MOTION PLANNING OF A MOBILE CARTESIAN MANIPULATOR FOR OPTIMAL HARVESTING OF 2-D CROPS

M. P. Mann, D. Rubinstein, I. Shmulevich, R. Linker, B. Zion

ABSTRACT. *A 3DOF mobile Cartesian robotic harvester for two-dimensionally distributed crops such as melons is being developed. A two-step procedure to calculate the trajectory of its manipulator that will result in the maximum number of melons harvested is described in this article. The goal of the first step is to calculate the minimum-time trajectory required to traverse between any two melons while adhering to velocity, acceleration, location, and endpoint constraints. This step is accomplished in a hierarchal manner by solving several subproblems involving optimal control and nonconvex optimization, enabling optimal (maximum) melon harvesting to be formulated as an orienteering problem with time windows. In the second step, the orienteering problem is solved using the moving branch and prune method, based on dynamic programming. This enables suboptimal sequences of melons (out of all options) to be eliminated on the fly without the need to solve the entire problem at once. An example is shown to demonstrate the efficacy of the algorithm.*

Keywords. *Dynamic programming, Orienteering, Robotic harvesting melons, Time optimal control, Time windows.*

In many places around the world, the agriculture sector suffers from declining profitability and shortage of labor (Srinivasan, 2007). In Israel, for example, agricultural output decreased 50% and farmers' income decreased 84% between 1990 and 2004 (Shafir, 2006), while the tremendous shortage of available manpower during peak harvestings seasons resulted in a financial loss of over 150 million NIS (Friedman, 2010).

Applying harvesting automation technology such as robots has potential to significantly curb these trends (Pedersen et al., 2006, 2008). While many prototypes of harvesting robots have been developed for various crops (Belforte et al., 2006; Edan et al., 2009), many challenges inhibit them from being deployed commercially. These challenges include the unstructured environment of the field, the constantly changing field conditions, obstruction of fruit by leaves or other fruit, and the sensitivity of fruit handling (Edan et al., 2000; Kassler, 2001; Perry, 2009). Most of all, until recently the economic feasibility of inexpensive manual labor obviated the need for employing robotic harvesters. Optimizing the robots' performance such that they become cost-effective is therefore particularly imperative in order to advance robots toward deployment in agriculture. This article sets out to accomplish one aspect of that goal by increasing the ratio of fruit harvested by robotic harvesters.

Submitted for review in April 2013 as manuscript number IET 10225; approved for publication by the Information & Electrical Technologies Division of ASABE in December 2013.

The authors are **Moshe P. Mann,** Graduate Student, **Dror Rubinstein, ASABE Member,** Senior Researcher, **Itzhak Shmulevich, ASABE Fellow,** Professor, and **Rafi Linker,** Associate Professor, Department of Civil and Environmental Engineering, Technion, Haifa, Israel; **Boaz Zion,** Senior Researcher, Agricultural Research Organization, Bet Dagan, Israel. **Corresponding author:** Moshe P. Mann, Technion City, Haifa 32000, Israel; phone: 077-887-3506; e-mail: mpm@tx.technion.ac.il.

The study presented here is one part of an attempt to develop a robotic harvester for two-dimensional crops in general and melons in particular. This is because over one quarter of the total costs of melon production is spent on manual labor (Westberry, 2009). Our approach is to separate the sensing and melon identification operation from the harvesting operation so that the melons are mapped in local coordinates using a separate platform. The coordinates of the mapped targets are then available for the robot before it begins its harvesting operation. The goal of optimizing the automated harvesting of melons is to yield maximum harvest at minimum cost. This entails devising a motion planner for the robotic manipulator that ensures that the robot harvests as many melons as possible.

Motion planning of robotic manipulators in agriculture is complex and has not received a lot of attention. Minimum-time robot motion planning, though, is a long-standing problem in applied robotics. No general analytical solution exists for time-optimal motions of robots (Jazar, 2010), and the most that can be analytically achieved is a lower bound on the time required to move from point to point (Shin and McKay, 1986). Edan et al. (1990, 1991) calculated the minimum-time trajectory and picking sequence of a citrus-picking robot between two points by minimizing the geodesic distance between them. However, the algorithms used were heuristic and the workspace was divided into subspaces, and so did not achieve an exact optimal solution. In addition, the minimum-time sequence was heavily dependent on subspace size, presenting an uncomfortable trade-off between optimality and ease of computing. Edan and Miles (1993, 1994) examined how the performance of a melon harvesting robot is affected by the number of manipulators, actuator speeds, advancement speeds, and mode of operation. The results of the corresponding experiments were presented by Edan et al. (2003). While they demonstrate that the robot's parameters

  ISSN 2151-0032 DOI 10.13031/trans.56.10225

have a significant effect on its harvesting time and harvesting percentage, a strictly optimal methodology for maximum harvest or minimum time was not presented. A more detailed analysis of the optimum robot parameters and trajectory is called for in this article.

Planning the optimal sequence of melons to harvest that will maximize the yield is an example of the orienteering problem (OP), also known as the selective traveling salesman problem, an NP hard problem (i.e., requiring exponential time to solve) first introduced by Tsiligirides (1984) (see Vansteenwegen et al., 2011, for a literature survey of orienteering). The OP is a game in which a number of locations are given, each with a given score, and the player must visit as many as possible within a given time frame. The goal of the OP is to maximize the total collected score within the time frame. A subset of OP is orienteering with time windows (OPTW), in which each vertex can only be visited within its specific time window. Heuristic solutions have been proposed by Kantor and Rosenwein (1992) and Bansal et al. (2004), while approximation algorithms were presented by Chekuri and Korula (2007), and a dynamic programming-based solution was presented by Righini and Salani (2006, 2009).

As described in the following Model Description section, the robot design used in this simulation is comprised of a mobile Cartesian manipulator that traverses over a row of melons at constant speed, picks the melons, and places them on a moving conveyor. Maximum robotic melon harvesting using this model is thus particularly suited to formulation as an OPTW. This is because, on the one hand, the continuously advancing mobile platform means that each melon can only be picked within a given time frame, while on the other hand, the Cartesian configuration means that the minimum traversal time between any two melons is time-independent. The objective of this article is therefore to compute the trajectory of the robot's manipulator that will result in the maximum number of melons harvested. This objective consists of first computing the minimum time trajectory between any two melons, followed by the sequence of melons for the robot to pick that results in a maximum harvest.

In the first step, we calculate the minimum-time trajectory to traverse from one melon to another and the time window within which this can take place. However, because the harvested melon can be placed at variable locations on the conveyor, this is a mixed optimization-optimal control problem, as both the minimal-time trajectory from point to point and the placement location are to be optimized. This is accomplished in several stages: the former stages using classical time optimal control methods (Sethi and Thompson, 2006), and the latter stages using nonsmooth and nonconvex optimization methods (Makela and Neittaanmaki, 1992).

In the second step of the procedure, we describe how to solve the OPTW using the moving branch and prune method. This method is similar to that of Righini and Salani (2006), but we introduce tighter bounds to the problem and expanded pruning criteria so that the optimal harvesting sequence is obtained more rapidly without the need for state space relaxation employed by the latter. An example demonstrating the efficacy of the algorithm by comparing it to a heuristic, the nearest neighbor, is shown in the Results section.

Model Description

The robot is designed as a rectangular frame that travels along a two-dimensional field at a constant velocity (v_t). An *x-y-z* Cartesian manipulator consisting of three perpendicular linear actuators is mounted on the frame (fig. 1). When the manipulator reaches a melon, the gripper picks it off the ground, places it on one of the two longitudinal conveyors that move at velocity v_c relative to the tractor, and then aims to reach the next melon. The longitudinal conveyors carry the harvested melons to the back of the structure, where they are collected and packed. To prevent damage to the melons, they must be placed at zero velocity relative to the belt, so knowledge of v_c is necessary. The harvesting cycle consists of the reach stage and the place stage, i.e., reaching for the fruit and placing it on the conveyor belt, respectively, as shown in figure 2. The time required for the gripper to grasp the fruit and the time required to release it are assumed constant. The locations of all melons in global coordinates are assumed known beforehand.

Between any two melons, the manipulator begins at time t_S at the first melon, places it at some placement point on the conveyor belt at time t_D, and then reaches the second melon at time t_E, as shown in figure 2. At all times, the manipulator remains within the rectangular frame. In addition, the relative velocity and acceleration of the manipulator are

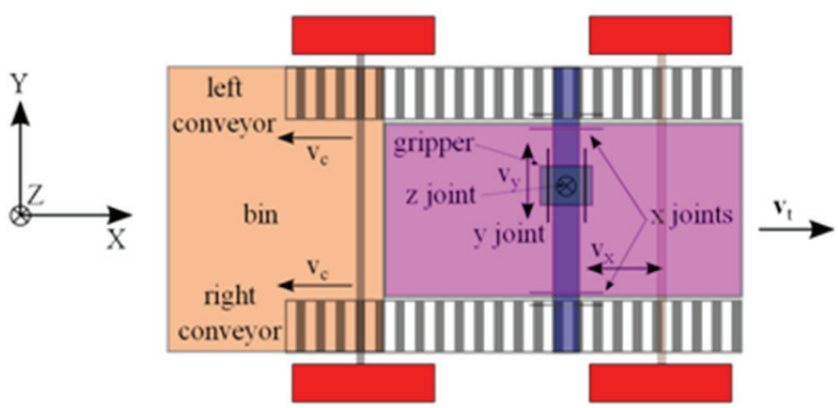


Figure 1. Schematic of robotic melon harvester.

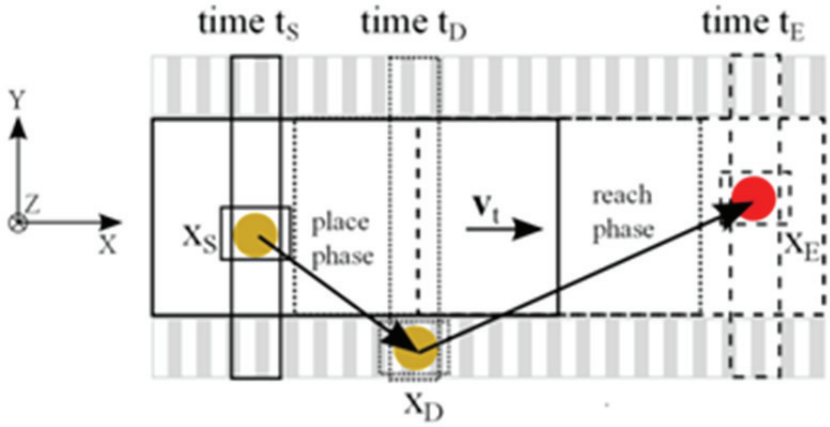


Figure 2. Time-lapse sequence of melon harvesting. Beginning at time t_S, the manipulator places a melon on the conveyor belt at time t_D, and then reaches the next melon at time t_E, repeating the cycle.

bounded by v_{max} and a_{max}, respectively. The time span between any two fruit thus consists of the place stage of the first fruit and the reach stage of the second fruit.

Step 1: Minimum-Time Trajectory between any Two Melons

Because the manipulator is Cartesian, the minimum possible time span of the place phase T^P and the reach phase T^R in each of the *x-y-z* directions can be treated separately. These time spans are denoted as (T_x^P, T_y^P) and (T_x^R, T_y^R), respectively. The total time to reach a given point is thus known as the $l\infty$ time (Korte and Vygen, 2008):

$$\begin{aligned} T^P &= max\left(T_x^P, T_y^P, T_z^P\right) \\ T^R &= max\left(T_x^R, T_y^R, T_z^R\right) \end{aligned} \tag{1}$$

The distribution of the melons is two-dimensional, so the time span of the *z*-actuator can be assumed constant for each harvesting cycle. The time span to reach a point in the *y-z* direction for both phases is thus concatenated into the following form:

$$\begin{aligned} T_{yz}^P &= max\left(T_y^P, T_z^P\right) \\ T_{yz}^R &= max\left(T_y^R, T_z^R\right) \end{aligned} \tag{2}$$

Only the *x-y* coordinates need be considered. Thus, denoting the *x-y* coordinates of the first and second melons of any two melons picked consecutively as $[x_S, y_S]$ and $[x_E, y_E]$, respectively, the placement point between them as $[x_D, y_D]$, and the displacement and velocity of the manipulator in global coordinates as $[x_1, y_1]$ and $[x_2, y_2]$, respectively, the optimization-optimal control problem can be stated as follows:

Find a piecewise continuous acceleration control input vector

$$u = \left[u_x \; u_y\right]^T : [t_S, t_E] \rightarrow \Omega \subseteq R^2$$

and dropoff location

$$x_D \in R, y_D \in \left\{-\frac{W}{2}, \frac{W}{2}\right\}$$

such that the time

$$max\left(\int_{t_S}^{t_D} dt_x^P, \int_{t_S}^{t_D} dt_{yz}^P\right) + max\left(\int_{t_D}^{t_E} dt_x^R, \int_{t_D}^{t_E} dt_{yz}^R\right)$$

traversed between any two points is minimized and that satisfies the differential equations:

$$\begin{gathered} \dot{x}_1(t) = x_2(t), \; \dot{x}_2 = u_x(t) \\ \text{(velocity and acceleration in } x\text{-direction)} \\ \dot{y}_1(t) = y_2(t), \; \dot{y}_2(t) = u_y(t) \\ \text{(velocity and acceleration in } y\text{-direction)} \end{gathered} \tag{3}$$

the following endpoint (first and second melons) and placement point conditions:

$$\begin{gathered} x_1(t_S) = x_S, \; y_1(t_S) = y_S, \\ x_1(t_E) = x_E, \; y_1(t_E) = y_E \\ \text{(endpoint positions)} \\ x_2(t_S) = 0, \; y_2(t_S) = 0, \\ x_2(t_E) = 0, \; y_2(t_E) = 0 \\ \text{(endpoint positions)} \\ x_1(t_D) = x_D, \; y_1(t_D) = y_D \\ \text{(placement position)} \\ x_2(t_D) = v_t + v_c, \; y_2(t_D) = 0 \\ \text{(placement velocity)} \end{gathered} \tag{4}$$

where $t_S < t_D < t_E$, and the following inequality constraints:

$$\begin{gathered} -a_{max} \le u_x(t) \le a_{max}, \; -a_{max} \le u_y(t) \le a_{max} \\ \text{(acceleration limits)} \\ v_t - v_{max} \le x_2(t) \le v_t + v_{max}, \; -v_{max} \le y_2(t) \le v_{max} \\ \text{(velocity limits)} \\ v_t t < x_1(t) < v_t t + L, \; -\frac{W}{2} \le y_1(t) \le \frac{W}{2} \\ \text{(position limits)} \end{gathered} \tag{5}$$

for all $t \in [t_S, t_E]$.

In other words, we want to determine both the control and the dropoff point that will minimize the time of traversal between any two melons. In addition, the traversal between any two melons must commence within a limited time window in order to be feasible. Therefore, it is also desired to find the range of start times $t_S \in [t_{Smin}, t_{Smax}]$ under which the problem has a solution, i.e., $t_E < \infty$.

The solution of the first step consists of four stages: (1) determination of the minimum-time trajectory for the manipulator, (2) determination of the optimal dropoff location's y-coordinate), (3) determination of the optimal dropoff location's x-coordinate, and (4) determination of the time window for traversal from melon to melon.

Stage 1: Minimum-Time Trajectory for Manipulator

The first step is to determine acceleration input u that will bring the position of the gripper in a single dimension (x or y) to any displacement for any arbitrary endpoint velocities for either the reach or place phase with no constraints on location. In practice, the endpoint velocities (the velocities while picking a fruit or when placing it on a conveyor), denoted by v_S and v_E in table 1, may be nonzero due to either the nonzero velocity of the conveyor belt or the impact velocity of the manipulator with the melon within tolerance limits, and so must be accounted for in an arbitrary case. This trajectory is given by the well-known time optimal control problem, solved using Pontryagin's minimum principle (PMP; Naidu, 2003). The solution is the classic trapezoidal velocity profile, with the acceleration a bang-off-bang input. The velocity profile is one of the four types given in table 1 depending on the displacement l, yielding the traversal time T as a function of the displacement, $T(l)$. These times are tabulated in table 1 for each type along with the profile's maximum/minimum velocity (v_m), switching times (t_1, t_2, t_m), acceleration profile (u), and range of traversal distances under which the profile is active (third and fourth columns). From equation 1, the total time for any phase as a function of the x-displacement (l_x) is then:

$$T(l_x) = max\left(T_x(l_x), T_{yz}\right) \tag{6}$$

with the times specific to the reach and place phase denoted as $T^R(l^R_x)$ and $T^P(l^P_x)$, respectively.

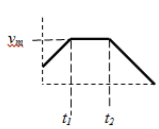
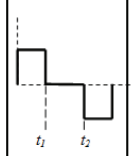
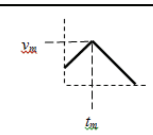
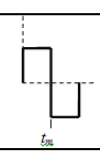
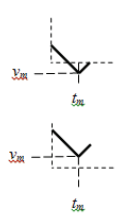
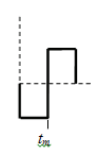
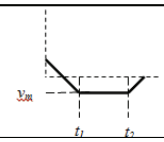
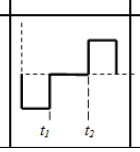
Stage 2: Optimal Dropoff Location, *Y*-Coordinate

The rule is: select the right conveyor belt if $y_{m2} \geq -y_{m1}$, and select the left conveyor belt if $y_{m2} < -y_{m1}$. This rule can be shown to reduce both the maximum and minimum of the reach and place times and is proven formally by Mann et al. (2014).

Stage 3: Optimal Dropoff Location, *X*-Coordinate

As the place phase time T^P and reach phase time T^R depend on the dropoff point x, so does the total traversal time $T(x) = T^P(x) + T^R(x)$. This is shown in figure 3 for two melons with x-coordinates $x_{m1} = 2$ and $x_{m2} = 3$, where the place time, reach time, and total traversal time are plotted as a function of the dropoff point. Notice how the minimum regions of $T^P(x)$ and $T^R(x)$, denoted respectively as M^R and M^P, are intervals that occur near the melon coordinates. This is

Table 1. Times and inputs for the four velocity profiles.

Type	velocity profile	occurs if l is less than	and if l is more than	maximum/ minimum velocity v_m	times	end time T	acceleration input u	$\frac{dT}{dl}$
IV		∞	$\frac{2v_m^2 - v_S^2 - v_E^2}{2a_{max}}$	$v_t + v_{max}$	$t_1 = \frac{v_m - v_S}{a}$ $t_2 = T - \frac{v_m - v_E}{a}$	$\frac{\left((v_m - v_E)^2 + (v_m - v_S)^2\right)}{2v_m a_{max}} + \frac{l}{v_m}$		$\frac{1}{v_m}$
III		$\frac{2v_m^2 - v_S^2 - v_E^2}{2a_{max}}$	$\frac{\lvert v_E^2 - v_S^2 \rvert}{2a_{max}}$	$\sqrt{a_{max} l + \frac{(v_E^2 + v_S^2)}{2}}$	$t_m = \frac{2v_m - v_S - v_E}{a}$	$\frac{2v_m}{a_{max}} - \frac{v_S + v_E}{a_{max}}$		$\frac{1}{v_m}$
II		$\frac{\lvert v_E^2 - v_S^2 \rvert}{2a_{max}}$	$\frac{v_E^2 + v_S^2 - 2v_m^2}{2a_{max}}$	$-\sqrt{\frac{(v_E^2 + v_S^2)}{2} - a_{max} l}$ $if \quad v_E^2 + v_S^2 > 2a_{max} l$ $\sqrt{a_{max} l - \frac{(v_E^2 + v_S^2)}{2}}$ $if \quad v_E^2 + v_S^2 \leq 2a_{max} l$	$t_m = \frac{v_S + v_E - 2v_m}{a}$	$\frac{v_S + v_E}{a_{max}} + \frac{2v_m}{a_{max}}$ $if \quad v_E^2 + v_S^2 > 2a_{max} l$ $\frac{v_S + v_E}{a_{max}} - \frac{2v_m}{a_{max}}$ $if \quad v_E^2 + v_S^2 \leq 2a_{max} l$		$\frac{1}{v_m}$
I		$\frac{v_E^2 + v_S^2 - 2v_m^2}{2a_{max}}$	$-\infty$	$v_t - v_{max}$	$t_1 = \frac{v_S - v_m}{a}$ $t_2 = T - \frac{v_E - v_m}{a}$	$-\frac{l}{v_m} - \frac{\left((v_m - v_E)^2 + (v_m - v_S)^2\right)}{2v_m a_{max}}$		$\frac{1}{v_m}$

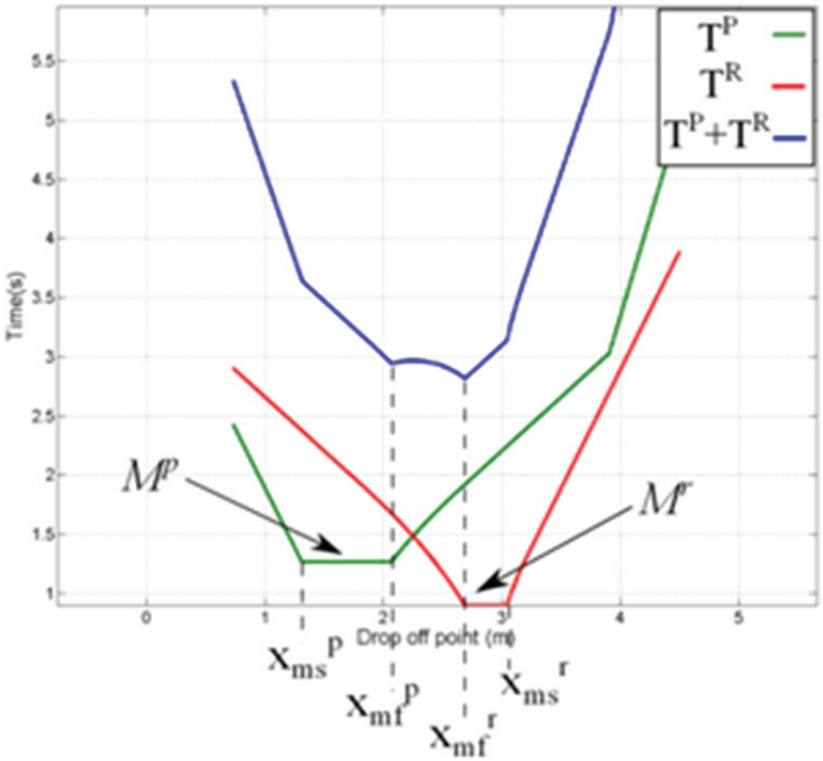


Figure 3. Place time $T^P(x)$, reach time $T^R(x)$, and total traversal time $T(x) = T^P(x) + T^R(x)$ as a function of the dropoff point x between two melons with coordinates $x_{m1} = 2$ and $x_{m2} = 3$. The minima of $T^P(x)$ and $T^R(x)$ are M^P and M^R, and their endpoints are $\{x_{ms}^P, x_{mf}^P\}$ and $\{x_{ms}^R, x_{mf}^R\}$. As shown, one of the endpoints is the minimum argument of $T(x)$.

clearly explained by equation 6; for small place phase or reach phase displacements in the x direction, $T_{yz} > T_x(x)$.

The edges of M^R and M^P are denoted as $\{x_{ms}^P, x_{mf}^P\}$ and $\{x_{ms}^R, x_{mf}^R\}$, respectively. Notice how some of these edges coincide with *arg min* $T^P(x) + T^R(x)$, i.e., the dropoff point that results in minimum total traversal time. This interesting result can be proven using nonsmooth optimization; this minimum is thus the optimal dropoff point x_D. The calculation of x_D is thereby provided explicitly:

Find the set of four points

$$X^* = \{x_{ms}^P, x_{mf}^P, x_{ms}^R, x_{mf}^R\}$$

and then find $\min_{x^* \in X^*} T(x^*)$

This greatly simplifies the optimization procedure from numerical calculation of the nonconvex function $T(x)$ to a simple algebraic calculation of at most four points. A proof of its optimality is presented by Mann et al. (2014).

Stage 4: Time Window for Traversal from Melon to Melon

The minimum time required to traverse from melon i to melon j (T_{ij}) can be calculated for all paths between melons using stages 1 through 3. However, the finite length of the platform restricts the manipulator at any given time to reaching melons only within a certain distance of it. Thus, the aforementioned trajectory can only commence within a certain time window, denoted as TW_{ij}. Denoting $x^*(x_i, x_j, t_S)$ as the minimum-time trajectory from melon i to melon j when starting at time t_S, the definition of TW_{ij} is the set of all start times at melon i under which the robot can feasibly reach melon j. The arc time window can be calculated graphically by first computing the optimal trajectory from stages 1 through 3, and then determining the earliest and latest start time t_S under which this trajectory will strictly satisfy the location constraints of equation 5. This process is illustrated in figure 4 for two representative examples. In the first case (left), the second melon is located ahead of the first melon, so the manipulator must advance forward, while in the second case (right), the second melon is located before the first melon, so the manipulator must move backward, all while remaining between the front and back edges of the advancing platform. Mathematically:

$$TW(i,j) = \{t_S : x^*(x_i, x_j, t_S) < v_t t + L, \quad x^*(x_i, x_j, t_S) > v_t t \quad \forall t_S \le t \le t_E\} \tag{7}$$

Step 2: Compute Picking Sequence to Maximize Number of Melons

Once the set of all time windows and time spans has been calculated, we can proceed with the second step: plan the sequence of melons that results in the maximum number harvested. This can be represented mathematically by the orienteering problem with time windows (OPTW). Orienteering is a game in which the player visits scattered checkpoints; the goal is to visit as many checkpoints as possible before the time has expired. In this application, the robot must collect as many melons as possible by the time it reaches the end of the row.

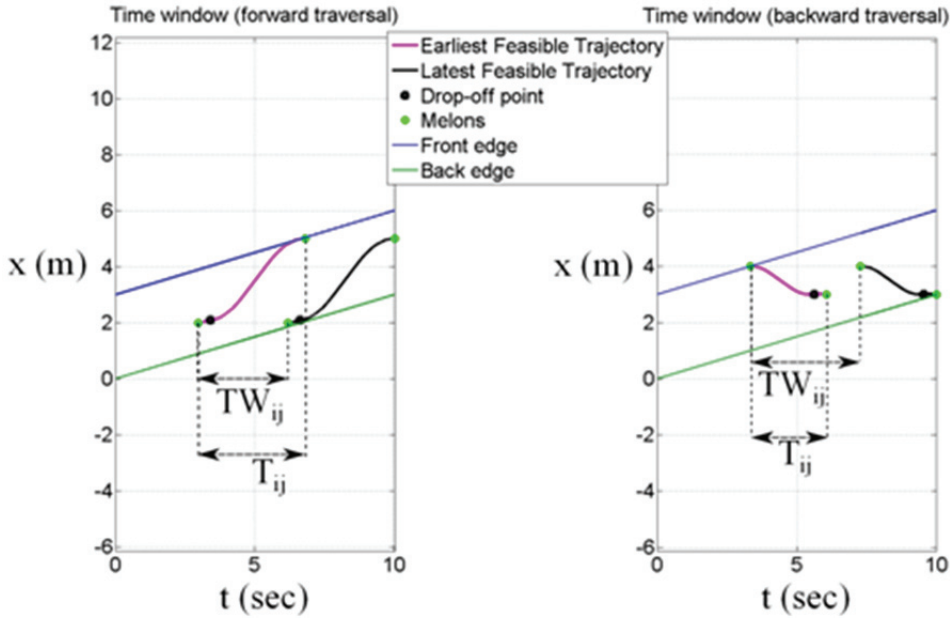


Figure 4. Arc time windows TW_{ij} and time span T_{ij} (denoted by dashed arrows) for forward traversal (left, x_{m1} = 2, x_{m2} = 5) and backward traversal (right, x_{m1} = 4, x_{m2} = 3) from melon *i* to melon *j* on a platform 3 m long. The trajectory must remain behind the front edge (top slanted line) and in front of the back edge (bottom slanted line) at all times.

In OPTW, each checkpoint can only be visited within its given time window. Similarly, the robot can only pick up a melon within the time that the melon is situated within the moving platform. The platform's constant forward motion means that this time is limited to a specified time window. The OPTW for the robotic melon harvester is thus written mathematically in equation 8 as a mixed binary-integer combinatorial optimization problem:

$$
\begin{aligned}
& max \sum_{i=0}^{N}\sum_{j=1}^{N} x_{ij} \quad \text{(number of melons collected)} \\
& \text{subject to} \\
& \sum_{j=1}^{N} x_{0j} = \sum_{i=1}^{N} x_{iE} = 1 \quad \text{(path constraints)} \\
& \sum_{i=0}^{N} x_{ik} = \sum_{j=1}^{N} x_{kj} \quad \forall k \in V \\
& \sum_{j=1}^{N} x_{kj} \le 1 \\
& s_i - max\left(TW_{ij}\right) \le M\left(1 - x_{ij}\right) \quad \forall i,j \in V \\
& \qquad \text{(time window constraints)} \\
& min\left(TW_{ij}\right) - s_i \le M\left(1 - x_{ij}\right) \quad \forall i,j \in V \\
& s_j - s_i - t_{ij} = M\left(1 - x_{ij}\right) \quad \forall i,j \in V \\
& \sum_{i,j \in P} x_{ij} \le |P| - 1 \quad \forall P \subseteq V \quad \text{(subtour elimination)} \\
& x_{ij} \in \{0,1\}, \quad s_i \in R^{+} \quad \forall i,j \in V, \\
& M \text{ is an arbitrary large number}
\end{aligned}
\tag{8}
$$

where the robot begins at node 0, ends at node E, and visits a subset of nodes 1, 2,…, N. The binary variable x_{ij} represents whether the robot traversed from node i to j (1) or not (0). The continuous variable s_i is the time that the robot visits node i. The time window constraints ensure that equation 7 is satisfied for every two consecutive node (melons) visited, while the path constraints ensure that each node is visited no more than once. Subtour elimination is necessary to prevent loops of visited nodes within the graph.

Because of the time windows, this is an example of a resource-constrained problem (RCP), where a resource (i.e., time) is restricted to an interval at each node or arc. While this is an NP hard problem (Irnich and Desaulniers, 2005), the key to finding a solution is to exploit the intervals, or windows, in a manner that narrows down the number of possible optimal paths.

Solution Method: Dynamic Programming

Bellman's principle of optimality, the cornerstone of dynamic programming, states that if a sequence of actions yields an optimum of some quantity for the entire problem, then it must also yield an optimum for all subsequences with respect to the same quantity. Applying this to the OPTW, there are three quantities, or resources, that define a path segment's ability to have an increased harvest yield:

- The number of melons collected N.
- The time remaining τ (= total time – current time t).
- The subset of melons available Q.

where a path segment is an ordered set of N melons plus the starting node [v_0, $v_{(2)}$,…, $v_{(N)}$] in the order that the robot picked them up. Thus, for any two path segments A and B ending at the same melon, A dominates B if all of its quantities are superior, i.e., path B is not Pareto optimal (see Intriligator, 2002, for a description of Pareto optimality):

$$\text{Path } A \succeq \text{Path } B \text{ if } \begin{bmatrix} N(A) \geq N(B) \\ \tau(A) \geq \tau(B) \\ Q(A) \supseteq Q(B) \end{bmatrix} \tag{9}$$

This allows us to prune out any path segment ending at the same node if its number of melons collected, time remaining, and the subset of melons available to it are inferior to some other path segment ending at that node without having to solve the entire problem. This is because it is suboptimal with respect to all of its resources, i.e., it is dominated by at least one other path. However, this principle must take advantage of the time windows in order to rapidly prune out suboptimal sequences. The time windows are used to speed up calculations in two aspects of the optimal sequence algorithm: branching and pruning.

Branching

Path segment A ending at node v_i can be branched out or extended to all nodes v_j that are valid based on the following constraints:

- The current time t_i is within the time window TW_{ij}.

AND

- They do not form a cycle, i.e., v_j is not a member of A (i.e., the same melon cannot be picked twice).

The branching forms a new set of paths $\{P\}$, whose times $t(P)$ are incremented by the corresponding time span T_{ij}. A valid path P that satisfies the aforementioned constraints is expressed mathematically as:

$$\begin{aligned} P = \{A\, v_j\} \text{ such that } t(A) \in TW_{ij}, \\ v_j \notin A,\ t(P) = t(A) + T_{ij} \end{aligned} \tag{10}$$

In a problem without time windows, every single node in a path segment would have to be checked when it is branched out in order to eliminate cycles. With time windows, however, only the nodes for which the current time is within their time window must be examined, as the nodes with expired time windows cannot be extended to. This is accomplished by an array $\tilde{P}_i$ associated with a path A that contains all of the visited nodes within the current time window. When A is extended to a new path P_j, those elements of $\tilde{P}_i$ whose time windows have expired by the new time t_j are deleted; the rest becomes the new array $\tilde{P}_j$:

$$\tilde{P}_j = \left\{\tilde{P}_i,\ v_i \setminus \left\{v_k : t_j\left(\text{Path } P_j\right) > max\left(TW_{jk}\right)\right\}\right\} \tag{11}$$

This process continues for each path extension; thus, only an array of melons within the time window of the last node in each path need be kept.

Pruning

The subset criterion means that a path segment is dominated, and thus pruned out of the tree of paths, only if the melons available to it are a subset of the melons available to another path at the same node. Thus, the more criteria for determining subset dominance, the quicker the paths will be pruned, and the algorithm will run more efficiently. However, by making use of the simple set relation:

$$Q(B) \subseteq Q(A) \quad \Leftrightarrow \quad \bar{Q}(A) \subseteq \bar{Q}(B) \tag{12}$$

the same criterion states that a path is dominated only if the set of melons unavailable to it contains the set of melons unavailable to any other path at the same node. Here is where the time windows come into play; a melon is unavailable to a given path if:

- It is already included in the path.

OR

- The time window of the arc from the last node of a path to the melon has expired.

This yields expanded domination criteria of path A over path B, enabling suboptimal sequences to be eliminated early on:

$$\begin{aligned} Q_i(B) \subseteq Q_i(A) \ \text{ if } \forall v_k \in \{P_i(A) \setminus v_i\}, \\ v_k \in P_i(B) \ \text{ or } \ max\left(TW(i,k)\right) < t_i(B) \end{aligned} \tag{13}$$

A formal proof of the domination is provided by Feillet et al. (2004).

Moving Branch and Prune Algorithm

The branching of each path to all feasible nodes and the pruning of suboptimal nodes constitute the moving branch and prune method (fig. 5). A decision tree is formed from all path extensions, starting from the initial point. All paths that are formed from the tree are immediately added to a priority queue after being branched, with priority given according to the time at the end node. When a path is popped from the priority queue, it is compared to all other paths ending in the same node. If it is dominated per equation 9, then it is pruned from the decision tree; if not, then it is branched out, with the newly formed branched path added to the priority queue. This continues until the row of melons is completely processed.

This is what makes the method "moving." The possible sequences are continuously branched and pruned such that any suboptimal sequence is deleted immediately upon being dominated. Eventually, an optimal path segment emerges. This accommodates online path planning, as an optimal subsequence of melons to pick can be computed with only a limited horizon of melon sensing. This fascinating result is a consequence of the pigeonhole principle: since for each sequence length, a certain number of sequences will be pruned, a clear choice of the most optimal subset of melons to pick will eventually emerge if one looks ahead far enough. A schematic of the algorithm and the accessory data structures are shown in figure 11.

Results

We demonstrate the results of the developed procedure by displaying the minimum traversal times, trajectory, harvesting sequence, and algorithm performance of a simulated melon harvesting robot. The simulation was executed in MATLAB Ver. 7.11 on a Dell Precision T3400 computer with an Intel Core Duo 2.66 GHz CPU. A simulated robotic

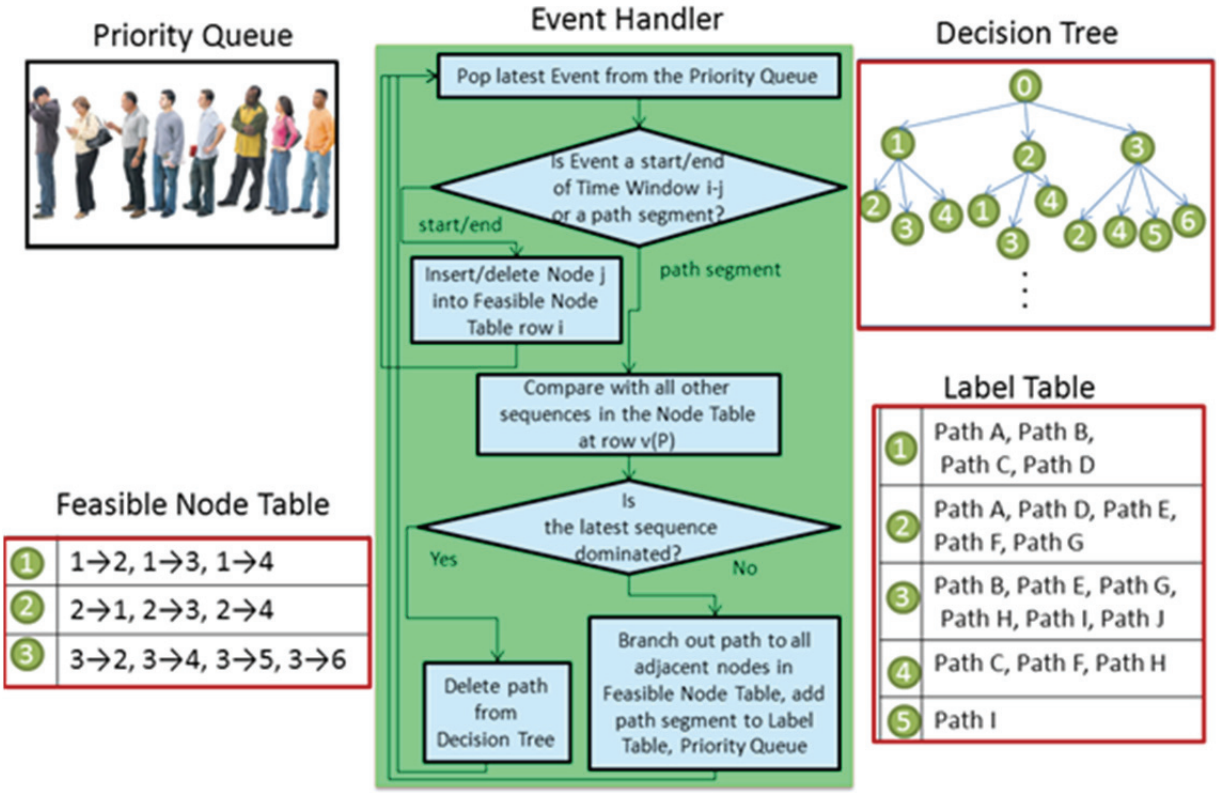


Figure 5. Schematic of the moving branch and prune algorithm and accessory data structures.

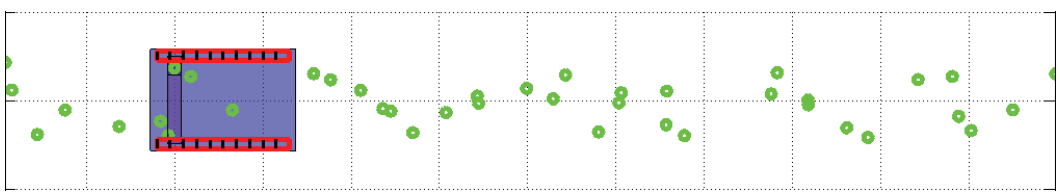


Figure 6. Top view of melon harvesting robot.

melon harvester, 1.8 m wide and 3 m long, was constructed that advanced at a constant speed of 30 cm s^{-1} in a 2 m wide row. The row contained 40 melons randomly scattered with a uniform distribution at an average density of one melon per m^2, as shown in figure 6. The conveyor belts were moving -15 cm s^{-1} relative to the platform. The speed limit of the actuators was 1 m s^{-1}, and their acceleration limit was 0.8 m s^{-2}.

Minimum Time from Melon to Melon

We applied our method to the simulated robot and compared it with the nearest neighbor method, in which the robot selects the next melon based on the smallest time span between melons. The connectivity graph for the system is shown in figure 7, and the melon-to-melon traversal times and time windows are shown in figure 8. The robot can traverse between any two melons that are connected in the connectivity graph. The time window within which it can begin the traversal is shown by the *x*-coordinate of the line segments in figure 8 (see inset), and the traversal time is shown by the *y*-coordinate. Under the specific kinematic conditions of the simulation, the robot succeeded in harvesting 26 of the 40 melons in the field, which, although rather low, is much higher than the nearest neighbor method, which succeeded in harvesting only 16 melons. Also note that the low harvest ratio results from the rapid forward speed of the robot; as this speed is reduced, the harvest ratio increases.

Maximum Harvest Sequence

The optimal trajectory of the manipulator in the preceding example is shown in figure 9. The dropoff points between each melon are calculated as well as the trajectory from point to point that satisfies the motion constraints. The velocity profiles and acceleration inputs for the *x* and *y* actuators are shown in figure 10, displaying a trapezoidal velocity profile and a bang-bang input. Several snapshots of the decision tree formed by the moving branch and prune algorithm are shown in figure 11, with the last shot (fig. 11d) forming an optimal path from start to finish. Notice that as the tree progresses, the optimal path forms gradually from the origin up until a certain point where there are still multiple decisions, i.e., branches. This is a very significant result which means that the harvesting robot can select the optimal harvest sequence as it progresses even without knowledge of the entire field. This system can

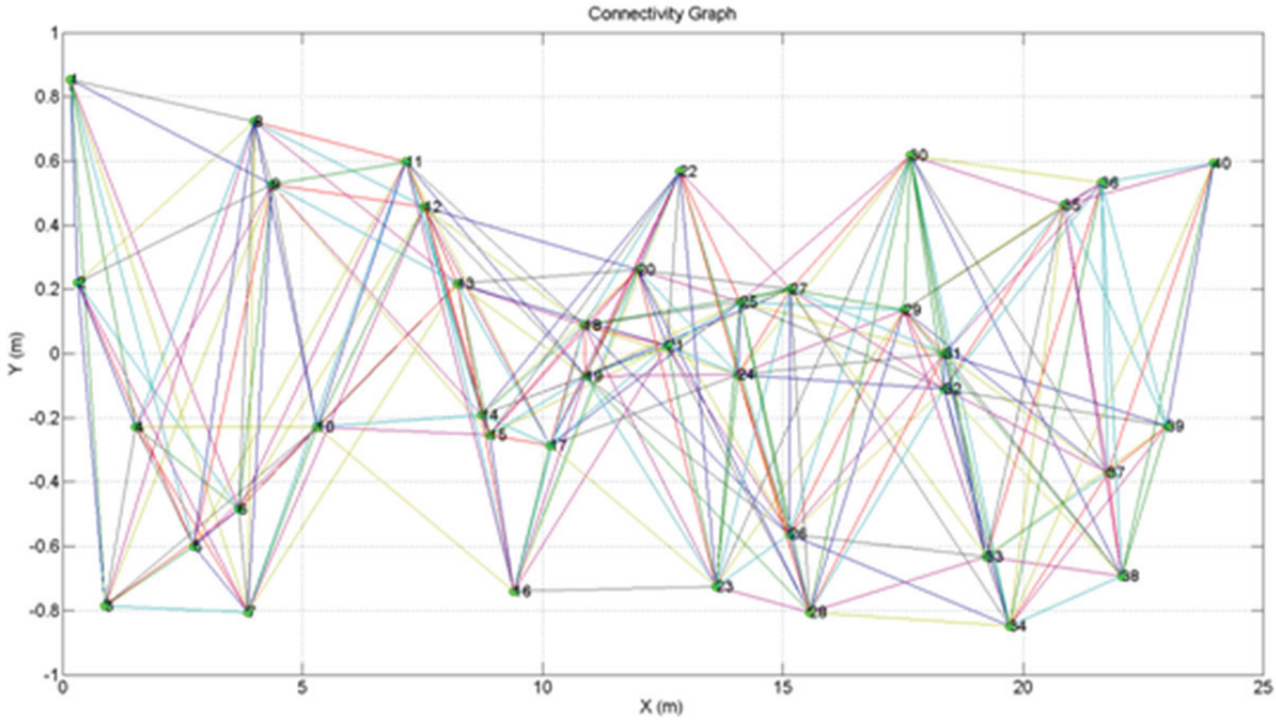


Figure 7. Connectivity graph of melons. Melons are denoted by their series numbers and are shown at their coordinates. All melons connected by a line can be traversed by the robot within some interval of time.

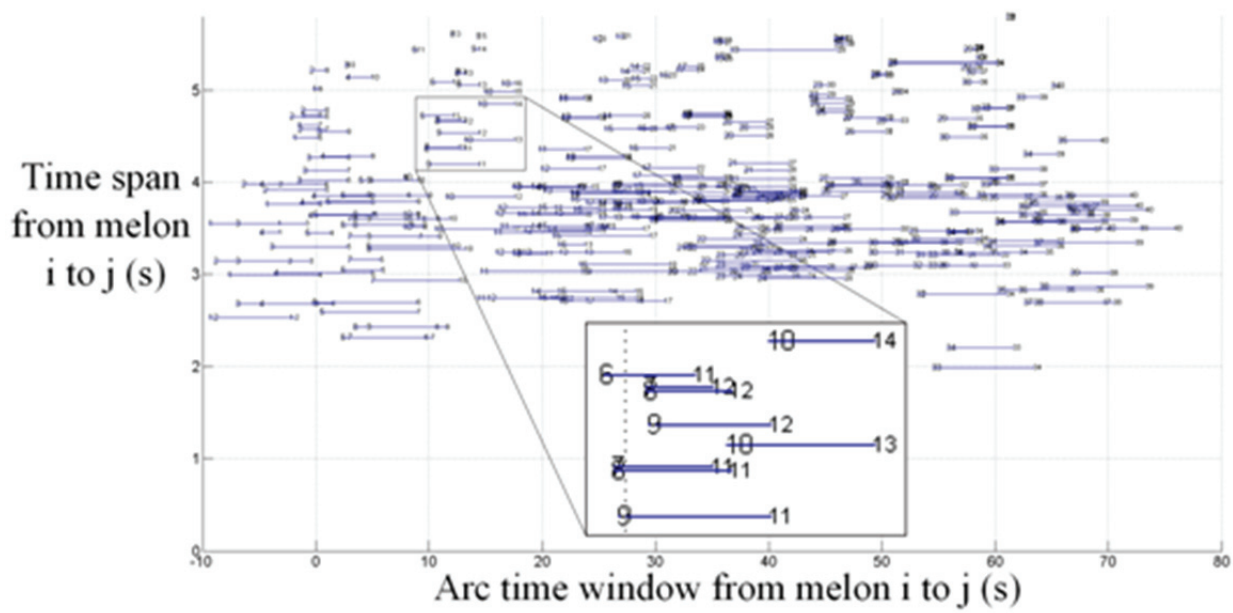


Figure 8. Traversal between melon *i* (left end of line segments) and melon *j* (right end of line segments) must begin within the respective time window (*x*-coordinate) and consume a certain amount of time (*y*-coordinate). The inset shows close-up view of some of the line segments.

therefore be implemented in real time for robots that can sense melon locations only up to a limited distance. On average, a greater number of feasible sequences result in a longer necessary sensing distance. In this example, the necessary sensing distance is about 9 m.

Algorithm Performance

It must be emphasized that the moving branch and prune algorithm yields a strictly optimal picking sequence. This is in contrast to the tree heuristic presented by Kantor and Rosenwein (1992) and the approximation algorithm of Chekuri and Korula (2007), which may yield near-optimal solutions but not strictly optimal solutions. While the decremental state space relaxation strategies of Righini and Salani (2009) are exact, the computation time increases exponentially with the number of nodes. The moving branch and prune method, on the other hand, makes use of time windows to eliminate visited nodes of a path segment

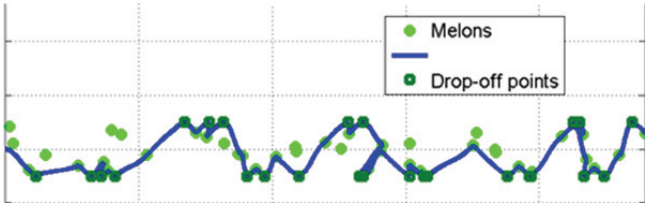


Figure 9. Trajectory of robotic melon harvester with dropoff points.

whose time slots are already expired, thereby decreasing the number of available Pareto optimal points and speeding up the sequence elimination process. This results in a very time-efficient algorithm whose computation time, for a constant field density, increases only linearly with the number of melons, as shown in figure 12 for the same sample robot parameters as above. The linear dependence is yet another advantageous aspect of possible implementation in real time.

The optimal harvest ration clearly depends on the robot's forward speed; the faster the robot advances, the fewer melons it will succeed in picking. This relationship is shown in figure 13 for the sample robot. In this example, in order to obtain a harvest percentage of at least 90%, the platform may only advance 16 cm s^{-1}. This slow speed means that for a reasonably efficient harvest, the actuators must be replaced with more powerful ones or multiple robotic arms need to be deployed. The algorithm thus also serves as a tool for robot design selection and optimization. Deriving the expected dependence of the harvest percentage on the robot parameters is a subject of a future article.

Conclusion and Future Work

A procedure to calculate the trajectory of a melon-harvesting Cartesian robot that results in the maximum number of melons harvested has been presented. The procedure consists of two steps. First, the minimum-time trajectory required to traverse between any two melons is calculated, and the time window in which it can occur is determined using analytical and graphical methods. This enables the problem to be expressed as one of orienteering with time windows, the solution of which is the second step. Dynamic programming is applied in manner that exploits the time windows to eliminate suboptimal sequences more efficiently. This can be carried out online, without the need to solve the entire problem at once, yielding the moving branch and prune method. The result is an exactly harvest-maximal solution that requires relatively little computation time and yields a much greater outcome than heuristics such as the nearest neighbor. It can also be implemented in robotic systems that carry only short-range sensing capabilities rather than *a priori* mapping of all targets due to both linear computation time dependence and a finite horizon of optimality. The mathematical theory that explains this significant result in more depth will be the subject of a future article.

An actual robot prototype for validation purposes is currently under construction. It will be used to verify the optimization methods presented in this article, as well as serving as a platform for incorporating practical implementation issues such as a sloped field, vibrations, and uncertainties. Optimizing a robotic fruit harvester that takes these issues into account is a promising topic of future research.

Acknowledgements

The authors would like to thank the Israeli Ministry of Agriculture and the Irwin and Joan Jacobs Graduate School of Technion for their partial support of this research.

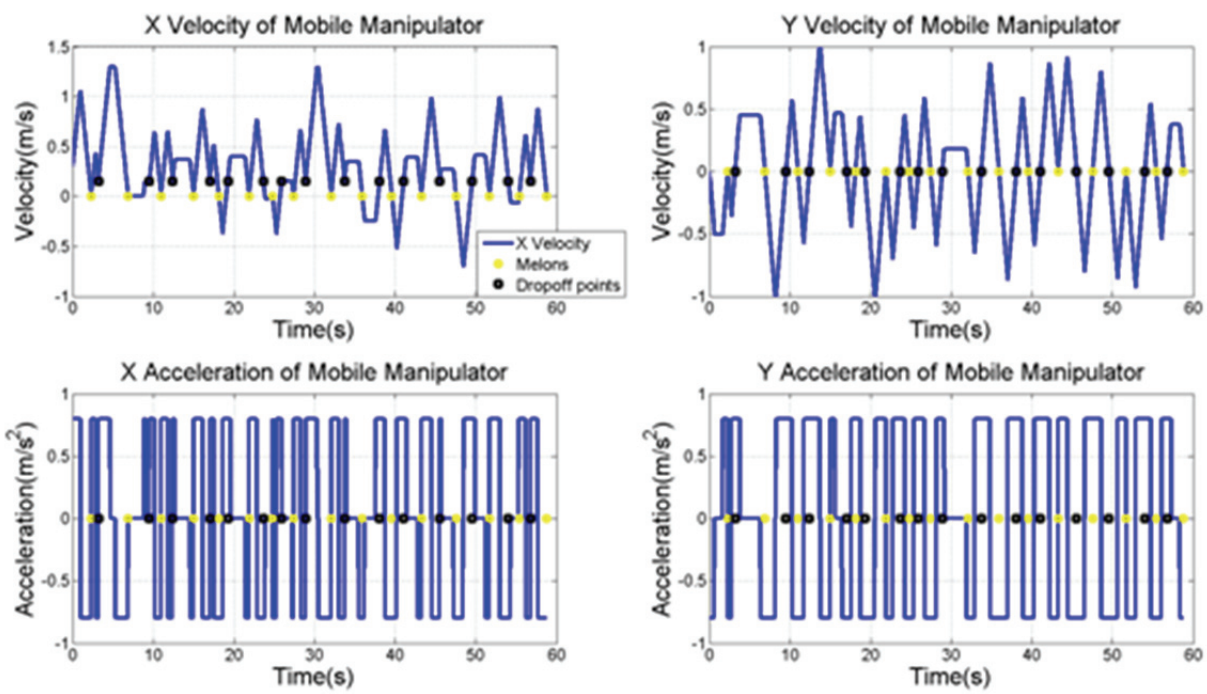


Figure 10. Velocity and acceleration profiles for the *x* and *y* actuators. The velocity is trapezoidal, and the acceleration is bang-bang.

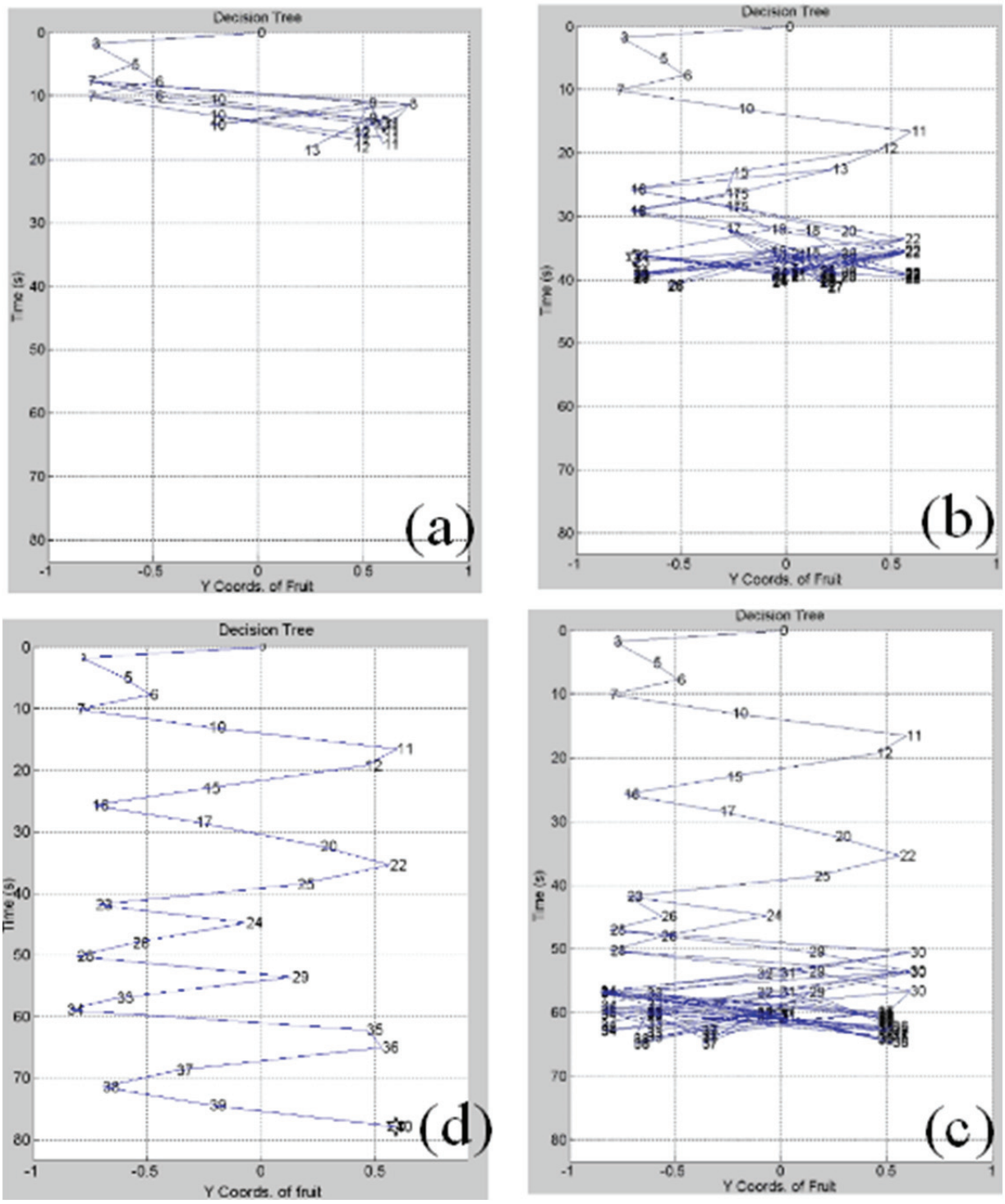


Figure 11. The decision tree at progressive times from (a) to (d). The optimal path forms gradually from the origin.

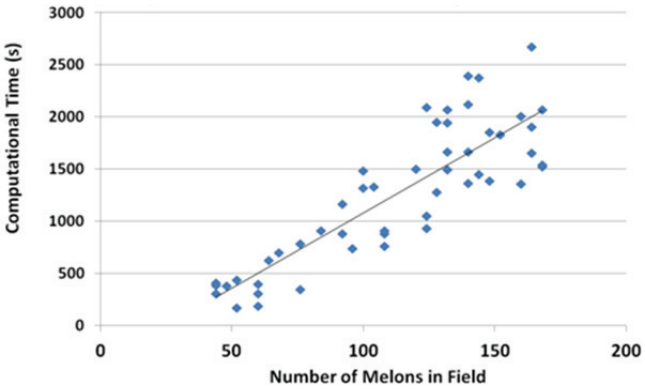


Figure 12. Time required by the moving branch and prune algorithm to compute the optimal sequence of melons to pick as a function of the number of melons in the row with 1.5 melons per m^2. This dependence is generally linear.

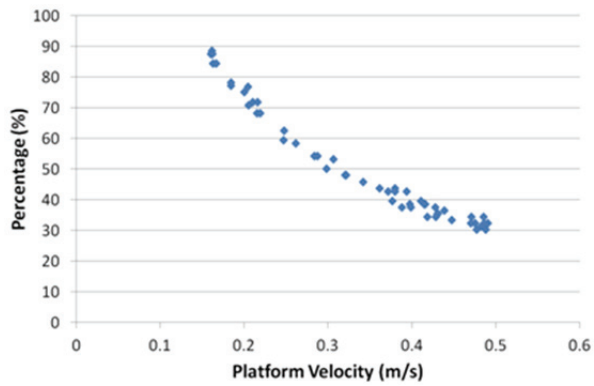


Figure 13. Percentage of melons harvested for different robot platform speeds in a field with 1.5 melons per m^2. The very small deviation from the average suggests a simpler model of robotic harvesting.

References

Bansal, N., Chawla, S., Blum, A., & Meyerson, A. (2004). Approximation algorithms for deadline TSP and vehicle routing with time windows. In *Proc. 36th ACM Symp. on Theory of Computing*, 166-174. New York, N.Y.: Association for Computing Machinery.

Belforte, G., Gay, P. & Aimonino, D. (2006). Chapter 35: Robotics for improving quality, safety, and productivity in intensive agriculture: Challenges and opportunities. In *Industrial Robotics: Programming, Simulation, and Applications*, 677-690. Augsburg, Germany: Pro Literatur Verlag.

Chekuri, C., & Korula, N. (2007). Approximation algorithms for orienteering with time windows. arXiv 0711:4825.

Edan, Y., & Miles, G. (1993). Design of an agricultural robot for harvesting melons. *Trans. ASAE, 36*(2), 593-603. http://dx.doi.org/10.13031/2013.28377.

Edan, Y., & Miles, G. (1994). Systems engineering of agricultural robot design. *IEEE Trans. Systems, Man, and Cybernetics, 24*(8), 1259-1265. http://dx.doi.org/10.1109/21.299707.

Edan, Y., Flash, T., Shmulevich, I., Sarig, Y., & Peiper, U. (1990). An algorithm defining the motions of a citrus picking robot. *J. Agric. Eng. Res. 46*, 259-273. http://dx.doi.org/10.1016/S0021-8634(05)80131-3.

Edan, Y., Flash, T., Peiper, U., Shmulevich, I., & Sarig, Y. (1991). Near-minimum-time task planning for fruit-picking robots. *IEEE Trans. Robotics and Automation, 7*(1), 48-56. http://dx.doi.org/10.1109/70.68069.

Edan, Y., Rogozin, D., Flash, T., & Miles, G. (2000). Robotic melon harvesting. *IEEE Trans. Robotics and Automation, 16*(6), 831-834. http://dx.doi.org/10.1109/70.897793.

Edan, Y., Engel, B., & Miles, G. (2003). Intelligent control system simulation of an agricultural robot. *J. Intel. Robotic Systems, 8*(2), 267-284. http://dx.doi.org/10.1007/BF01257998.

Edan, Y., Han, S., & Kondo, N. (2009). Chapter 63: Automation in agriculture. In *Springer Handbook of Automation*, 1095-1128. Berlin, Germany: Springer Verlag.

Feillet, D., Dejax, P., Gendreau, M., & Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks, 44*(3), 216-229. http://dx.doi.org/10.1002/net.20033.

Friedman, R. (2010). Israel to let in another 3,700 foreign workers. *Jerusalem Post* (May 4). Retrieved from: www.jpost.com/Israel/Israel-to-let-in-another-3700-foreign-workers.

Intriligator, M. (2002). Chapter 10: Welfare economics. In *Mathematical Optimization and Economic Theory*, 258-290. Philadelphia, Pa.: Society for Industrial and Applied Mathematics.

Irnich, S., & Desaulniers, G. (2005). Chapter 2: Shortest path problems with resource constraints. In *Column Generation*, 33-65. New York, N.Y.: Springer.

Jazar, R. (2010). Chapter 14: Time optimal control. In *Theory of Applied Robotics: Kinematics, Dynamics, and Control*, 791-826. 2nd ed. New York, N.Y.: Springer.

Kantor, M., & Rosenwein, M. (1992). The orienteering problem with time windows. *J. Oper. Res. Soc., 43*(6), 629-635.

Kassler, M. (2001). Agricultural automation in the new millennium. *Computers and Electronics in Agric., 30*(1-3), 237-240. http://dx.doi.org/10.1016/S0168-1699(00)00167-8.

Korte, B., & Vygen, J. (2008). Chapter 21: The traveling salesman problem. In *Combinatorial Optimization: Theory and Algorithms*, 473-506. Berlin, Germany: Springer Verlag.

Makela, M., & Neittaanmaki, P. (1992). *Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control*. Singapore: World Scientific Publishing.

Mann, M., Shmulevitch, I., Rubinstein, D., & Zion, B. (2014). Minimum time kinematic motions of a Cartesian mobile manipulator for a fruit harvesting robot. *ASME J. Systems, Dynamics, and Control* (accepted).

Naidu, D. (2003). *Optimal Control Systems.* Boca Raton, Fla.: CRC Press.

Pedersen, S., Fountas, S., Have, H., & Blackmore, B. (2006). Agricultural robots: System analysis and economic feasibility. *Prec. Agric., 7*(4), 295-308. http://dx.doi.org/10.1007/s11119-006-9014-9.

Pedersen, S., Fountas, S., & Blackmore, B. (2008). Chapter 21: Service robot applications. In *Agricultural Robots: Applications and Economic Perspective*, 369-382. Rijeka, Croatia: InTech Education and Publishing.

Perry, T. (2009). Fruitless: A strawberry-picking robot won't be displacing farmworkers any time soon. *IEEE Spectrum* (January), 55-58. http://dx.doi.org/10.1109/MSPEC.2009.4734317.

Righini, G., & Salani, M. (2006). Dynamic programming for the orienteering problem with time windows. Milan, Italy: University of Milan, Department of Information Technology.

Righini, G., & Salani, M. (2009). Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers and Oper. Res., 36*(4), 1191-1203. http://dx.doi.org/10.1016/j.cor.2008.01.003.

Sethi, S., & Thompson, G. (2006). *Optimal Control Theory: Applications to Management Science and Economics*. 2nd ed. Dordrecht, The Netherlands: Kluwer Academic.

Shafir, M. (2006). Agriculture in Israel 2004. Jerusalem, Israel: Israel Central Bureau of Statistics. Retrieved from: www.cbs.gov.il/statistical/agri04_e.pdf.

Shin, K., & McKay, N. (1986). Selection of near-minimum time geometric paths for robotic manipulators. *IEEE Trans. Auto. Control, 31*(6), 501-511. http://dx.doi.org/10.1109/TAC.1986.1104316.

Srinivasan, N. (2007). Profitability in agriculture and inclusive growth. *CAB Calling* (January-March), 18-22. Maharashtra, India: College of Agricultural Banking.

Tsiligirides, T. (1984). Heuristic methods applied to orienteering. *J. Oper. Res. Soc., 35*(9), 797-809.

Vansteenwegen, P., Souffriau, W., & van Oudheusen, D. (2011). The orienteering problem: A survey. *European J. Oper. Res., 209*(1), 1-10. http://dx.doi.org/10.1016/j.ejor.2010.03.045.

Westberry, G. (2009). Production costs. In *Cantaloupe and Specialty Melons*, 24. Bulletin 1179. Athens, Ga.: University of Georgia Cooperative Extension. Retrieved from: www.caes.uga.edu/publications/pubdetail.cfm?pk_id=6278.