing if

$$t_j^{c,r} + T_X^j \leq max(TW_i^c) - (T_{yz}^{ji} + T_X^i + t_{grab}), \tag{31}$$

then $i$ can be harvested after $j$. Otherwise, if

$$t_j^{c,r} + T_X^j > max(TW_i^c) - (T_{yz}^{ji} + T_X^i + t_{grab}), \tag{32}$$

then $i$ cannot be harvested by $c$ and must be harvested by an arm further back, if possible. If $i$ *can* be harvested after $j$, then the algorithm must figure out at what time to start the harvesting process. If the harvest time and retraction to harvest fruit $j$ is less than $i$ fruit's time of entry into the frame's workspace plus all the steps to reach the fruit, or,

$$t_j^{c,r} + T_X^j \leq min(TW_i^c) - (T_{yz}^{ji} + T_X^i), \tag{33}$$

then $i$ is too far forward to start right after finishing the retraction step for $j$. The arm is assumed to move to $i$'s $z$-coordinate and the front frame while it waits, and, at $min(TW_i^c) - T_X^i$, $c$ will extend so that at $min(TW_i^c) + t_{grab}$ the arm can grab the fruit. If instead

$$t_j^{c,r} + T_X^j > min(TW_i^c) - (T_{yz}^{ji} + T_X^i), \tag{34}$$

then $c$ will be busy harvesting $j$ past $min(TW_i^c)$, and arm $c, r$ should start moving towards $i$ immediately.

**Algorithm 1** FCFS arm-to-fruit scheduler

**Require:** List of fruits sorted by $y$-coordinates in ascending order, $TYZ[j,i]$ and $TX[i]$ calculated beforehand. $Y[i]$ and $Z[i]$ fruit $i$'s $y$ and $z$-coordinates, $Q$ in the same frame as $Y[i]$ and $Z[i]$, $V$ in m/s.

**declare:** $busy\_till[R,C]$ : 2D ARRAY of FLOAT
**declare:** $busy\_with[R,C]$ : 2D ARRAY of INT
**declare:** $i\_harvested[N]$ : 1D ARRAY of BINARY

**for** $i = 0,\ N-1$ **do**
  $tw_{s,0} := (Y[i] - (Q + d_c))/V$ ▷ Start, $i$ enters column front
  $tw_{s,C} := (Y[i] - (Q + (C-1) * d_c + (C-2) * d_o))/V$
  $tw_{e,0} := (Y[i] - Q)/V$ ▷ End, $i$ exits column back
  $tw_{e,C} := (Y[i] - (Q + (C-2) * d_c + (C-3) * d_o))/V$
  $tw_s := range(tw_{s,0}, tw_{s,C}, step = -(d_c + d_o)/V_m)$ ▷ All column start times
  $tw_e := range(tw_{e,0}, tw_{e,C}, step = -(d_c + d_o)/V_m)$ ▷ All column end times
  **for** $c = C-1,\ 0$ **do** ▷ Start at the front column $(C-1)$ and move backwards
    **if** $i\_harvested[i] == 1$ **then** ▷ If fruit $i$ already harvested, go to $i+1$
      break
    **end if**
    **for** $r = 0,\ R-1$ **do**
      **if** $Z[i] > z_{bot}[c,r]$ and $Z[i] < z_{top}[c,r]$ **then**
        $j := busy\_with[r,c]$ ▷ Previously harvested fruit by arm $c,r$
        **if** $i > 0$ **then**
          $j2i = TYZ[j,i]$
        **else** ▷ If $i = 0$, $j = 0$ and $TYZ[0,0] = inf$
          $T_y = Vprofile(Q, Y[i])$ ▷ Calc TYZ movement from $Q$ to $i$
          $T_z = Vprofile((z_{top}[c,r] - z_{bot}[c,r])/2, Z[i])$
          $j2i = max(T_y, T_z)$
        **end if**
        **if** $busy\_till[c,r] + j2i + TX[i] + t_{grab} \leq tw_e[c]$ **then** ▷ Mark $i$ as picked
          $i\_harvested[i] := 1$
          **if** $busy\_till[c,r] \leq tw_s - (j2i + TX[i])$ **then** ▷ Wait to start harvesting
            $t := tw_s[c] + t_{grab}$
          **else** ▷ No need to wait, start once not busy
            $t := busy\_till[r,c] + j2i + TX[i] + t_{grab}$
          **end if**
          $busy\_till[r,c] := t + TX[i]$
          $busy\_with[r,c] := i$
          break ▷ Fruit $i$ is done, move to $i+1$
        **end if**
      **end if**
    **end for**
  **end for**
**end for**

## 5.5 Dual-objective scheduling to maximize FPT while meeting a minimum FPE

We hypothesize that the best results will be achieved if the objective function optimizes both FPE and FPT. Results from using FCFS as the scheduling algorithm are suboptimal not only because the algorithm is uninformed, but it also focuses more on FPE. A more optimal way to compute schedules for the harvester involves using Mixed-Integer Linear Programming (MILP), a form of mathematical optimization, and converting the problem into a dual-objective optimization problem. The problem is defined mathematically, including an objective function as well as any constraints, and solved using a commercial solver. More specifically, we will use scalarization to find the best schedule and harvester speed combination, given constraints such as a minimum achieved FPE.

As shown by Mann et al., 2016, the multi-armed harvester problem can be described mathematically by the Time Dependent Team Orienteering Problem with Time Windows (TDTOPTW). This is an extension of the Orienteering Problem, or OP, fist described in Fomin and Lingas, 2002. The goal of the OP is to maximize the total score of an individual that starts at a specified point, visits as many checkpoints (vertices) as possible, and returns to the end point within a given time frame. Vertices can be missed and can have specific scores related to them, so some are offer a greater incentive if chosen as part of the visited points. The orienteering problem by itself is $\mathcal{NP}$-hard. When extended to the TDTOPTW, it is $\mathcal{NP}$-hard and MAX-$\mathcal{SNP}$-hard, as shown by Fomin and Lingas, 2002; Gavalas et al., 2014. Solving small problem instances optimally is time and computationally expensive, while larger problems cannot be solved optimally with current technologies and knowledge. Although there are heuristics developed for the TDTOPTW (see Gavalas et al., 2014; Mann et al., 2016), they can be very situational and must be tested within the specific work. Instead, we will keep the problem size small by breaking the orchard rows into segments and solving each segment independently.

It is important to note that the underlying TDTOPTW MIP formulation used for this project's multi-armed harvester is similar, but not identical, to the formulation in Mann et al., 2016. The time window calculations are changed such that the 0-th column is the back-most column and fruits can start within the system's view window, whereas in Mann et al., 2016 the fruits always start in front of the system's front-most work cell. The equation to calculate the time window is also extended and is written in Equation 19. Multiple rows of arms are a further addition to the MIP formulation. Instead of only having arms in series, one after the other, the system has a grid-like matrix of arms all working together both in series and in parallel. This means that an arm in a row cannot harvest fruits outside their workspace which is limited both in the $y$-axis by the column's front and back frames and in the $z$-axis by software limits or the column's top and bottom frame. It also means that calculating FPE and FPT has to take into account the rows and the columns, as shown in the following Equations 15 and 17,

$$FPE = \frac{1}{N} \sum_{r=0}^{(R-1)} \sum_{c=0}^{(C-1)} \sum_{i=0}^{(N-1)} x_i^{c,r}$$

and

$$FPT = \frac{V}{d_w} \sum_{r=0}^{(R-1)} \sum_{c=0}^{(C-1)} \sum_{i=0}^{(N-1)} x_i^{c,r}.$$

$c = 0, 1, ..., (C-1)$ is the column index, $r = 0, 1, ..., (R-1)$ is the row index, and $i = 0, 1, ..., (N-1)$ the fruit index, with $C$, $R$, and $N$ being the total number of columns, rows, and harvestable fruits, respectively. $x_i^{c,r}$ is a binary value indicating if fruit $i$ is or is not harvested by the arm in column $c$, row $r$.

To make this problem into a dual-objective optimization problem, we use Scalarization. This method, described in Gunantara, 2018, is used to solve for single answers when working with multiple objectives. It requires that "subjective preferences" be provided by a Decision Maker because, when there are multiple, conflicting objectives there usually isn't a single, best

answer; every answer is a trade-off between the objectives. This is true of the multi-armed harvester because FPE and FPT are conflicting objectives; the highest FPE would be at the lowest speeds, the highest FPT would result in too few fruits being harvested. We can use knowledge of the growers' preferences to provide constraints such as a minimum FPE that must be harvested. The simplest way of formulating scalarized multiple-objective problems is through "Classical" solving methods. These are simple extensions to existing single objective optimization problems. Though, it is important to note that the Classical methods are potentially inefficient and only locally non-dominated, Ngatchou et al., 2005.

We will focus on a single method, Goal Programming (see Ngatchou et al., 2005), to solve for both FPE and FPT. It does so by choosing solutions where the FPE and FPT are optimized on given desired values, $FPE^*$ and $FPT^*$. For FPT, these values are determined based on its relationship to the FPE, the number of fruits in a segment $N$, the length traveled over a segment $D$, and the harvester speed $V$,

$$FPT = \frac{N * V * FPE}{D}. \tag{35}$$

All of these additions mean that the problem space is much greater than the problem space for the max FPE formulation. However, they do not require an outer loop to determine the harvester speed and in theory should produce better results. To make them more likely to succeed, initial guesses for the harvester speed are obtained before running the MILP, decreasing the overall problem space. The lower bound for the harvesters' speed is computed based on the mean handling time, $\overline{T_h}$, distance traveled, the number of arms, and the number of available fruits,

$$V_{lb} = \frac{D * C * R}{N * \overline{T_h}}. \tag{36}$$

The upper bound is set to be 5 cm/s more than the lower bound. This is to limit the problem space and increase the chance of finding a solution.

### 5.5.1 Maximizing FPT using Goal Programming

The first scalarized multi-objective formulation which includes the harvester speed as a decision variable is based on Goal Programming (GP), described in Caramia and Dell'Olmo, 2020; Deb et al., 2016; Jones and Tamiz, 2016, which uses an objective function alongside an aspiration level, $\bar{z}_g$, to form a goal $f_g(x) \leq \bar{z}_g$, where $G$ is the number of objectives. Rather than maximize or minimize a value, GP minimizes the deviation, $\delta_g = \bar{z}_g - f_g(x)$ between the objective function value and the aspiration level. The deviation is presented as two positive values, $\delta_g = \delta_g^- - \delta_g^+$, where $\delta_g^-$ is the slack and $\delta_g^+$ is the surplus. GP can be formulated in multiple ways, though this thesis will focus on the well established weighted sums methods. Because the goal is to maximize FPE and FPT while still meeting the goals, the slack will be used in the objective function. The number of objectives will be $G = 2$, where $g = 1$ for FPE and $g = 2$ for FPT. The FPT slack's lower bound will be a negative value to allow it to go below the "desired" minimum. This bound is calculated by finding a maximum theoretical FPT,

$$FPT_{max} = \frac{C * R}{T_{h,min}}, \tag{37}$$

which uses the minimum handling time, $T_{h,min}$, seen empirically. The lower bound for the FPE slack will be set at 0 so that the solver focuses on FPT.

In this formulation FPE and FPT will be defined using Equations 15 and 17, respectively. The aspiration values will be 95% for FPE and the solution to 35 for FPT. To make sure that the units in the objective function are the same, the weight multiplied with the FPT slack will be $D/N$. Furthermore, only the FPT objective will be rewarded if the slack value goes below $FPT^*$. Rewarding FPE as well has the solver prioritize FPE by a large margin producing low FPT results.

$$\text{minimize} \quad \sum_{g=1}^{2} w_g \delta_g^- \tag{38a}$$

subject to

$$\sum_{r=0}^{R-1}\sum_{c=0}^{C-1} x_i^{c,r} \leq 1, \qquad i = 0, \ldots, (N-1) \tag{38b}$$

,

$$t_j^{c,r} + TX_i + TYZ_{ji} + TX_j + t_{grab} - t_i^{c,r} \leq M(2 - x_i^{c,r} - x_j^{c,r}), \quad i, j : Y_j < Y_i, \tag{38c}$$
$$i, j = 0, \ldots, (N-1),$$
$$c = 0, \ldots, (C-1),$$
$$r = 0, \ldots, (R-1),$$

$$t_i^{c,r} - TX_i - t_{grab} \geq -(TX_{max} + t_{grab})(1 - x_i^{c,r}) + 0.0001 x_i^{c,r}, \quad i = 0, \ldots, (N-1), \tag{38d}$$
$$c = 0, \ldots, (C-1),$$
$$r = 0, \ldots, (R-1),$$

$$t_i^{c,r} \geq \min(TWstart_i^c, TWend_i^c), \quad i = 0, \ldots, (N-1), \tag{38e}$$
$$c = 0, \ldots, (C-1),$$
$$r = 0, \ldots, (R-1),$$

$$t_i^{c,r} \leq \max(TWstart_i^c, TWend_i^c), \quad i = 0, \ldots, (N-1), \tag{38f}$$
$$c = 0, \ldots, (C-1),$$
$$r = 0, \ldots, (R-1),$$

$$x_i^{c,r} = 0, \quad i, r : z_i < z_{r,bot}, \tag{38g}$$
$$i, r : z_i > z_{r,top},$$

$$x_i^{c,r} = 0, \quad i : S_i = 2 \tag{38h}$$

,

$$x_i^{c,r} = 0, \quad i, c : \max(TW_i^c) \leq 0, \tag{38i}$$

$$TWstart_i^c * V = y_i - (Q + d_{cell}(c+1)), \quad i = 0, \ldots, (N-1), \tag{38j}$$
$$c = 0, \ldots, (C-1),$$

$$TWend_i^c * V = y_i - (Q + d_{cell} * c), \quad i = 0, \ldots, (N-1), \tag{38k}$$
$$c = 0, \ldots, (C-1),$$

$$TWstart_i^c = \max(0, TWstart_i^c), \quad i = 0, \ldots, (N-1), \tag{38l}$$
$$c = 0, \ldots, (C-1),$$

$$TWend_i^c = \max(0, TWstart_i^c), \quad i = 0, \ldots, (N-1), \tag{38m}$$
$$c = 0, \ldots, (C-1),$$

$$FPE \leq \frac{1}{N} \sum_{r=0}^{(R-1)} \sum_{c=0}^{(C-1)} \sum_{i=0}^{(N-1)} x_i^{c,r} \tag{38n}$$

,

$$FPE \geq FPE^* - \delta_1^-, \tag{38o}$$

$$FPT \leq \frac{V}{d_w} \sum_{r=0}^{(R-1)} \sum_{c=0}^{(C-1)} \sum_{i=0}^{(N-1)} x_i^{c,r} \tag{38p}$$

,

$$FPT \geq FPT^* - \delta_2^- \tag{38q}$$

where

$x_i^{c,r} \in \{1, 0\}$ $\quad$ $t_i^{c,r} \in \Re^+$ $\quad$ $V_{min} \leq V \leq V_{max}$ [m]

$0 \leq TWstart_i^c \leq t_{ub}$ $\quad$ $0 \leq TWend_i^c \leq t_{ub}$ $\quad$ $FPT, FPE \in \Re^+$

$\delta_1^- \geq 0$ $\quad$ $\delta_2^- \geq T_{h,min}$

(38b) means that at most one arm can be assigned to one fruit. (38c) means that time elapsed between pickup of any two fruits reached by the same arm is at least retraction for fruit $i$ plus movement from $i$ to $j$ plus extension to $j$ plus a grabbing time. (38d) means that if the fruit is at or behind the back edge of the column, then it cannot be picked because the arm, at minimum, has to extend out and grab the fruit. (38e) and (38f) mean that the time of harvest for fruit $i$ by the arm in column $c$ and row $r$ has to be within the calculated time windows for that fruit and that column, the smaller value is the start time and the larger value is the end time. (38g) means that if fruit $i$'s $z$-coordinate is outside the arm's z-axis range of movement, do not pick it. (38h) means that if fruit was removed because it has already been scheduled and picked, do not schedule it again. (38i) means that if $TWend$ is negative, the fruit has passed the column k's back edge and cannot be harvested by any arm in that column. (38j) and (38k) calculate the start and end of the time window over which arms in column $c$ can harvest fruit $i$ based on the chosen $V$. (38l) means that a fruit cannot be harvested at a negative time so if the $TWstart$ value is negative, at least part of the time window does not count because the fruit is close to the back edge of the column and the actual $TWstart$ should be at 0 s. (38m) means that if $TWend$ is negative, the $i$th fruit is

completely behind a column's back edge, resulting in an unpickable fruit for that column. Set the start time window value to 0 s so that the solver can run. (38n) means that defines $FPE$ and sets it as an upper bound so that a later soft constraint can set a lower bound for $FPE$. (38o) means that a soft constraint that sets a lower bound to the $FPE$ value that can be violated if it's not possible to get that value. If $FPE_{sl}$ is positive, it shows by how much the constraint was violated. (38p) means that defines $FPT$ and sets it as an upper bound so that a later soft constraint can set a lower bound for $FPT$. (38q) means that a soft constraint that sets a lower bound to the $FPT$ value that can be violated if it's not possible to get that value. If $FPT_{sl}$ is positive, it shows by how much the constraint was violated.

## 5.6 Harvesting whole orchard rows using the Sliding Planning Window model

There is a need to expand the optimization of the harvester from a single segment to a whole row. We propose to implement a Sliding Planning Window (SPW) with a finite time-horizon determined by the harvester's Sliding Window length, $d_p$, travel length, $D$, and harvester speed, $V$. The whole orchard row is broken up into short, sequential, and overlapping segments, similar to those solved in the single segment stage, and scheduled based only on the knowledge of the individual segments. Each segment will make up a "Sliding Window," a representation of the section of the orchard row that the harvester's cameras can "see" and for which it knows the location of all harvestable fruits. As shown in Figure 13, the Sliding Window includes the volume in front of the arms and a "horizon," a small volume at the front of the harvester that locates the fruits that will soon enter the work volume. This provides knowledge of the upcoming fruits which will enter the front column's work volume, improving the system's ability to plan. Thus, a Sliding Window spans the length

$$d_p = d_w + d_h \tag{39}$$

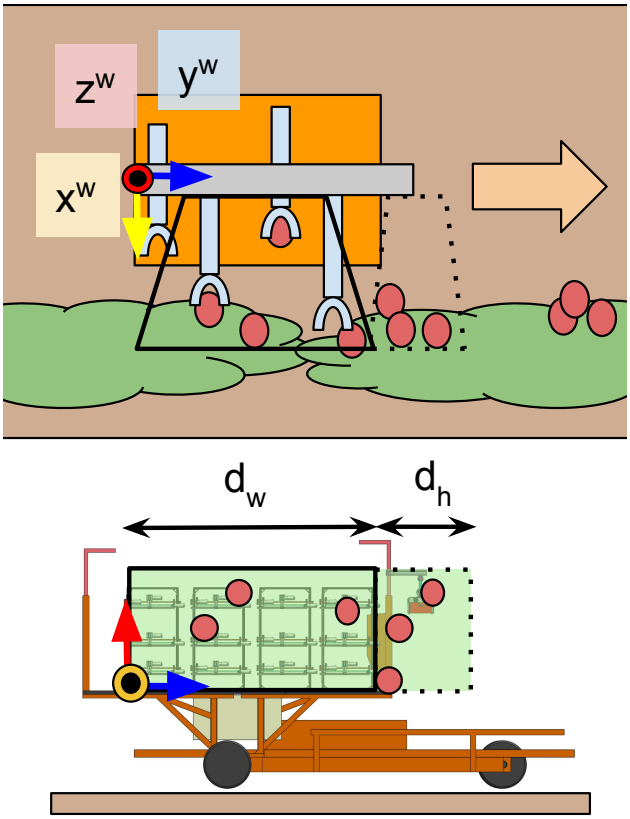



Figure 13: Top-down view and 2-D projections of the Sliding Window, $d_p$, made up of the workspace, $d_w$, and horizon, $d_h$, view windows.

and height $h_c$, which is based on the height of the columns. The Sliding Window can be exploited to create local, static data of the currently harvestable fruits by "freezing" the fruit data within the volume while the harvester moves along its length. It updates this knowledge every time it travels $D$ distance, breaking up the orchard row into smaller problems that are easier to schedule. Each section is treated like an individual optimization problem with a fraction of the total number of fruits. Furthermore, by breaking up the whole orchard row into sections, the system can react to changes in fruit density or distributions in a more dynamic fashion. Row $z$-limits, the harvester speed, and the schedule can be recalculated every traveled distance, $D$, to adjust to more local changes in fruit distribution, density, and locations. We assumed that fruit localization does not include tracking fruits between Sliding Windows because of the high computation time needed to do so.

The sliding window methodology allows the system to optimize for the present and future knowledge but still remain flexible enough that it can react to changes while harvesting. At the start of the orchard row, the harvester is located a vehicle length outside the canopy. Only the horizon will contain any fruits and the harvester has to move into the canopy before the arms start working. This is similar to computing the schedule and harvester speed combination for single segments. Once inside the orchard row, however, most Sliding Windows contain fruit within their work volume and the arms should be busy immediately. Assuming the harvester starts at the $y$-coordinate $Q$, the system goes through the following steps:

1. the origin of the Sliding Window is set at $Q$, such that the system has knowledge of all fruit between the coordinates $Q$ and $Q + d_p$,
2. this knowledge is "frozen" and treated like a static optimization problem, even though the system never stops moving,
3. using the fruit location data, the system determines the boundaries for the rows of grippers,
4. the system computes the schedule and harvester speed combination based on the Sliding Window's fruit locations and gripper row limits,
5. while moving at the chosen harvester speed, the system harvesters the fruits according to the resulting schedule,
6. once the harvester has traveled $D$ distance to $Q + D$, the system "slides" the Sliding Window to the new starting coordinate $Q + D$ and restarts the process.

Figure 14 visualizes how the Sliding Windows move along with the harvester. Importantly, when working in a whole orchard row, the travel length is smaller than the harvester's Sliding Window, unlike the single segment section where the travel length is larger than the length of segment with fruits. The shorter the travel length, the more the system can reschedule and

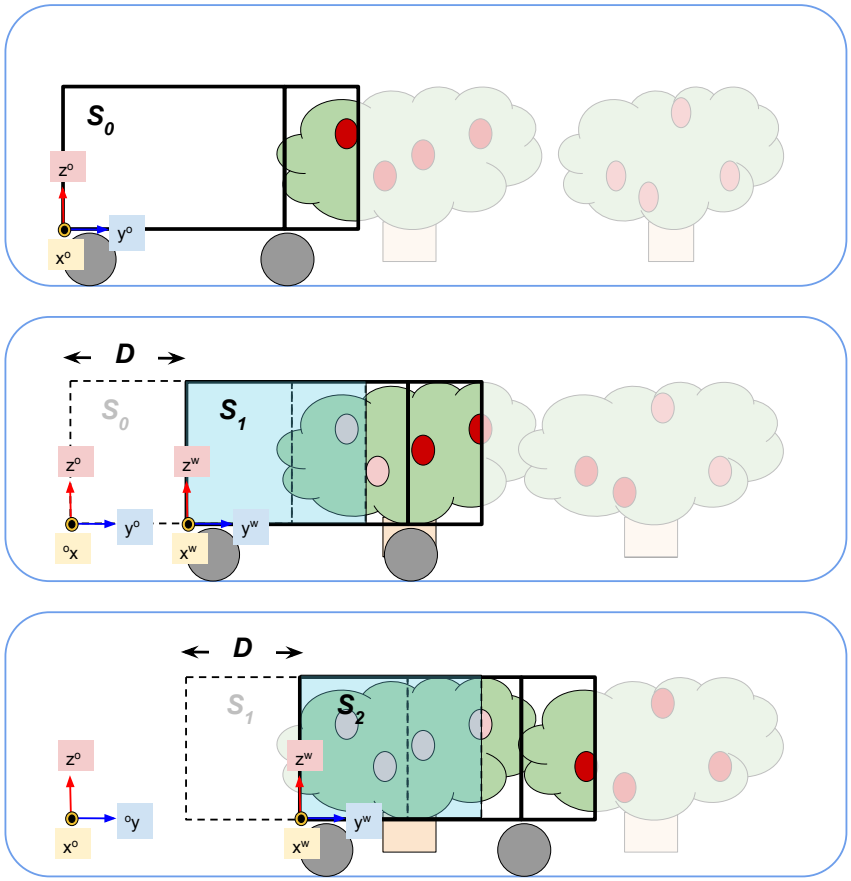



Figure 14: 2D projections of a series of static views updated every time the vehicle moves a given distance $D$. The views are called Sliding Windows, denoted as $S_l$. Each Sliding Window saves the static set of fruits in view of the system at that point in time, for which the system then creates a schedule for harvest. A travel length $D$ is chosen to be smaller than the length of the workspace to have subsequent snapshots overlap (shown in blue) so that that knowledge of that set of fruits can be updated and rescheduled over time.

react to changes in the orchard such as movement in branches or fruits that were previously not located. If it is too short, though, the time spent computing the schedule will drown out everything else because scheduling is a time-consuming process.

Each schedule is computed as if the whole Sliding Window was to be harvested. These schedules should lead to a minimum FPE that matches the minimum FPE desired for the whole orchard row. Once the schedule is finalized, any fruits that will not be reached by the

end of the travel length $D$ are removed from the schedule. This means that the real harvested FPE for that Sliding Window will be lower than the desired FPE, but, by the end of the orchard row, the overall percent of harvested fruits will be closer to the desired FPE than the individual Sliding Window results. If the schedule was instead computed based on the travel length, it would lose the "future" information that the rest of the Sliding Window provides.

# 6 Workspace partitioning and speed selection to improve harvesting speeds

We performed two experiments to test the effects of partitioning the column workspaces into horizontal rows using two methods, alongside speed selection, on scheduling for multi-armed robotic harvesters. Tests were performed through simulations, using digitized distribution data from v-trellised apple orchards. Schedules were computed for 111, 3.5 m segments using FCFS and FCFS with ESS. The resulting FPE and FPT were analyzed to determine the effects of different workspace partitioning methods, harvester speeds, and harvester configurations. The first experiment compared two methods to partition the harvester's columns into rows of arms when the harvester moved along orchard segments at a chosen "best" harvester speed. The second experiment tested how the FPE and FPT obtained using the best harvester speed compared to using fixed harvester speeds for all segments. For this second experiment, the columns were always partitioned so that each row had an equal number of fruits. All tests were run on the same 111 segments and nine harvester configurations shown in Fig. 15.

## 6.1 Harvester settings

The harvester has columns 1 m long and 3.5 m tall, with 0.15 m of space between columns and 0.05 m high dead bands. Grippers will be set at a distance of 0 m from the fruit with the $x$-coordinate closest to the harvester. For all tests, the harvester started at $Q = -3.3$ m with respect to the segment and ended at the $y$-coordinate 3.5 m. Each arm axes had their own maximum acceleration velocity, and deceleration shown in Table 1.

| | max acceleration (m/s$^2$) | max velocity (m/s) | max deceleration (m/s$^2$) |
|---|---|---|---|
| $x$-axis | 2 | 4 | 2 |
| $y$-axis | 1.4 | 2.8 | 1.4 |
| $z$-axis | 1.3 | 2.8 | 1.3 |

Table 1: Maximum acceleration, velocity, and deceleration settings for the three axis of every arm.

(a) 1/1/1 (b) 2/1/2 (c) 3/1/3

(d) 1/2/2 (e) 2/2/4 (f) 3/2/6

(g) 1/3/3 (h) 2/3/6 (i) 3/3/9

Figure 15: Harvester column and row configurations, denoted as C/R/C*R, used in the experiments. Configurations range from one column with one row to three columns each with three rows. The dashed line shows potential locations of software-defined $z$-limits.

## 6.2 Digitizing apple locations

On October 13, 2022, a field experiment was conducted at an orchard in Lodi, California, USA (38.075018, -121.182455). The apples were v-trellised Pink Lady apples which had already undergone the first of two harvest passes. The experiment involved using a cart with a mast and a camera system consisting of four cameras. However, only one camera was used for this research. The cart was moved inside the orchard to capture the position of the fruit in the apple canopy. The camera used in the experiment was the Realsense D435i, recorded at a resolution 1280x720, 30 fps, and with portrait orientation. The images were captured using the Robot Operating System (ROS), a meta-operative system.

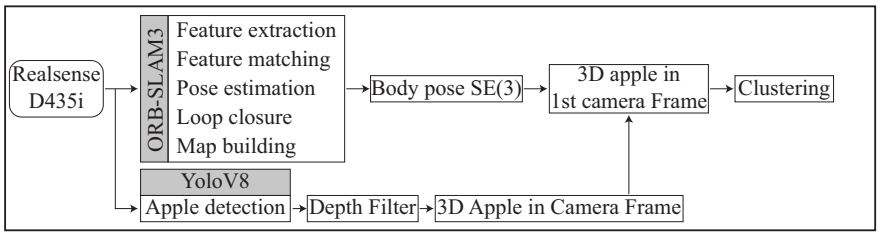



Figure 16: Diagram showing the mapping of apples to digitize their location in 3D space.

A diagram of the mapping of the apples is shown in Figure 16. The image sequence is processed by ORB-SLAM3, which is an established approach for simultaneous localization and mapping (Campos et al., 2021). The resulting camera pose in SE(3) is used to convert the apple detected in the camera frame into a common frame (R0, t0). In addition, the image is also used to detect the apples using YoloV8 Jocher et al., 2023, and the dataset used to train the object detector was published in Villacrés et al., 2023. Once the apples are detected, the center of the bounding boxes is converted into a 3D camera frame using the intrinsic camera parameters and a depth map from the Realsense camera.

To avoid possible detection of apples belonging to trees in other rows, all detections with a distance more significant than 2 m are filtered out. The 3D center apples are then converted using the body pose SE(3) obtained with ORB-SLAM, thus giving them a common reference frame. Finally, clustering-based Density-based spatial clustering is applied to mitigate duplicate apple detections.

## 6.3 Comparing workspace partitioning methods with best speed

This experiment compares two workspace partitioning method described in Section 5.2 and their effect on FPT and FPE when the harvester is moving at the best speed. The first partitioning method divides the workspace, so all rows are equal in height (partition by height). The second method divides the workspace so that all rows contain an equal amount of fruits (partition by fruits) and, thus, more equal workloads across rows. Because fruit

distributions are non-uniform, partitioning the columns of arms by height and by fruits would intuitively produce different results. Partitioning by fruits should react to the non-uniformity while by height does not. Thus, we hypothesize that partition by fruit will produce better FPT results for configurations with multiple rows of arms. FPE should be similar for both partition methods because ESS chooses speeds that lead to schedules that harvest at least 95% of fruits in a segment. We test these methods on the nine harvester configurations shown in Fig. 15.

In order to evaluate if there is a difference between partitioning the columns based on fruits and partitioning based on height, we tested the changes to FPE and FPT of the two methods and different harvester configuration. We use Welch's t-test to compare the mean FPE and FPT obtained in each experiment, with the null hypotheses being that the mean of both partitioning methods does not differ for any one harvester configuration. A 95% confidence level is used for all t-tests. To determine the direction of possible difference in mean, the spread of the FPE and FPT are plotted in a box plot and analyzed visually.

## 6.4 Improving combined FPE and FPT by determining a "best" harvester speed

We hypothesize that determining the best harvester speed for each orchard row segment leads to a higher combined mean FPT and mean FPE output, compared to setting all segment harvester speeds to be the same. To determine the "best" harvester speed for individual segments, we use FCFS with ESS, where the code loops through a set of discrete harvester speed values. The schedule in the loop that results in the last FPE above 95% is used for that segment. Six fixed speeds, $V = 1, 5, 10, 15, 20, 100$ cm/s, are used as baseline speeds while the discrete speed values are $V_{ess} = 1, 2, 3, ..., 100$ cm/s. Once a speed is chosen, that value is constant along the whole segment.

V = best cm/s, 95% confidence level

| | FPE | | FPT | |
|---|---|---|---|---|
| C/R/C*R | p-value | Decision | p-value | Decision |
| 1/1/1 | 1.00 | Accept the null | 1.00 | Accept the null |
| 1/2/2 | 0.66 | Accept the null | 0.00 | Reject the null |
| 2/1/2 | 1.00 | Accept the null | 1.00 | Accept the null |
| 1/3/3 | 0.24 | Accept the null | 0.00 | Reject the null |
| 3/1/3 | 1.00 | Accept the null | 1.00 | Accept the null |
| 2/2/4 | 0.66 | Accept the null | 0.00 | Reject the null |
| 2/3/6 | 0.63 | Accept the null | 0.00 | Reject the null |
| 3/2/6 | 0.96 | Accept the null | 0.00 | Reject the null |
| 3/3/9 | 0.81 | Accept the null | 0.00 | Reject the null |

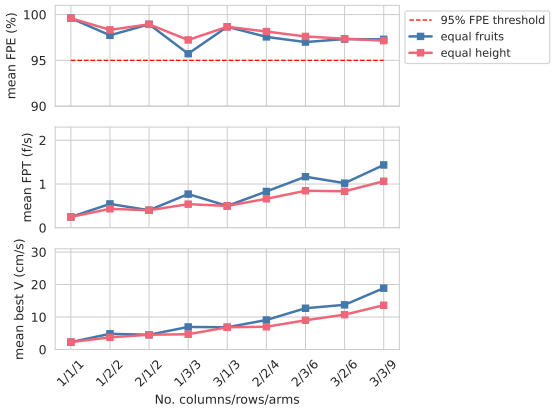
Table 2: Results of the t-test comparing the FPE and FPT means of workspace partitioning by fruit versus by height for all configurations at V = best cm/s.

Like in the experiment comparing column partitioning methods, the harvester speeds are tested on 111, 3.5 m orchard row segments. The five baseline speeds and the best speeds are all tested on the same nine $C/R/C*R$ configurations. All runs have the arm workspaces partitioned by fruits. The mean FPE and mean FPT of each harvester speed and configuration are plotted to help visualize the effects of the speeds on the means.

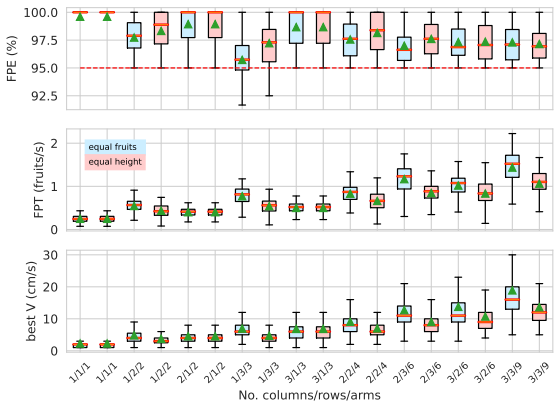
## 6.5 Experiment results and discussion

### 6.5.1 Comparing workspace partitioning method results when using the best harvester speeds

As the number of arms increased, we saw that the mean FPT generally rose while the mean FPE remained stable above 95%. For all configurations with more than one row of arms, the means of the two partition methods were likely not the same (Confidence level of 95%). Visually, both plots in Fig. 17 show that partition by fruit obtained a higher mean FPT. The largest difference was seen for configuration 3/3/9, where partition by fruit achieved a mean FPT of 1.374 fruits/s while partition by height achieved 1.049 fruits/s. Compared to the FPT obtained by both partition methods for configuration 1/1/1, at 0.247 fruits/s, partition by fruit has a gain of 0.618 fruits/s per arm, while partition by height gets a gain of 0.472 fruits/s per arm. For configurations with the same number of arms, more rows performed better




(a) Line plots comparing the means

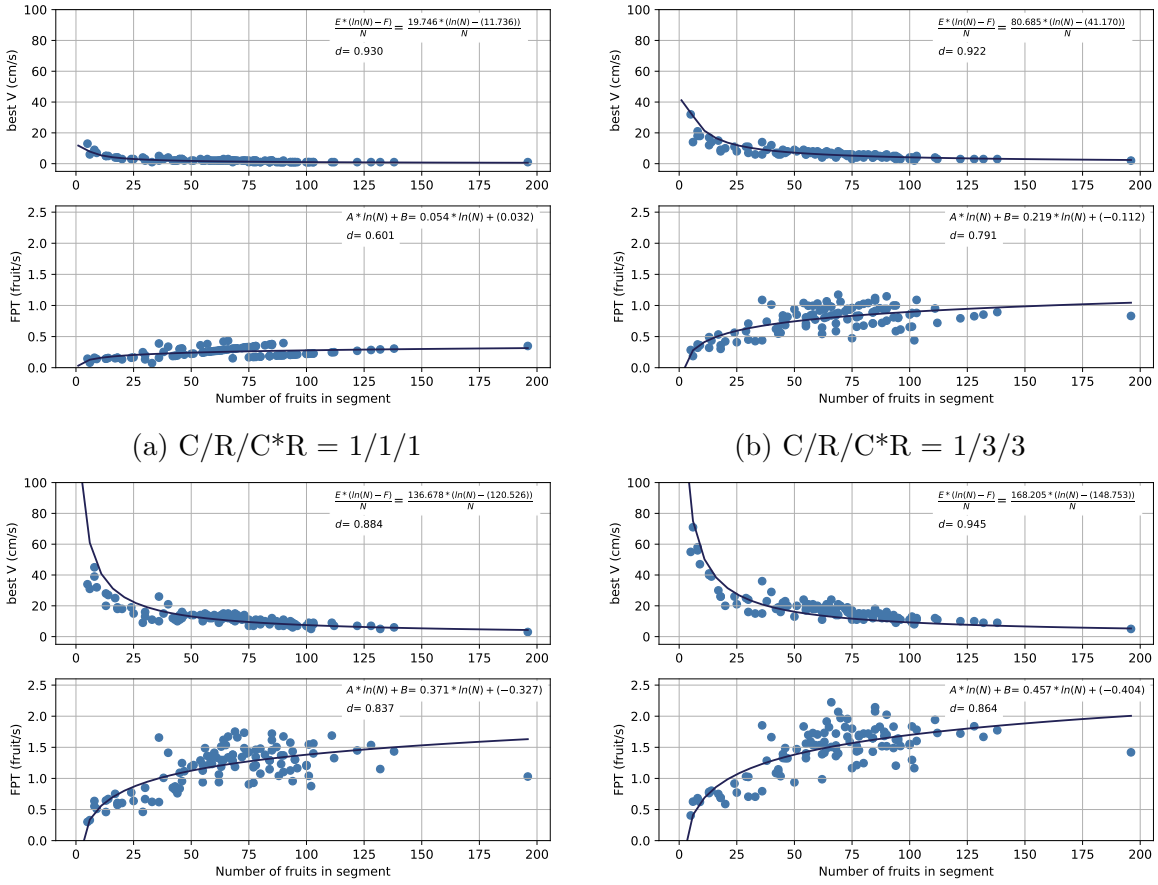



(b) Box plots comparing the spread

Figure 17: Plots comparing the results of partitioning workspaces by fruit vs. by height for 111, 3.5 m segments of digitized apples. The plots show the FPE, FPT, and the chosen 'best' harvester speed for nine harvester column and row configurations. In the box plots, the means are indicated by green triangles.

than more columns, as shown by dips in the mean FPT in Fig. 17a. The columns are 3.5 m high, and it takes the arms a long time to move up and down that length. Furthermore, the left-to-right movement of the arms within the column is aided by the harvester's forward speed.

For the 1/2/2 and 1/3/3 harvester configurations, a drop is seen in FPE. These are the single column harvester configurations which means that any fruits within the dead-space bands between rows cannot be harvested. As described in Section 3.5.3, when there are more columns, the dead-bands for all columns are offset so that all fruits are harvestable. Testing showed that for 1/3/3, partition by fruits had, on average, 1.25 fruits within the dead bands, though the max was 5 fruits, while partition by height had on average 0.75 fruits in the dead-band with a max of 2 fruit. On the other hand, harvester configurations 2/1/2 and 3/1/3 saw a drop in FPT. As single row configurations, the arms had to harvest fruits along the whole column, potentially having to move 3.5 m up and down between fruits. This led to schedules with slower harvester speeds to make sure the 95% FPE threshold was met.




(a) C/R/C*R = 1/1/1

(b) C/R/C*R = 1/3/3

(c) C/R/C*R = 2/3/6

(d) C/R/C*R = 3/3/9

Figure 18: Scatter plots showing the relationship between the number of fruits in a segment and both the best harvester speed and FPT when the rows are partitioned by fruits. Each point represents the results from running the scheduler on one 3.5 m segment out of 111 of these segments. Harvester speed is chosen with ESS to maximize FPT while achieving a minimum FPE of 95%.

The relationships between the number of fruits in each segment and the FPT and harvester speed, when the workspaces are partitioned by fruits, can be visualized in Fig. 18. These plots showed that the harvester speeds trended down while FPT trended up as the number of fruits increased until reaching a steady state when there are too many fruits for the arms to handle. The harvester speed's maximum value was limited by its relationship to the number of arms, the distance traveled, the number of fruits, and the handling time of fruits defined by Equation 36. In the plots in Fig. 18, these were constant parameters, except for the number of fruits. Because of this, the harvester speed followed a curve related to $1/N$, as

$N$ increased. This affected FPT which is defined by Equation 35. The limits on the speed meant that FPT remained low when there were few fruits. As the number of fruits increased, FPT increased, though very slowly, as described in Equation 37. This indicated that FPT followed a logarithmic curve with the form $A * ln(N) + B$. Using the Index of Agreement error metric (Willmott, 1981), we saw that the fit for the logarithmic curve was a bit low, perhaps due to the discrete nature of the number of fruits and the harvester speeds. The curve for the harvester speed was found by setting Equation 35 equal to FPT's logarithmic equation and solving for the harvester speed. Thus,

$$V = D\frac{E\ln(N) + F}{N}, \tag{40}$$

where $E = \frac{D*A}{FPE}$ and $F = \frac{D*B}{FPE}$.

It is important to note that the fruit localization data was obtained after a first harvest had already taken place. This means that the tests are based on lower densities than the harvester would be expected to work on.

### 6.5.2 Effect on FPE and FPT caused by fixed versus best harvester speeds

This experiment tested how choosing the best harvester speed for individual 3.5 m orchard segments compared to using six baseline speeds on every segment. The harvester workspaces were partitioned by fruit for all tests. Fig. 19 showed that low baseline speeds had high mean FPE and low mean FPT. Higher speeds had the opposite problem. The highest fixed speed, 100 cm/s, had the lowest FPE and a lower FPT than the best speed, as well as 5, 10, and 20 cm/s. This indicated that after a speed threshold, the harvester picks too few fruit to cancel out the gains in FPT because the speed is too high. In contrast, choosing the best speed led to a mean FPE above 95% while the mean FPT remained high. The best speed matched or exceeded the mean FPT of the following fixed speeds: 1, 5, 10, and 100 cm/s.

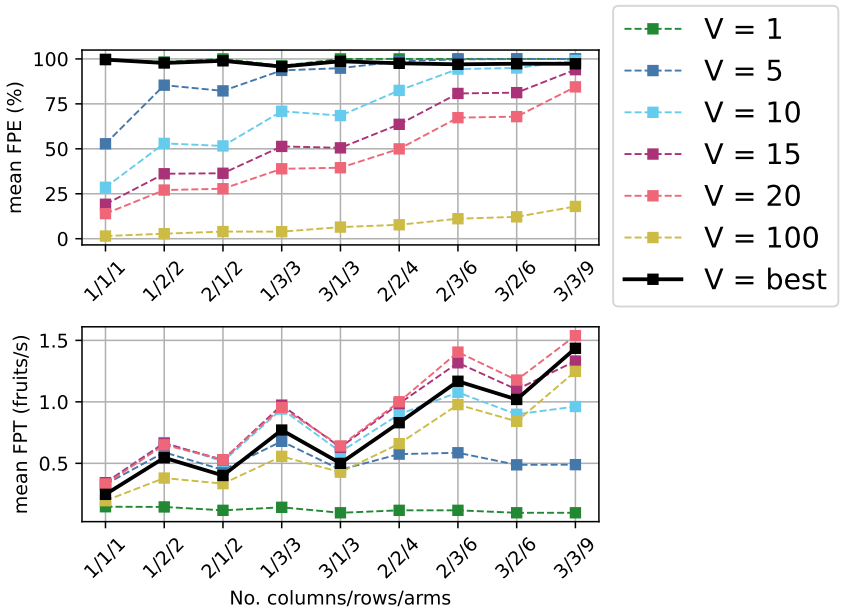



Figure 19: Plot comparing the mean FPE and mean FPT when using fixed speeds compared to using the best speed for each segment. Scheduling performed with FCFS with ESS. Results are shown for five baseline speeds, $V = 1, 5, 10, 15, 20, 100$ cm/s and the best speed for all nine harvester configurations.

This meant that, although finding the best speed never resulted in the highest FPE or FPT, it led to the best combined FPE and FPT.

# 7 Evaluating scheduling strategies for multi-armed fruit harvesters

In this chapter we examine if a dual-objective MILP formulation leads to higher FPT, while harvesting 95% of fruits, compared to FCFS with ESS. Both scheduling strategies were tested on four harvester configurations over 35, 3.5 m segments of digitized apple data. Harvester settings for all tests are the same as described in Section 5.1. The resulting FPE, FPT, and harvester speeds for each scheduling strategy were compared and analyzed. To test if either had an effect on workload balancing, we plotted the percent of time each arm spent idle versus harvesting over a single, 72 fruit segment. Finally, the running time of the two algorithms was compared. MILP is known to be time-consuming which can be problematic on systems that need results quickly. When solving single segments, this is not a problem; however, once outside of simulations, a harvester's FPT could be impacted if it takes too long calculating schedules to harvest fruits in real orchards. Thus, the final running time for each algorithm could be very important.

## 7.1 Comparing scheduling results between FCFS with ESS and dual-objective MILP

This experiment compares the results obtained using FCFS with ESS (described in Section 4.4) and a dual objective MILP formulation based on Goal Programming (described in Section 4.5.2). Both scheduling algorithms attempt to maximize FPT while obtaining a minimum FPE of 95%. Although the 95% value is not a hard bound, the algorithm must be able to reliably meet the given threshold to be useful. FCFS is an uninformed task partitioning algorithm, which we hypothesize will lead to worse results than the MILP formulations, though it will probably compute them faster. In fact, when running the MILP algorithm with a range of possible harvester speeds between 1 and 100 cm/s, the algorithm was unable to compute solutions. This required that the range of possible speeds be decreased by providing

an upper and lower bounds on the speed, based on Equation 36, with the mean handling time empirically computed as 2.75 s/fruit. FCFS with ESS could pick a harvester speed between 1 to 100 cm/s with discrete time intervals of 1 cm/s. In this chpater we will do a performance evaluation to compare the difference between the two algorithms.

In order to evaluate if there is a difference between FCFS with ESS and the MILP algorithm, we tested how the FPE, FPT, and the running time differed for the two algorithms for four harvester configurations. All experiment configurations are tested on 6, 10 m digitized fruit distributions divided into segments of 3.5 m lengths. This leads to a total of 35 usable segments, where a segment has at least twenty fruit. We use Welch's t-test to test if the algorithms' FPE, FPT, and running time means differ for each harvester configuration. A 95% confidence level is used for all t-tests. The null hypothesis we use for all tests is that the means do not differ. A box plot is used to visualize the spread of FPE, FPT, and the running time, for each method and each configuration. If the t-test indicates a difference between means, the box plot is used to determine the direction.

Lastly, we will also examine the percent of time each arm spends harvesting compared to idle for both scheduling strategies. The total time being examined includes the time that the harvester spends entering and exiting each segment, increasing the idle time by a workspace length for every arm. The harvester will move a total of 6.8 m in each run. The time harvesting is separated into four states, moving in the $Y - Z$ plane, extending, grabbing, and retracting. These results can be used to visually examine if either scheduling strategy provides workload balancing across the columns.

### 7.1.1 Important Gurobi settings

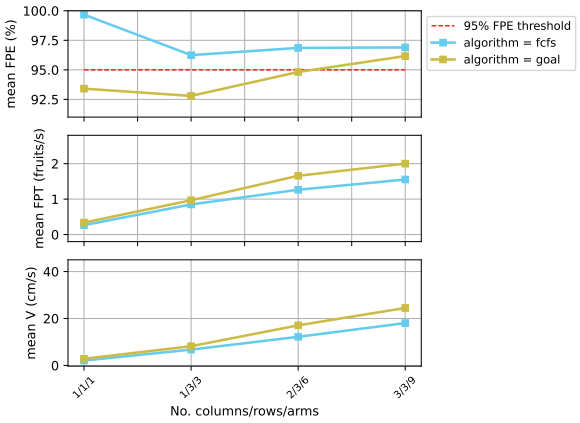
Solving MILP problems requires a dedicated solver. We used a commercial solver Gurobi Optimizer version 10.0.3 build v10.0.3rc0 for this job. All tests were computed on an AMD

Ryzen 9 7950X 16-Core Processor with 16 physical cores and 32 logical processors, allowing the use of up to 32 threads. This included the FCFS runs.

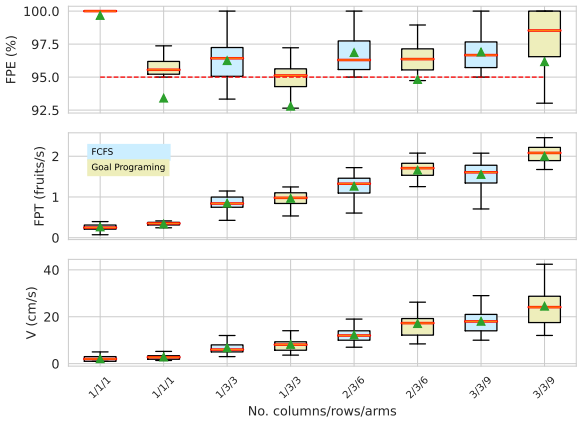
Because the dual-objective formulation is non-convex, runs had the NonConvex variable set to 2. Settings for every run on Gurobi included a 10 min solving time limit (TimeLimit), with the initial 35 s solved using the Gurobi NoRel heuristic (NoRelHeurTime). This timer would be too long to be realistic on a moving harvester, however, it allows most runs to succeed and provide results. As this indicates, MILP is computationally expensive and slow, especially as the number of variables increase. However, improvements to the hardware and formulations could lead lower the computation time needed while still providing better results than naive algorithms such as FCFS. If the time-limit was reached before the MIP Gap reached the default ($1E-4$), the last solution found was used, which means optimality was not always reached.

## 7.2 Experiment results and discussion

We expected that using MILP to solve for harvesting schedules would result in better FPT than using FCFS. Looking at Fig. 20a, results varied. Harvester configurations 2/3/6 and 3/3/9 showed a difference between both scheduling strategies, but it seemed that configurations 1/1/1 and 1/3/3 did not. However, the t-test results are shown in Table 3 indicate that there was a difference between scheduling strategies for all configurations. 1/1/1 and 1/3/3 resulted in mean FPE values lower than 95%, however they differed by very little, 2.5% at most. For these tests, the Goal Programming bounds on FPE did not reward solutions that went above the desired value for FPE, while the bounds for FPT did provide rewards for going above the desired FPT value. This was done in a bid to maximize FPT while keeping FPE close or above the desired value. Changing the bounds to reward FPE for going above 95% would likely remove this issue, though the final mean FPT for 1/1/1 and 1/3/3 might drop to compensate for the difference. We believe that one and three arms were close to their




(a) Line plots comparing the means




(b) Box plots comparing the spread

Figure 20: Plots comparing the spread in FPE, FPT, the best harvester speed when using FCFS with ESS versus Goal Programming to maximize the FPT while harvesting 95% of fruits. 35, 3.5 m segments of digitized apples were tested for each scheduling strategy and four harvester configurations.

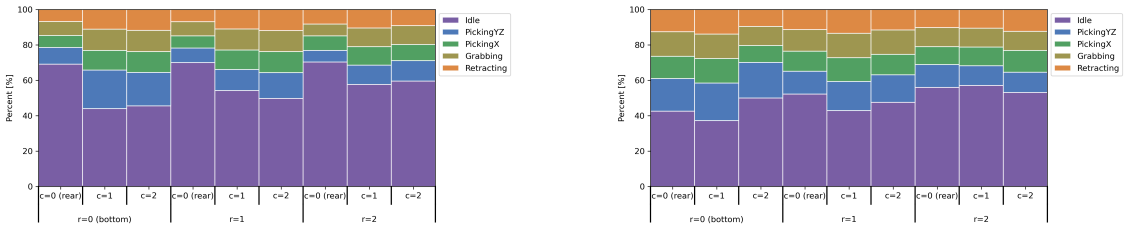
Difference in mean, 95% confidence level

| | FPE | | FPT | |
|---|---|---|---|---|
| C/R/C*R | p-value | Decision | p-value | Decision |
| 1/1/1 | 0.00 | Reject the null | 0.00 | Reject the null |
| 1/3/3 | 0.02 | Reject the null | 0.01 | Reject the null |
| 2/3/6 | 0.11 | Accept the null | 0.00 | Reject the null |
| 3/3/9 | 0.63 | Accept the null | 0.00 | Reject the null |

Table 3: Results of the t-test comparing the FPE and FPT means when using FCFS with ESS versus Goal Programming for 35 segments with length 3.5 m. Four configurations are used in this experiment with all rows partitioned to have the same number of fruits.

maximum workload, so the lack of reward for a higher FPE had a larger effect on them. The scheduler likely prioritized increasing FPT over FPE, since it would be rewarded, leading to the current results. The configurations with six and nine arms, on the other hand, were not at their maximum workload, which resulted in both a high FPT and an FPE at or above the desired FPE value.

Using MILP for scheduling achieved mean FPT values that are closer to the theoretical maximum FPT for each arm configuration. This theoretical maximum is calculated using Equation 37 with a minimum handling time equal to 2.55 s/fruit. For 1/1/1 this theoretical




(a) FCFS (b) Goal Programming

Figure 21: Plots comparing the percent time each arm spent idle versus in one of the four harvesting steps for a vacuum-type gripper. A single segment with 72 fruits was harvested in simulation using both FCFS and Goal Programming to compare the effects of the scheduling method on workload balancing. The harvester was set to have configuration 3/3/9 with rows partitioned to have the same number of fruits.

maximum is equal to 0.392 fruits/s and for 3/3/9 it is 3.529 fruits/s. When using FCFS, a single arm only reached 68% of the theoretical maximum FPT and nine arms 44%. MILP, on the other hand, reached 86% for one arm and 57% for nine arms. We had hypothesized that MILP could also obtain an increase in FPT more closely resembling a linear increase proportional to the number of arms, compared to FCFS. When we calculated the speedup from one to nine arms, Goal Programming had a slightly higher speedup value at 5.94, while FCFS achieved a speedup of 5.86. The values were more similar than expected, so we believe further experiments are important to determine if there is an intrinsic limit to the possible speedup due to some property of the system, or if changing limits and bounds for Goal Programming would lead to higher speedup values.

We hypothesized that MILP led higher mean FPT because it could better balance the workload between columns. By plotting the percent of time each arm in a 3/3/9 configuration spent idle versus harvesting in a single segment, we showed that the MILP does lead to better workload balancing between columns. Fig. 21 showed that when using FCFS, the back-most arms did less work than the two arms in front of them. This was true of all rows. When using MILP, there percent of time harvesting was similar between columns in the same rows.

Mean running time for each algorithm, in seconds

| Algorithm | C/R/C*R | | | |
|---|---|---|---|---|
| | 1/1/1 | 1/3/3 | 2/3/6 | 3/3/9 |
| FCFS | 0.00 +/- 0.00 | 0.00 +/- 0.00 | 0.01 +/- 0.00 | 0.02 +/- 0.00 |
| GP | 235.95 +/- 272.82 | 32.28 +/- 14.19 | 472.10 +/- 237.11 | 335.59 +/- 291.86 |

Table 4: Table showing the running time when using First Come First Served versus Goal Programming for the four harvester configurations. Running time is in seconds.

Furthermore, Fig. 21b showed that each row of arms spent a higher total percent of time harvesting instead of idle when MILP was used, compared to Fig.21a.

When comparing running times for the algorithms, FCFS with ESS produced results much faster than MILP, as shown in Table 4. Although it ran multiple loops to find the best speed and schedule combination, the mean running time for all configurations was 0.020 s or less. In contrast, Goal Programming took, for 1/1/1, 2/3/6, and 3/3/9, over 230 s per schedule, with the slowest mean running time of 472 s by 2/3/6. Configuration 1/3/3 was a little faster at 32.3 s. Decreasing the time limit in Gurobi too much would likely degrade the results. Instead, some potential ways to decrease the solving time include making the segments shorter, reducing the number of fruits needed to be scheduled, or reducing either the 95% FPE threshold or the maximum theoretical FPT threshold determined by Equation 37. Determining good bounds for the decision variables also has a large effect, not only on the results, but also on the running time.

# 8 Extending the harvest to whole orchard rows using Sliding Planning Windows

We want to extend our results form solving single segments to solving whole orchard rows. To do so, will test if the Sliding Planning Window model, described in Section 4.6, will result in a series of schedules that result in a maximized total Orchard Row-FPT (OR-FPT) and at least 95% Orchard Row-FPE (OR-FPE). These OR values will be calculated based on the total number of fruits, total distance, and total running time of the orchard row sections, unlike the Sliding Window-FPE (SW-FPE) and FPT (SW-FPT) values which are calculated over their respective Sliding Window. It was hypothesized that the shorter the travel length with respect to the Sliding Window length, the better the results. We believed that the faster rate of scheduling and increased overlapping lengths would allow the harvester to better optimize the row partitions, harvester speed, and schedule. Each Sliding Window will be scheduled using FCFS with ESS because results could be calculated in real time.

## 8.1 Sliding Planning Window for fruit harvesting

In this experiment we are testing if the Sliding Window method will result in at least 95% of fruits harvested while maximizing FPT. We will examine how changing the length of the travel length with respect to the harvester length affects the global Orchard Row results. We hypothesize that the smaller the travel length, the better the results.

Tests are performed on 32 digitized orchard row sections and four harvester configurations, using three travel lengths: the same size as the workspace length ($D = 1 * d_w$), 1/2 of the workspace length ($D = 1/2 * d_w$), and 1/3 of the workspace length ($D = 1/3 * d_w$). The harvester Sliding Window length is sized according to the settings in Section 5.1 for the workspace length and a 0.5 m long horizon. As an example, the harvester configuration 3/3/9 has a Sliding Window size of 3.8 m; its first travel length is equal to 3.3 m with a 0.5 m

| | C/R/C*R | | | |
|---|---|---|---|---|
| $D$ | 1/1/1 | 1/3/3 | 2/3/6 | 3/3/9 |
| $1 * d_w$ (m) | 1.0 | 1.0 | 2.15 | 3.30 |
| $1/2 * d_w$ (m) | 0.50 | 0.50 | 1.08 | 1.65 |
| $1/3 * d_w$ (m) | 0.33 | 0.33 | 0.72 | 1.10 |

Table 5: Sliding window lengths in meters for every harvester configuration.

overlap with the next Sliding Window due to the horizon. This is followed by the second and third travel lengths of 1.65 m with 2.15 m overlap and 1.1 m travel length with 2.7 m of overlap. The three Sliding Window lengths for every configuration are shown in Table 5.

The digitized orchard row sections all have different lengths between 9.33 to 17.50 m. Because of this, each run has a different number of Sliding Windows even if the travel lengths are the same. For all tests, rows are partitioned to have equal number of fruits and solved with FCFS with ESS. The possible harvester speeds range between 1 to 80 cm/s with increments of 1 cm/s.

We will examine how the travel length affects the final OR-FPE and OR-FPT, as well as the mean Sliding Window-FPT (SW-FPT), harvester speed, and the number of fruits. The OR-FPE and OR-FPT will be used to determine which of the three travel lengths is able to obtain the highest combined total FPT and FPE. Success will require that the harvester achieves OR-FPE of 95% or greater. The Sliding Window values, on the other hand, will be used to assess how the harvester performs as it travels along the orchard row.

We use ANOVA to test if there is a difference between mean OR-FPE and mean OR-FPT for the three travel lengths. Assumptions were tested using the Shapiro test of normality assumption and Bartlett test of homogeneity of variance. Neither assumption was generally met for any harvester configuration; though, for Normality, the Central Limit Theorem applies because the number of observations was greater than 15 for all samples. To work

Welch's ANOVA for OR-FPE, 95% CI

| $C/R/C*R$ | F | p-value | Decision |
|---|---|---|---|
| 1/1/1 | 0.115 | 0.891 | Accept the null |
| 1/3/3 | 0.015 | 0.985 | Accept the null |
| 2/3/6 | 0.214 | 0.808 | Accept the null |
| 3/3/9 | 0.581 | 0.560 | Accept the null |

Table 6: Results from using Welch's ANOVA to determine if at least one the Orchard Row-FPE means differed between the three travel lengths for four different harvester configurations. Each travel length was tested on 32 orchard row sections.

Welch's ANOVA for OR-FPT, 95% CI

| $C/R/C*R$ | F | p-value | Decision |
|---|---|---|---|
| 1/1/1 | 8.285 | 0.001 | Reject the null |
| 1/3/3 | 199.906 | 0.000 | Reject the null |
| 2/3/6 | 136.200 | 0.000 | Reject the null |
| 3/3/9 | 96.075 | 0.000 | Reject the null |

Table 7: Results from using Welch's ANOVA to determine if at least one of the Orchard Row-FPT means differed between the three travel lengths for four different harvester configurations. Each travel length was tested on 32 orchard row sections.

with the lack of homogeneity of variance, we use Welch's ANOVA and the Games-Howell Test for multiple comparisons to determine if there are differences between the means. All testes were performed with a 95% Confidence Level.

## 8.2 Experiment results and discussion

This experiment tested how the travel length affected results when using the Sliding Planning Window method. Three travel lengths were tested, the first was the same length as the harvester's workspace, the other two were shorter at one-half and one-third the workspace length. Welch's ANOVA tests were performed to determine if the were statistically significant differences between OR-FPE means and OR-FPT means for the three travel lengths for each harvester configuration. Results are shown in Tables 6 and 7.

The ANOVA results indicated that there was a statistically significant difference only between OR-FPT means and not OR-FPE means. The Games-Howell post-hoc test for multiple

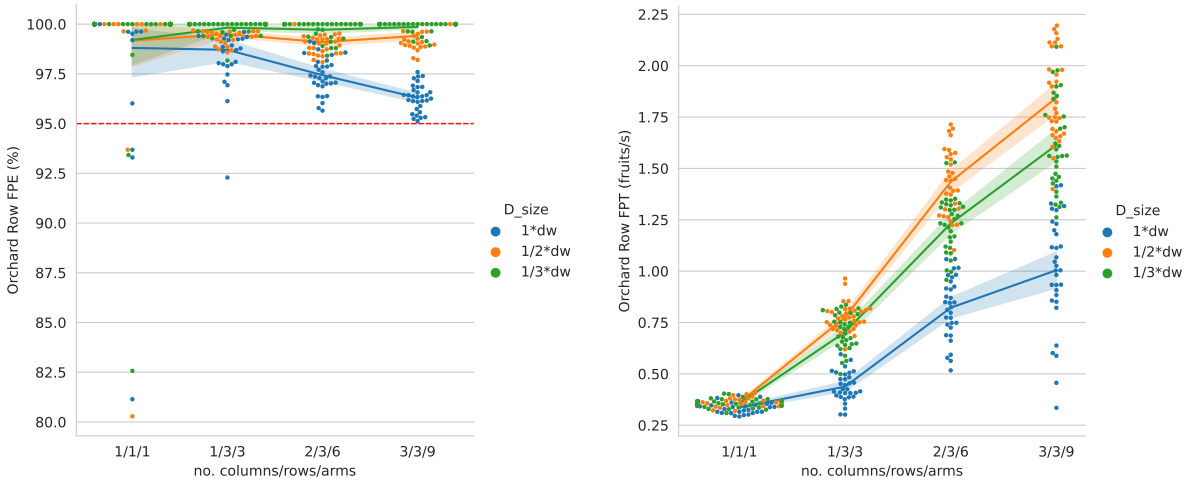
Games-Howell post-hoc test, 95% CI

| $A$ and $B$ | $1 * d_w$ and $1/2 * d_w$ | | | $1 * d_w$ and $1/3 * d_w$ | | | $1/2 * d_w$ and $1/3 * d_w$ | | |
|---|---|---|---|---|---|---|---|---|---|
| $C/R/C * R$ | T | p-value | Decision | T | p-value | Decision | T | p-value | Decision |
| 1/1/1 | -4.015 | 0.001 | Reject the null | -3.104 | 0.008 | Reject the null | 0.965 | 0.601 | Accept the null |
| 1/3/3 | -19.448 | 0.000 | Reject the null | -13.275 | 0.000 | Reject the null | 3.655 | 0.002 | Reject the null |
| 2/3/6 | -16.116 | 0.000 | Reject the null | -11.529 | 0.000 | Reject the null | 5.528 | 0.000 | Reject the null |
| 3/3/9 | -13.821 | 0.000 | Reject the null | -9.835 | 0.000 | Reject the null | 4.236 | 0.000 | Reject the null |

Table 8: Results from Games-Howell post-hoc test to determine exactly which OR-FPT group means are different.

comparisons was used with a 95% Confidence Level to test which OR-FPT means differed for each configuration. It found that the mean value of OR-FPT was significantly different between all groups and all configurations except between $1/2 * d_w$ and $1/3 * d_w$ for harvester configuration 1/1/1 (see Table 7).

We determined the direction of difference between means by plotting the mean and spread of the two Orchard Row variables over all four harvester configurations. Looking at Figure 22a, we saw that all travel lengths achieved the minimum 95% mean OR-FPE. It was surprising that the mean OR-FPE for $1 * d_w$ dropped as the number of arms increased, though it was a slight drop. $1/3 * d_w$ performed the best for OR-FPE, achieving close to 100% for all configurations. Figure 22a shows that the mean OR-FPT increased as the number of arms




(a) Orchard Row FPE

(b) Orchard Row FPT

Figure 22: Orchard Row FPE and FPT plots comparing the mean and spread obtained when using three different travel lengths, $D = 1 * d_w$, $D = 1/2 * d_w$, and $D = 1/3 * d_w$.

increased. Surprisingly, the best mean OR-FPT was achieved by the travel length $1/2 * d_w$ and not $1/3 * d_w$. It seems that decreasing the travel length below $1 * d_w$ improves both mean OR-FPE and mean OR-FPT up until a certain point. At some unknown shorter length, the mean OR-FPT begins to drop. This is a similar response in OR-FPT to that of FPT when using too high a harvester speed for single segments (see results for fixed speed $V = 100$ cm/s in Section 5.5.2), though the effect on OR-FPE is the opposite. When using a single fixed speed that it too high for single segments, the mean FPE drops to values way below the minimum threshold. Lastly, $1 * d_w$ produced the lowest OR-FPT.

To understand what produced these Orchard Row-FPE and FPT values, we plotted the mean Sliding Window-FPE, FPT, and harvester speed against the number of fruits in the Sliding Windows. From the scatter plots in Fig. 23, we saw that the shorter travel lengths habitually harvested less than 95% of the fruits in individual Sliding Windows, though with the re-scheduling this translated in OR-FPE values above 95%. Interestingly enough, $1 * d_w$ generally had higher individual SW-FPE values close to or above 95%, but it still had the lowest OR-FPE. For individual Sliding Windows, the SW-FPE does not have to reach the FPE threshold for the OR-FPE to reach the OR-FPE threshold as long as the travel length is smaller than the length of the workspace. The smaller the Sliding Window compared to the workspace, the lower the SW-FPE can be. $1/2 * d_w$ and $1/3 * d_w$ both resulted in higher individual SW-FPT values, with very similar scatter patterns in size and shape. The OR-FPT seemed to be related to the mean of the SW-FPT with respect to the number of fruits in the Sliding Window. However, $1/3 * d_w$ resulted in more total Sliding Windows, and they were more likely to have fewer fruits in them because there was so much overlap. This likely limited how high the OR-FPT could get. Lastly, the harvester speed scatter plot in Fig. 23 revealed that, between $1/2 * d_w$ and $1/3 * d_w$, $1/2 * d_w$ could choose higher harvester speeds for the same number of fruits. The same pattern emerged for the two larger travel lengths, $1 * d_w$ could choose higher speeds than $1/2 * d_w$.

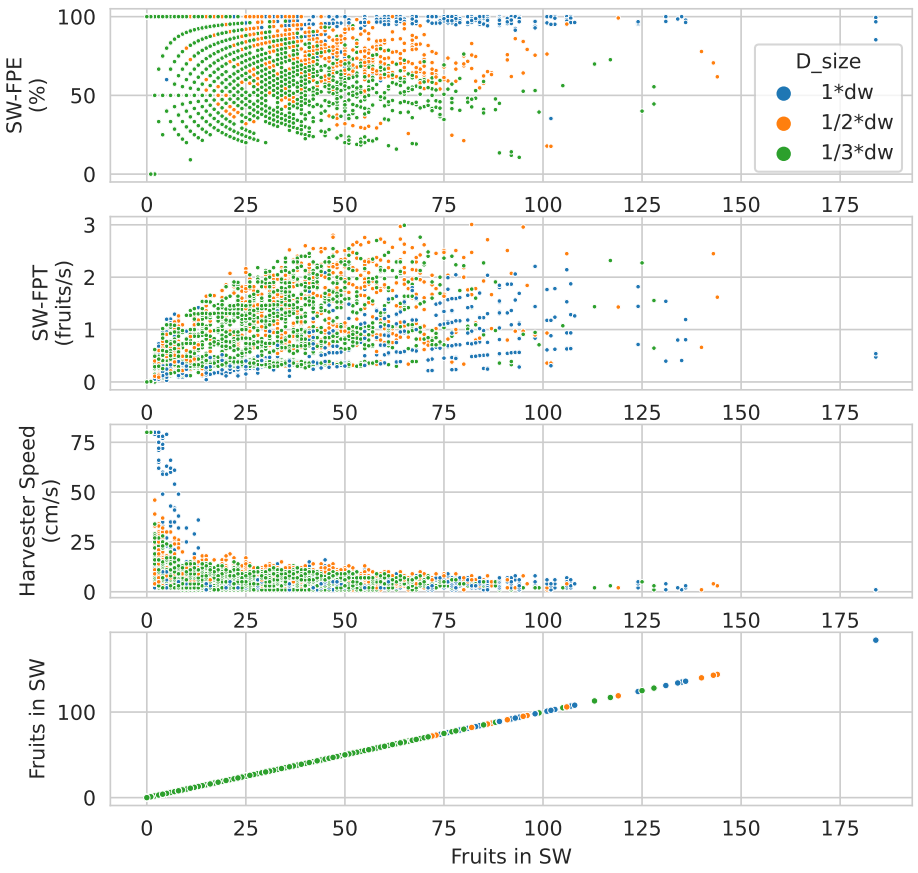
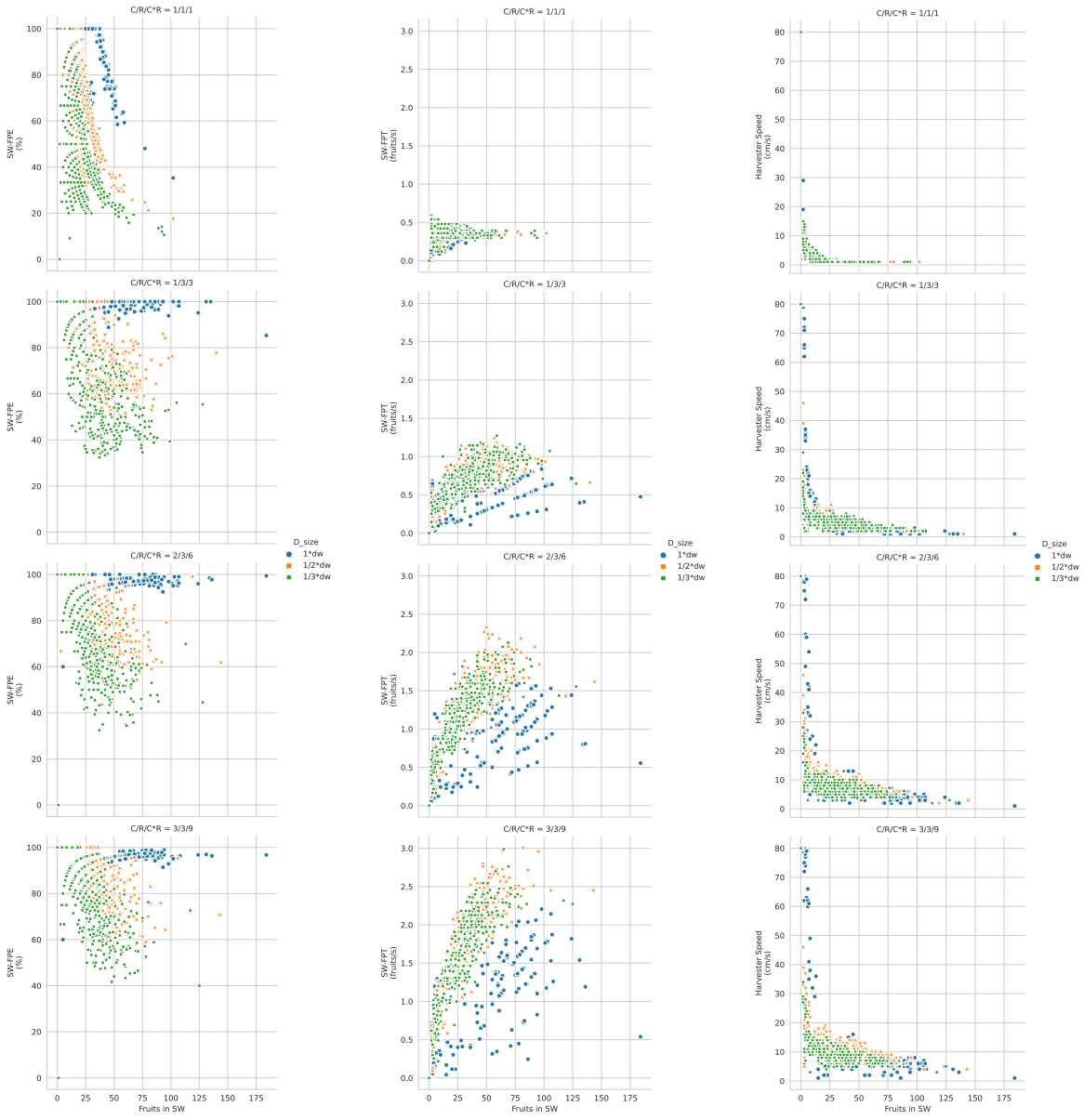



Figure 23: Scatter plots showing the relationships between the Sliding Window-FPE, FPT, harvester speed, and the number of fruits and the number of fruits in each Sliding Window. Points colored to denote the travel length that produced the point. Results obtained scheduling 32, 10 m orchard row sections using the Sliding Planning Window method.

The harvester configuration has a large effect on the results. The scatter plots were separated by harvester configuration in Fig. 24. These scatter plots show that 1/1/1 was limited to the lowest speeds, resulting in all SW-FPT values remaining below 0.75 fruits/s. This harvester configuration also had the widest SW-FPE spread for all travel lengths. As the number of arms increased, the SW-FPE spread dropped but the spread for SW-FPT and the harvester speed all increased. More arms allowed the harvester to choose higher speeds while maintaining the 95% OR-FPE, which translated into higher possible SW-FPT and OR-FPT values.




(a) Sliding Window FPE (b) Sliding Window FPT (c) Sliding Window V

Figure 24: Scatter plot showing the relationship between the Sliding Window-FPE, FPT, and the harvester speed with the number of fruits in each Sliding Window for each harvester configuration. Points colored to denote the travel length that produced the point. Results obtained scheduling 32, 10 m orchard row sections using the Sliding Planning Window method.

# 9 Conclusion

## 9.1 Row partitioning and speed selection

The first set of experiments presented simulation studies that compared how partitioning columns into rows and speed selection affected the FPT and FPE when harvesting single segments. FCFS was used to schedule the arm-to-fruit assignments. If the best speed was required, FCFS was run with ESS to find the speed that maximized FPT, provided that 95% fruits were harvested. Experiments were performed on nine harvester configurations spanning from one arm in a single column to nine arms in three columns and three rows. Digitized fruit locations for v-trellised apples were used in the simulations.

In the first experiment, two row partitioning strategies were compared, dividing the columns, so rows either had equal heights or an equal number of fruits. The best speed was found for each segment. Both partitioning methods achieved the minimum 95% FPE, however, partitioning by fruit allowed the harvester to move at faster speeds leading to higher FPT. As the number of arms increased, the difference in FPT between the partitioning methods increased. With that said, for the same number of arms, results were better when there were more arms set into more rows rather than more columns. This was because there was a maximum horizontal distance of 1 m compared to a maximum vertical distance of 3.5 m. The more arms in a column, the less they would have to move up and down to harvest all fruits while the horizontal travel stayed the same.

Results also showed that FPT followed a logarithmic curve with respect to the number of fruits in a segment. Few fruits limited the speed such that FPT remained low. As the number of fruits increased, FPT increased, though the rate of increase became slower. Speed, on the other hand, followed a curve related to $\ln(N)/N$ with respect to the number of fruits. Few fruits in a segment allow the system to choose faster speeds. As the number of fruits in a

segment increased, a slower speed would be chosen to reach the required 95% FPE. This shows that the number of arms has a large effect on the chosen speed and resulting FPT. More arms allow for faster speeds and, thus, faster FPT.

In the second experiment, we compared scheduling 3.5 m segments with six fixed speed versus choosing the best speed for each segment for both partitioning methods. Low fixed speeds resulted in high FPE but low FPT, while fast fixed speeds had the opposite effect. Choosing the best speed resulted in the best combined FPE and FPT. It led to high FPT values even while keeping FPE at or above 95%, unlike the fixed speeds which traded one for the other. This indicates that speed selection is very important when maximizing FPE and FPT at the same time. FCFS with ESS could be made faster by switching ESS to binomial search, speeding up results. Although FCFS's speed is not a bottleneck, per se, Binomial Search could be useful if the number of speeds is increased by adding fractional values, such as having the step between speeds be 0.25 cm/s. However, more informed algorithms would likley lead to even better combined FPT and FPE. For example, the use of MILP would provide near-optimal schedule and speed combinations, though its generally slow computation times need to be assessed.

## 9.2 Comparing FCFS and Goal Programming

The second set of experiments presented a comparison between using FCFS with ESS and a dual-objective MILP formulation based on Goal Programming; both algorithms were used to find schedules that maximize FPT while matching or exceeding a minimum FPE threshold. Schedule and speed combinations were computed in simulation for 35, 3.5 m segments of digitized orchard rows. Speed selection was limited to a range of 1 to 100 fruits/s. FCFS was limited to discrete speed values while the MILP formulation could choose from a continuous range.

Using Goal Programming resulted in higher mean FPT for all four harvester configurations tested in this experiment. The configurations with one and three arms had FPE lower than the threshold by around 2.5%. We hypothesize that the configurations with fewer arms were too close to their maximum workload to obtain the minimum FPE with the current MILP FPE upper bound settings.

Goal Programming was expected to result in an almost linear increase in FPT proportional to the increase in arms. Although it did result in better mean FPT for all configurations, MILP did not result in a much higher speedup than FCFS as the number of arms increased. Goal Programming resulted in a speedup of 5.94, while FCFS resulted in a speedup of 5.86 from one to nine arms. More research is necessary to determine if there is a limit to the possible speedup which is inherent to the problem, or if changes to the MILP formulation could improve this value. We suspect that MILP achieves higher mean FPT values compared to FCFS because it might be better at balancing the workload between columns, allowing it to choose faster speeds. However, only one example of this is presented, so it cannot be said with certainty.

Lastly, Goal Programming performs better than FCFS with ESS for all harvester configurations, but it is very expensive computationally and time-wise. FCFS with ESS took at most 0.020 s to compute, while the fastest MILP solved took 32.4 s. More realistically, however, Goal Programming took over 230 s to solve for a schedule. A smaller time limit on the solver would speed up the computation but would result in worse schedules and speeds. Instead, work should go into determining if changes to the bounds in the formulation, segment length, or other strategies can be used to decrease solving time. FCFS, on the other hand, should be used if real time scheduling is a requirement, rather than near-optimal results.

## 9.3 Sliding Planning Window

The third set of experiments presented a scheduling algorithm-agnostic method to solve whole orchard rows semi-dynamically, while still maximizing the orchard row's final FPT and harvesting at least 95% of fruits. Scheduling and speed selection were computed for Sliding Window lengths made up of the harvester's workspace length plus a 0.5 m horizon. However, the schedule and harvester speed were only acted upon over a set travel distance before a new schedule and harvester speed combination were produced. The Sliding Planning Window method was tested with three different travel lengths to see if there was any difference, and if there was, which of the three travel lengths worked the best. Schedules and speed combinations were computed for 32 digitized orchard rows of varying lengths using FCFS with ESS. Speed selection was limited to the range 1 to 80 cm/s.

All three travel lengths achieved the minimum mean OR-FPE. The best mean OR-FPT was produced by the middle-valued travel length which was equal to half the harvester's workspace length. The worst outcome was produced by the longest travel length which was equal to the workspace length, such that there was no overlap between Planning Windows. There does seem to be a "best" travel length, since the shortest travel length produced a worse OR-FPT than the middle-length one; however, finding this best length is left as an exercise for future research.

By plotting the individual SW-FPE, SW-FPT, harvester speed, and number of fruits, it was determined that longer travel lengths allow the harvester to choose higher speeds. Furthermore, individual SW-FPE values can be a lot lower than the FPE threshold and the OR-FPE will still match or exceed the threshold. The smaller the travel length compared to the harvester workspace, the lower the SW-FPE can be. Dividing the SW-FPE, SW-FPT, and harvester speed scatter plots by the harvester configuration showed that all configurations followed these general trends. However, this also allowed us to show that the number of

arms has a large effect on the results. Like for the single segments, as the number of arms increase, the harvester can choose higher speeds and still obtain the same or better SW-FPE values. Because of this, more arms can obtain higher SW-FPT values which translate to higher OR-FPT values.

This method is simple to employ, but it allows the harvester to react more dynamically to incoming fruit distributions and to changes in the orchard. A hypothesized benefit of this method is that if there are changes during the harvest of a Sliding Window, e.g. apples move due to a branch moving, they will be taken into account within the next schedule as long as the change is within the new Sliding Window. In fact, this method does not require fruit tracking between Sliding Windows because each one is solved independently. This means that the computer vision system requires less resources and there are fewer potential points of failure.

A challenge to this method is that it requires constant rescheduling. FCFS with ESS works well with it because it can be calculated in real time. However, more time intensive scheduling algorithms might have trouble keeping up. If done thoughtfully, it might be possible to use the time between Planning Windows, while the harvester harvests apples, to compute schedules. This comes with some challenges, such as needing to create the next fruit map ahead of time, reducing the system's ability to react to uncertainty in fruit locations or changes between Planning Windows. More research is needed to evaluate this method.

# 10 Future Research

There are still several limitations in this research that will be explored in future research:

1. Testing should be performed on higher fruit densities to evaluate any changes, if any, to the results. Because the apples were digitized after a first harvest pass had already

been performed, the densities were lower than might be expected. We hypothesize that with higher densities, the harvester will choose lower harvester speeds to continue harvesting 95% of fruits. The effects on FPT are less clear since the increase in fruits harvested might offset the lower speeds.

2. Future work should incorporate uncertainty, such as the chance that identified apples might be false positives or that apples may have moved between Planning Windows. Although research such as Li et al., 2023 incorporates some uncertainty, it does so by allowing arms to attempt the harvest the same fruit three times before removing the apple from consideration. For the continually moving harvester, this may not be appropriate because a Planning Window is treated as a static problem over the Travel Length. If an arm has to make multiple attempts, the system might not be able to follow through on the rest of the existing schedule. Instead, shorter Travel Distances between Planning Windows might allow the system to replan fast enough that the harvest of the problematic fruits could be re-attempted in each Planning Window.

3. It will be essential to add checks to avoid collisions due to changes in arm and row limit placement between Planning Windows. Current simulations do not take into account how the arm's starting position and the changes in row limits between Planning Windows might cause conflicts as the arms move to harvest fruits within their new rows. A safety measure, such as giving the arms time to move to their new positions, would help improve the safety of the system.

4. All experiments were performed in simulation. Future work needs to be implemented on a physical prototype to obtain more realistic results.

# References


2023 technology research review. (2022). *Washington Tree Fruit Research Commission*.

*Adapting to Farm Worker Scarcity Survey 2019* (tech. rep.). (2019). California Farm Bureau Federation. https://www.cfbf.com/wp-content/uploads/2019/06/LaborScarcity.pdf

*Agricultural Labor Availability Survey Results* (tech. rep.). (2017). California Farm Bureau Federation. http://www.cfbf.us/wp-content/uploads/2017/10/CFBF-Ag-Labor-Availability-Report-2017.pdf

Arikapudi, R. (2019). *Model-based estimation of fruit harvesting performance of arrays of telescopic robotic arms* [Doctoral dissertation, University of California, Davis].

Arikapudi, R., & Vougioukas, S. G. (2023). Robotic tree-fruit harvesting with arrays of cartesian arms: A study of fruit pick cycle times. *Computers and Electronics in Agriculture*, *211*, 108023.

Bac, C. W., van Henten, E. J., Hemming, J., & Edan, Y. (2014). Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *Journal of Field Robotics*, *31*(6), 888–911. https://doi.org/https://doi.org/10.1002/rob.21525

Barnett, J. (2018). *Prismatic axis, differential-drive robotic kiwifruit harvester for reduced cycle time* [Doctoral dissertation, The University of Waikato]. https://hdl.handle.net/10289/12444

Barnett, J., Duke, M., Au, C. K., & Lim, S. H. (2020). Work distribution of multiple Cartesian robot arms for kiwifruit harvesting. *Computers and Electronics in Agriculture*, *169*, 105202. https://doi.org/10.1016/j.compag.2019.105202

Besset, P., & Béarée, R. (2017). FIR filter-based online jerk-constrained trajectory generation. *Control Engineering Practice*, *66*, 169–180. https://doi.org/10.1016/j.conengprac.2017.06.015

Blanco, O. (2016, September). The worker shortage facing America's farmers. Retrieved August 26, 2020, from https://money.cnn.com/2016/09/29/news/economy/american-farm-workers/index.html

Bozma, H. I., & Kalalıoğlu, M. (2012). Multirobot coordination in pick-and-place tasks on a moving conveyor. *Robotics and Computer-Integrated Manufacturing*, *28*(4), 530–538.

Calvin, L., Martin, P., & Simnitt, S. (2022). Adjusting to higher labor costs in selected us fresh fruit and vegetable industries.

Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M., & Tardós, J. D. (2021). Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, *37*(6), 1874–1890.

Caramia, M., & Dell'Olmo, P. (2020). Multi-objective optimization. In *Multi-objective management in freight logistics: Increasing capacity, service level, sustainability, and safety with optimization algorithms* (pp. 21–51). Springer International Publishing. https://doi.org/10.1007/978-3-030-50812-8_2

Choudhury, S., Gupta, J. K., Kochenderfer, M. J., Sadigh, D., & Bohg, J. (2022). Dynamic multi-robot task allocation under uncertainty and temporal constraints. *Autonomous Robots*, *46*(1), 231–247.

Daoud, S., Chehade, H., Yalaoui, F., & Amodeo, L. (2014). Efficient metaheuristics for pick and place robotic systems optimization. *Journal of Intelligent Manufacturing*, *25*(1), 27–41.

Deb, K., Sindhya, K., & Hakanen, J. (2016). Multi-objective optimization. In *Decision sciences* (pp. 161–200). CRC Press.

Edan, Y., Engel, B., & Miles, G. E. (1993). Intelligent control system simulation of an agricultural robot. *Journal of intelligent and Robotic Systems*, *8*(2), 267–284.

Edan, Y., & Miles, G. E. (1993). Design of an agricultural robot for harvesting melons. *Transactions of the ASAE*, *36*(2), 593–603.

*Farm Labor (May 2020)* (tech. rep.). (2020, May). USDA, National Agricultural Statistics Service.

Fomin, F. V., & Lingas, A. (2002). Approximation algorithms for time-dependent orienteering. *Information Processing Letters*, *83*(2), 57–62.

Garey, M. R., & Johnson, D. S. (1978). "strong"np-completeness results: Motivation, examples, and implications. *Journal of the ACM (JACM)*, *25*(3), 499–508.

Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G., & Vathis, N. (2014). Efficient heuristics for the time dependent team orienteering problem with time windows. *Applied Algorithms: First International Conference, ICAA 2014, Kolkata, India, January 13-15, 2014. Proceedings 1*, 152–163.

Gerkey, B. P., & Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International journal of robotics research*, *23*(9), 939–954.

Gunantara, N. (2018). A review of multi-objective optimization: Methods and its applications. *Cogent Engineering*, *5*(1), 1502242.

Harrell, R. (1987). Economic analysis of robotic citrus harvesting in florida. *Transactions of the ASAE*, *30*(2), 298–0304.

Haspel, T. (2017). Perspective — Illegal immigrants help fuel U.S. farms. Does affordable produce depend on them? *Washington Post*. Retrieved June 22, 2020, from https://www.washingtonpost.com/lifestyle/food/in-an-immigration-crackdown-who-will-pick-our-produce/2017/03/17/cc1c6df4-0a5d-11e7-93dc-00f9bdd74ed1_story.html

Huang, Y., Chiba, R., Arai, T., Ueyama, T., & Ota, J. (2012). Part dispatching rule-based multi-robot coordination in pick-and-place task. *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 1887–1892.

Huang, Y., Chiba, R., Arai, T., Ueyama, T., & Ota, J. (2015). Robust multi-robot coordination in pick-and-place tasks based on part-dispatching rules. *Robotics and Autonomous Systems*, *64*, 70–83.

Humbert, G., Pham, M., Brun, X., Guillemot, M., & Noterman, D. (2015). Comparative analysis of pick & place strategies for a multi-robot application. *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, 1–8. https://doi.org/10.1109/ETFA.2015.7301450

Humbert, G., Pham, M.-T., Brun, X., Guillemot, M., & Noterman, D. (2015). Comparative analysis of pick & place strategies for a multi-robot application. *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, 1–8.

Jocher, G., Chaurasia, A., & Qiu, J. (2023, January). *Ultralytics YOLO* (Version 8.0.0). https://github.com/ultralytics/ultralytics

Jones, D., & Tamiz, M. (2016). A review of goal programming. *Multiple criteria decision analysis: State of the art surveys*, 903–926.

Koostra, G., Wang, X., Blok, P. M., Hemming, J., & van Henten, E. (2021). Selective harvesting robotics: Current research, trends, and future directions. *Current Robotics Reports*, *2*(1), 95–104. https://doi.org/https://doi.org/10.1007/s43154-020-00034-1

Li, T., Xie, F., Zhao, Z., Zhao, H., Guo, X., & Feng, Q. (2023). A multi-arm robot system for efficient apple harvesting: Perception, task plan and control. *Computers and Electronics in Agriculture*, *211*, 107979.

Mann, M. P., Zion, B., Shmulevich, I., Rubinstein, D., & Linker, R. (2016). Combinatorial Optimization and Performance Analysis of a Multi-arm Cartesian Robotic Fruit Harvester—Extensions of Graph Coloring. *J Intell Robot Syst*, *82*(3-4), 399–411. https://doi.org/10.1007/s10846-015-0211-5

Maria, B., & Salassi, M. S. (2019). Trends in U.S. Farm Labor and H-2A Hired Labor: Policy and Related Issues. *MLR*, *34*(1). https://doi.org/10.21916/mlr.2013.39

Mattone, R., Adduci, L., & Wolf, A. (1998). Online scheduling algorithms for improving performance of pick-and-place operations on a moving conveyor belt. *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, *3*, 2099–2105.

Ngatchou, P., Zarei, A., & El-Sharkawi, A. (2005). Pareto multi objective optimization. *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, 84–91. https://doi.org/10.1109/ISAP.2005.1599245

Recce, M., Taylor, J., Plebe, A., & Tropiano, G. (1996). Vision and neural control for an orange harvesting robot. *Proceedings of International Workshop on Neural Networks for Identification, Control, Robotics and Signal/Image Processing*, 467–475. https://doi.org/10.1109/NICRSP.1996.542791

Scarfe, A. J. (2012). *Development of an autonomous kiwifruit harvester: A thesis presented in partial fulfilment of the requirements for the degree of doctor of philosophy in industrial automation at massey university, manawatu, new zealand.* [Doctoral dissertation, Massey University].

Tang, Y., Chen, M., Wang, C., Luo, L., Li, J., Lian, G., & Zou, X. (2020). Recognition and localization methods for vision-based fruit picking robots: A review. *Frontiers in Plant Science*, *11*, 510. https://doi.org/10.3389/fpls.2020.00510

Tika, A., Gafur, N., Yfantis, V., & Bajcinca, N. (2020). Optimal scheduling and model predictive control for trajectory planning of cooperative robot manipulators. *IFAC-PapersOnLine*, *53*(2), 9080–9086.

USDA ERS - Farm Labor. (n.d.). Retrieved July 23, 2020, from https://www.ers.usda.gov/topics/farm-economy/farm-labor/#wages

Villacrés, J., Viscaino, M., Delpiano, J., Vougioukas, S., & Cheein, F. A. (2023). Apple orchard production estimation using deep learning strategies: A comparison of tracking-by-detection algorithms. *Computers and Electronics in Agriculture*, *204*, 107513.

Williams, H. A., Jones, M. H., Nejati, M., Seabright, M. J., Bell, J., Penhall, N. D., Barnett, J. J., Duke, M. D., Scarfe, A. J., Ahn, H. S., et al. (2019). Robotic kiwifruit harvesting using machine vision, convolutional neural networks, and robotic arms. *biosystems engineering*, *181*, 140–156.

Willmott, C. J. (1981). On the validation of models. *Physical geography*, *2*(2), 184–194.

Yang, S., Jia, B., Yu, T., & Yuan, J. (2022). Research on multiobjective optimization algorithm for cooperative harvesting trajectory optimization of an intelligent multiarm straw-rotting fungus harvesting robot. *Agriculture*, *12*(7), 986.

Zahniser, S., Taylor, J. E., Hertz, T., & Charlton, D. (2018, November). *Farm Labor Markets in the United States and Mexico Pose Challenges for U.S. Agriculture* (tech. rep. No. 201). USDA, Economic Research Service.

Zhang, Q., & Karkee, M. (2016). Fully Automated Tree Fruit Harvesting. *23*(6), 16–17.

Zhang, Q., & Pierce, F. J. (2016). *Agricultural automation: Fundamentals and practices*. CRC Press.

Zhang, Z., Heinemann, P. H., Liu, J., Baugher, T. A., & Schupp, J. R. (2016). The development of mechanical apple harvesting technology: A review. *Transactions of the ASABE*, *59*(5), 1165–1180.

Zion, B., Mann, M., Levin, D., Shilo, A., Rubinstein, D., & Shmulevich, I. (2014). Harvest-order planning for a multiarm robotic harvester. *Computers and Electronics in Agriculture*, *103*, 75–81. https://doi.org/10.1016/j.compag.2014.02.008