POLITECNICO DI TORINO Repository ISTITUZIONALE

An integer linear programming model for efficient scheduling of UGV tasks in precision agriculture under human supervision

Original

An integer linear programming model for efficient scheduling of UGV tasks in precision agriculture under human supervision / FOTIO TIOTSOP, Lohic; Servetti, Antonio; Masala, Enrico. - In: COMPUTERS & OPERATIONS RESEARCH. - ISSN 0305-0548. - STAMPA. - 114:(2020). [10.1016/j.cor.2019.104826]

Availability: This version is available at: 11583/2764434 since: 2020-07-15T22:08:40Z

Publisher: Elsevier

Published DOI:10.1016/j.cor.2019.104826

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright Elsevier postprint/Author's Accepted Manuscript

© 2020. This manuscript version is made available under the CC-BY-NC-ND 4.0 license http://creativecommons.org/licenses/by-nc-nd/4.0/.The final authenticated version is available online at: http://dx.doi.org/10.1016/j.cor.2019.104826

(Article begins on next page)

An Integer Linear Programming Model for Efficient Scheduling of UGV Tasks in Precision Agriculture under Human Supervision

Lohic Fotio Tiotsop, Antonio Servetti, Enrico Masala

Control and Computer Engineering Department Politecnico di Torino Corso Duca degli Abruzzi 24, 10129 Torino, Italy

Abstract

In precision agriculture more and more robots are being used to perform tasks that may include some farming activities, such as pruning, inspection or spraying, assigned to the robot as a result of a previous analysis activity or autonomously identified by the machine itself. In this sensitive scenario, reporting difficult situations to a decision maker, e.g., a human operator or some sophisticated software tools that cannot be integrated with the robot, could be useful to perform the correct action that the machine has to execute. Unfortunately, this key aspect is still neglected in current literature that focuses, instead, on fully automated operations by robots. Moreover, it is necessary to consider that in rural areas it often happens that successful data communication can only be achieved in certain locations in the field. In this context, we aim to address all the previous shortcomings by formulating a more comprehensive optimization problem, which also models the necessity to report to a central location and get instructions on the task to be done before proceeding to perform each action. After presenting two alternative analytical formulations of the problem, i.e. an integer linear programming model (ILP) and a mixed integer linear programming model, we propose a branch and bound algorithm that is guaranteed to find the global minimum cost solution in terms of navigation time. Simulation

Email address: lohic.fotiotiotsop@polito.it, antonio.servetti@polito.it, enrico.masala@polito.it (Lohic Fotio Tiotsop, Antonio Servetti, Enrico Masala)

results show that our proposed algorithm performs about 20 to 30 times faster with respect to commercial linear programming solvers using any of the two analytical models proposed. Moreover, we also propose further improvements to reduce computational time while maintaining solution optimality. Finally, some insight into the development of future heuristics is given by analyzing the speed of convergence towards the optimal solution.

Keywords: data acquisition, farming activities, UGV navigation, integer linear programming, service robotics

1 1. Introduction

The amount of robots employed in agriculture applications is constantly increasing. Two types of robots are typically used: unmanned ground vehicles (UGV) and unmanned aerial vehicles (UAV) [1]. Examples are rovers that navigate the cultivation to perform specific agriculture tasks, e.g., inspection or physical operations such as collecting samples, pruning, spraying [2]. UAVs, instead, are mainly used for imagery tasks, i.e., to take pictures for later processing in order to understand the status and requirements of different parts of the cultivation [3, 4]. Collaboration between UAVs and UGVs is also being actively explored [5, 6].

Despite some operations can be done autonomously by the robots, sometimes 11 communication with a central location is needed, so that a human operator or 12 an algorithm, more sophisticated compared to the one that can run on the 13 robot, can be used in the loop to take decisions and act accordingly in a spe-14 cific terrain position. In fact, a recent survey about the usage of agricultural 15 robots [7] pointed out the lack of models considering such aspect in the current 16 literature, while at the same time highlighting its necessity to correctly model 17 actual problems. 18

¹⁹ Note that this work relies on an agricultural scenario to present the models ²⁰ just because it seems to be one of the most challenging real applications, but ²¹ it can also cover other real situations, such as the one described in the follow-

ing. In case of industrial operations being performed by robots with human 22 supervision, there could be A (A typically large) jobs located in different places 23 to be performed by a robot and a supervisor operator may just have a rough 24 knowledge of what is the exact job to be done at each location. There are also B25 locations where the robots can collect what is needed to perform a job. The pro-26 posed models can be used to schedule the A jobs minimizing the time required 27 to complete them. At any job location the robot first performs an inspection, 28 then moves to a point of type B, if necessary, by sending in the meanwhile a 29 report to the supervisor operator that decides what to do and what is needed 30 to perform the job and informs the robots that then act accordingly. Another 31 example is repair companies: they may rely on the proposed models to sched-32 ule their logistic operations. The customer locations would represent points of 33 interests and the company warehouses are the points to be visited to perform 34 repairs at the point of interests. More precisely, when an operator leaves the 35 main warehouse in the morning, at each customer location he/she either fixes 36 the problem (if the description was exhaustive) or collects information about 37 the problem and what is needed, then visits a company warehouse in order 38 to pick up all he needs and go back later to the customer location to fix the 39 problem. More in general, our models can be used for efficiently planning oper-40 ations requiring data acquisition, data elaboration or transmission, and finally 41 an action. 42

In this work we focus on the specific agriculture scenario considering and 43 modelling the constraint imposed by communicating with a decision maker, 44 e.g., a human operator, before performing physical actions. Considering the 45 optimization problem, we assume to have a rough preliminary knowledge about 46 the terrain and how a UGV can move in the area. In other words, obstacle 47 positions and possible navigation paths are known. This can be obtained by 48 previous knowledge or by means of a survey, e.g., made using a UAV or other 49 similar technologies. During the survey, points of interests (POIs) are also 50 identified, i.e., areas to be visited by the UGV in order to gather information, 51 decide what to do and act accordingly. Since the decision of the action to 52

perform in each POI might be difficult to take only relying on the on-board processing capabilities or simply might require human judgment not available through algorithms, we assume that after the UGV has gathered information by moving to the POI and taking, e.g., a picture, it must communicate with a central location to get an answer about the farming action to perform. The UGV must then perform the action returning to the POI, if necessary.

However, it is well know that communications, including wireless, in rural 59 areas might be difficult [8], mainly due to two reasons. The first reason is, 60 sometimes, the presence of massive physical constraints, such as hills, rocks, 61 subsidence [9]. The second reason is the typically scarce coverage of 3G/4G62 cellular technology, since there is small incentive for network operators to pro-63 vide good quality coverage of such areas due to, e.g., the low people density. As 64 consequence, depending on the physical position, either no communication is а 65 possible or only wireless communications are available but with locally-deployed 66 infrastructure such as Wi-Fi access points or similar systems, potentially tuned 67 to communicate over longer distance compared to the usual home devices. In 68 our scenario, we assume that we roughly know that in certain places on the 69 terrain communication is possible, whereas in others this is not possible at all. 70 Such information could come as a result of a previous survey, or considering the 71 relative position of the devices, the terrain topography and similar features. 72

Our goal is to provide an efficient algorithm to solve the navigation problem 73 of the UGV on the terrain while minimizing the time needed to perform all the 74 tasks in the previously determined POIs, considering that before performing a 75 task in a POI it has to i) visit the POI and collect some data about its state, 76 ii) visit a location where those data can be transmitted and instructions on 77 the task to be done are received, iii) go to the POI and perform the action. 78 Obviously the UGV does not have to perform the three steps sequentially for 79 each POI, but it can visit multiple POIs before moving to a place where it 80 can communicate, and then it can go back to each one of them, in any order, 81 to execute the received instructions. Minimizing the time taken by the UGV is 82 economically advantageous for many reasons. For instance, if the UGV is rented, 83

it can operate in more places therefore serving more than one customer, thus
the company renting it can maximize its earnings. Conversely, if the UGV needs
to be bought, a lower number of them is needed to perform all the activities
in the timeframe required by the specific agriculture scenario so that the UGV
could potentially be employed for other activities.

This paper significantly extends our preliminary work [10]. In particular, 89 we present an improved version of our proposed branch and bound algorithm 90 which relies on the computation of a lower bound to reduce the execution time. 91 Moreover, an additional mixed ILP model employing dummy vertices is pre-92 sented for comparison purposes. Finally, this paper includes an extensive set 93 of performance results on several different graph instances, i.e. fifty instances 94 for each level of complexity, both using the two ILP models and the proposed 95 algorithm. 96

The paper is organized as follows. Section 2 reviews related work in the 97 field, whereas Section 3 explains how to represent the graph which is the basic 98 input data for our algorithm. Then, the optimization problem is analytically 99 formulated in Section 4 as two ILP problems, using two different approaches, to 100 study its characteristics. In Section 5 a specific algorithm for the problem is de-101 signed and implemented in the form of a branch and bound algorithm, and some 102 efficient pruning and lower bounding strategies are proposed. Section 6 shows 103 some practical experimental results on realistic graphs, followed by conclusions 104 in Section 7. 105

106 2. Related work

Sensing technologies are more and more used in agricultural applications to perform crop monitoring [11, 12]. For instance, imaging sensors in the visible spectrum [13] or at other wavelengths [14] are often used. Image data can then be elaborated to detect cultivation type, e.g., vineyard [15], or to detect single plants and potential anomalies [16]. A more comprehensive overview of such technologies can be found in [17].

Potential issues can be identified and transformed into a set of POIs so that, 113 on the basis of the cultivation map, a UGV can move in such positions in order 114 to perform a more detailed analysis, e.g., taking closer pictures or perform local 115 measures with sensors, to later perform some specific physical action [18, 19]. 116 For the purpose of solving the UGV movement problem, the cultivation field 117 can be abstracted in the form of a graph with nodes (i.e., the POIs) connected 118 by edges whose weight represent the cost, in terms of time, needed to move from 119 one point to the other. 120

Graph visiting problems have been well investigated from the theoretical 121 point of view. The travelling salesman problem (TSP) or more in general the 122 vehicle routing problem (VRP) and some variants have been intensively studied 123 in literature [20, 21, 22, 23, 24, 25]. In short, they consider a set of vehicles 124 that can transport goods and a graph whose nodes represent a set of customers, 125 each one characterized by a given demand. Each edge represents the cost that 126 should be sustained to move from a customer to another. The VRP aims at 127 determining which customers should be served by any vehicle and the schedule 128 of the operations of any vehicle in order to minimize the total cost satisfying 129 the capacity constraint of each vehicle and the demand of each customer. In 130 case of a single vehicle, the problem is reduced to the TSP. 131

Reducing field work time in agriculture by optimally scheduling agricultural 132 tasks is often assimilated in the literature to graph visiting problem. In [26, 133 27, 28] the authors explain how the VRP can be used to gain efficiency in field 134 logistics. In particular in [26] the numerical experiments show that it is possible 135 to save up to 32% of the time if the operations are guided by the solution of 136 the proposed VRP instead of an intuitive approach. The TSP is used in [29] to 137 schedule the farming activities in fields characterized by the presence of a large 138 number of obstacles. 139

While the goal of these works is to minimize the time needed to complete the assigned farming tasks, similarly to our navigation problem, nevertheless they are substantially different in the fact that they do not consider the communication aspect and thus the interaction with an external decision maker, e.g., a human operator or any other external tool, whose presence coordinates
the operations and addresses critical situations.

Some VRP variants are somehow more similar to our case, for instance the 146 so called vehicle routing problem with intermediate stops (VRP-IS) [30]. An 147 intermediate stop (IS) is a stop that occurs at a node whose visit is neces-148 sary to keep the vehicle operational or to perform any other action necessary 149 to pursuit the main task assigned to the vehicle. ISs have been considered 150 for replenishment [31], unloading of waste [32], refueling [33, 34], rest [35] and 151 synchronization requirements [36]. Similarly to the VRP-IS, our problem con-152 siders, besides the POIs where physical actions are required, the points covered 153 by wireless network where the UGV might stop to transfer the data collected 154 until that point and get instructions about the correct actions to perform later 155 at the POIs. In our problem the UGV is hence constrained to visit each POIs 156 again once the data acquired during the first visit have been transmitted. This 157 clearly introduces further constraints compared to the cited works. 158

The more practical problem of computing the exact physical path corre-159 sponding to each edge in our graph is addressed in the robotics research field 160 by works focusing on robot navigation problems. For instance, one of the most 161 common problems is the so-called path-planning, in which a trajectory must be 162 computed so that a robot can move to a certain target position while fulfilling 163 certain constraints, i.e., avoiding obstacles and forbidden or dangerous condi-164 tions [37, 38, 39]. More complex constraints can be taken into account, e.g., the 165 slope of the terrain, as in [40] that presents the case of a UGV that needs to 166 visit all the rows in a vineyard. 167

Once edge weights and node positions are known, our problem resembles (but it is not the same) the Steiner TSP (STSP) [41]. Given a graph, a cost for each edge of the graph and a subset of nodes that represent customers, the STSP aims at finding the minimum-cost tour that passes through each customer node. Edges may be traversed more than once, and nodes visited more than once, if necessary. Such a problem, despite being somehow similar to ours, presents an important difference. In that problem, all clients have to be visited at least once. In ours, we require to visit at least *twice* the POI nodes making sure that at least one of the visits at a given POI occurs *after* the transmission of the related information. This further increases the complexity, since time variables need to be introduced in order to manage the chronology of the tasks.

179 3. Graph representation

In our work we assume that a picture has been used to derive a graph that 180 represents the field in which the UGV must operate. The task of creating a 181 graph from a picture by detecting the different types of areas and how they are 182 connected and which is the optimal movement path between different areas is 183 a problem well addressed in literature (see [42, 43, 44, 45]). Therefore, here 184 we assume that such a graph is available. In more details, in our scenario 185 the graph includes nodes, which correspond to physical locations in the terrain. 186 Such nodes are connected with edges whose weight represents the cost of moving 187 from one node to the other. Obstacles and possible movement paths are already 188 modeled in such weights. 189

¹⁹⁰ Concerning nodes, they can be classified into three types.

- Type A: The nodes that we want to visit, in which a physical operation has to be performed by the robot.
- Type B: The nodes in which we are sure that wireless communication can be established in order to both transmit the information collected at previous 'A' nodes and receive instructions about what to do in those nodes.
- Type AB: The nodes with the characteristics of both 'A' and 'B'. When visited, the physical operation can be immediately performed after data communication has taken place, since it is possible to immediately communicate and receive instructions.
- Detecting 'A' nodes heavily depends on the specific task to do (e.g., detecting potential weeds, or places where plant pruning operations might be necessary).

²⁰³ The same applies for 'B' nodes, in which wireless communication models, com-

²⁰⁴ bined with the terrain geography can be used to determine the possible coverage,

²⁰⁵ or maybe the information is simply available from a previous survey.

206 4. Mathematical formulations

In this section two alternative models of the problem are presented. Both 207 models need to address the problem of modeling the movements of the UGV 208 that needs to traverse nodes and edges an unknown number of times. The first 209 formulation addresses the issue by means of an ordered set of displacements (i.e., 210 a movement from a node to another) that the UGV must follow to complete 211 the required tasks. In this formulation the UGV can pass more than one time 212 on the same node, whereas in the second model this condition is considered by 213 means of dummy vertices (i.e., replica nodes to consider multiple visits). 214

215 4.1. Displacement-based formulation (Model I)

- ²¹⁶ Let us define the following elements:
- \mathcal{A} : the set of nodes to visit (Type 'A' or 'AB')
- \mathcal{B} : the set of nodes where wireless communication can be established (Type 'B' or 'AB')
- $\mathcal{V} = \mathcal{A} \cup \mathcal{B}$ set of all nodes
- \mathcal{K} : the set of displacements of the UGV
- \mathcal{E} : the set of edges connecting the different nodes
- A, B, E and K respectively the cardinality of $\mathcal{A}, \mathcal{B}, \mathcal{E}$ and \mathcal{K}
- *s* the depot node or the central location
- t_{ij} : the time required to cover the edge $(i, j) \in \mathcal{E}$
- x_{ij}^k : a boolean variable equal to 1 if during its k-th displacement the UGV moves along the edge $(i, j) \in \mathcal{E}$

• y_i^k : a boolean variable equal to 1 if during the k-th displacement the UGV transfers the data collected from the node $i \in A$

The problem is formulated as follows

$$\min_{x,y} \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{E}} x_{ij}^k t_{ij} \tag{1}$$

230 st:

228

229

$$\sum_{\{j \in \mathcal{A} \cup \mathcal{B}: (s,j) \in \mathcal{E}\}} x_{sj}^1 = \sum_{\{i \in \mathcal{A} \cup \mathcal{B}: (i,s) \in \mathcal{E}\}} x_{is}^K = 1,$$
(2)

$$\sum_{(i,j)\in\mathcal{E}} x_{ij}^k = 1 \quad \forall k \in \mathcal{K},$$
(3)

$$\sum_{k \in \mathcal{K}} y_i^k = 1 \quad \forall i \in \mathcal{A}, \tag{4}$$

$$\sum_{k \in \mathcal{K}} \sum_{\{i \in \mathcal{A} \cup \mathcal{B}: (i,j) \in \mathcal{E}\}} x_{ij}^k \ge \begin{cases} 2 & \forall j \in \mathcal{A} \setminus \mathcal{B}, \\ 1 & \forall j \in \mathcal{A} \cap \mathcal{B}, \end{cases}$$
(5)

$$x_{ij}^{k-1} \le \sum_{\{i \in \mathcal{A} \cup \mathcal{B}: (j,i) \in \mathcal{E}\}} x_{ji}^k \quad \forall (i,j) \in \mathcal{E} \ \forall k \in (\mathcal{K} \setminus \{1\}),$$
(6)

$$y_i^k \le \sum_{\{j \in \mathcal{B}: (i,j) \in \mathcal{E}\}} x_{ij}^k \quad \forall i \in \mathcal{A} \ \forall k \in \mathcal{K},$$
(7)

$$y_j^k \le \sum_{1 \le t \le k} \sum_{\{i \in \mathcal{A} \cup \mathcal{B}: (i,j) \in \mathcal{E}\}} x_{ij}^t \quad \forall j \in \mathcal{A} \quad \forall k \in \mathcal{K},$$
(8)

$$y_{j}^{k} \leq \begin{cases} \sum_{k+1 \leq t \leq K} \sum_{\{i \in \mathcal{A} \cup \mathcal{B}: (i,j) \in \mathcal{E}\}} x_{ij}^{t} & \forall j \in \mathcal{A} \setminus \mathcal{B} \ \forall k \in \mathcal{K} \setminus \{\mathbf{K}\}, \\ 0 & \forall j \in \mathcal{A} \setminus \mathcal{B} \ , k = K \end{cases}$$
(9)

$$x_{ij}^k \in \{0,1\} \quad \forall (i,j) \in \mathcal{E} \quad \forall k \in \mathcal{K},$$

$$(10)$$

$$y_i^k \in \{0, 1\} \quad \forall i \in \mathcal{A} \quad \forall k \in \mathcal{K},$$

$$(11)$$

Eq. (1) requires the minimization of the total time required to perform all 231 the operations, Eq. (2) imposes that the path of the UGV should start and 232 end at the depot. The constraints in Eq. (3) ensure that each displacement 233 of the UGV corresponds to one edge in the solution and Eq. (4) requires that 234 the data collected from any node $i \in \mathcal{A}$ are transmitted once and only once. 235 The constraints in Eq. (5) ensure that the UGV visits, i) at least twice, the 236 nodes of interest in the set $\mathcal{A} \setminus \mathcal{B}$ where it is not possible to communicate and 23 ii) at least once, the other nodes of interest in the set $\mathcal{A} \cap \mathcal{B}$. Eq. (6) are flow 238 conservation constraints. The fact that the UGV might transfer from a node 239 only if such node belongs to the set \mathcal{B} is given by Eq. (7). Eq. (8) requires that 240 the UGV transfers the information related to a given node in the set \mathcal{A} only if 241 such node has already been visited, Eq.(9) requires that, once the information 242 corresponding to the visit of a node in the set $\mathcal{A} \setminus \mathcal{B}$ where communication is not 243 possible has been transmitted, the UGV must visit the node again once more. 244 Finally Eq. (10) and Eq. (11) ensures that the variables of the optimization 245 problem have a binary value. 246

One of the main reason of the complexity of the problem in Eq. (1)-(11) is 247 that it includes the parameter K whose value is unknown a priori. K actually 248 represents the number of displacements the UGV needs to perform before com-249 pleting the required tasks. This issue could be solved by assigning a very large 250 value to K and allowing the UGV not to move to a new node to match the K251 value. Unfortunately, the number of binary variables in the problem increases 252 with K, therefore the complexity of the problem easily increases. On the other 253 hand, if the value of K is underestimated the problem might become infeasible or 254 the model may lead to a feasible solution that is not the optimal one. Another 255 alternative to tackle such difficulty could be the column generation approach 256 (CGA) whose implementation requires to formulate a restrictive master prob-257 lem (RMP) associated with the problem in Eq. (1)-(11) and progressively add 258 new columns and thus new variables to the RMP until the minimum number of 259 variables necessary to get the optimal solution is reached. More details about 260 the CGA can be found in [46, 47]. The main drawback of this approach is the 261

difficulty of getting a RMP associated with a given integer linear programming
problem that guarantees both the effectiveness and the efficiency of the CGA.

The minimum value for K needed to obtain the optimal solution is strongly related to the topology of the underlying graph of the problem, nevertheless the following proposition holds:

Proposition 4.1. Let L denote the number of edges that constitutes the shortest path between a pair of nodes in \mathcal{V} having the largest number of edges and ABthe cardinality of $\mathcal{A} \cap \mathcal{B}$. If $K \ge L(3A - 2AB + 1)$ then the solution provided by the model in Eq. (1)-(11) is the optimal one.

Proof First, consider the case in which the graph is complete and thus L = 1. Denoting by N_i the total number of times the UGV visits and leaves a node $i \in \mathcal{V}$ before any solution is found, it follows that

$$\sum_{i \in \mathcal{V}} N_i = 2K. \tag{12}$$

Since each node $a \in \mathcal{A} \setminus \mathcal{B}$ has to be visited twice and that since the graph is complete, extra visits cannot occur in a, therefore

274

277

280

$$N_a = 4. (13)$$

Furthermore, a node $b \in \mathcal{B}$ is visited to transmit the data collected at one or more nodes $a \in \mathcal{A}$, hence

$$\sum_{b \in \mathcal{B}} N_b \le 2A. \tag{14}$$

Using (13) and (14) and considering that the UGV needs to leave the depot sand return back there at the end of the tour (thus 2 more displacements are to be added to N_s):

$$2K = \sum_{i \in \mathcal{V}} N_i = \sum_{a \in \mathcal{A} \setminus \mathcal{B}} N_a + \sum_{b \in \mathcal{B}} N_b \le 4(A - AB) + 2A + 2 = 6A - 4AB + 2$$

Hence if $K \ge 3A - 2AB + 1$ any feasible solution will be explored. In the more general case in which the graph is connected but not complete, extra visits to a given node than those needed for data collection, transmission or to perform ²⁸⁸ some action are expected to occur. For instance the UGV might move from a B ²⁸⁹ node to another B node to reach an A node. By definition of L, any operation ²⁹⁰ requiring a single displacement, when the graph is complete, can be performed ²⁹¹ after at most L displacements of the UGV in case the graph is connected but ²⁹² not complete. Hence, all the feasible solutions will be examined if

$$K \ge L(3A - 2AB + 1). \tag{15}$$

294

3(

30

293

295 4.2. Dummy-Vertices-Based formulation (Model II)

In this section, we explore an alternative formulation of the problem. Following the approach of [34] we use the so-called dummy vertices to take into account that vertices can be visited more than once. Let us define the following elements:

• d the number of dummy replications of each vertex

• \mathcal{V}_d set of all vertices including the dummy vertices

•
$$\mathcal{E}_d$$
 set of all edges including the dummy edges

- $\mathcal{D}_i = \{i_1, i_2, \dots, i_d\}$ set of the *d* dummy vertices associated with the vertex
- $i \in \mathcal{V}$, e.g., s_1 and s_d are respectively the first and the last dummy vertex associated to the depot s
- $x_{ij}=1$ if the arc $(i,j) \in \mathcal{E}_d$ is in the solution and 0 otherwise

•
$$\tau_i$$
 the time at which the vertex $i \in \mathcal{V}_d$ is visited

•
$$y_{ij} = 0$$
 if $\tau_j \notin [\tau_{i_1} \tau_{i_d}] \quad \forall i \in \mathcal{A} \setminus \mathcal{B}, j \in \mathcal{B}_d$

•
$$T = d \sum_{(i,j) \in \mathcal{E}} t_{ij}$$

The problem can then be alternatively formulated as follows:

$$\min_{x,y} \sum_{(i,j)\in\mathcal{E}_d} x_{ij} t_{ij} \tag{16}$$

310 st:

$$\sum_{i \in \mathcal{V}_d, (i,j) \in \mathcal{E}_d} x_{ij} \le 1 \quad \forall j \in \mathcal{V}_d \tag{17}$$

$$\sum_{j \in \mathcal{V}_d, (i,j) \in \mathcal{E}_d} x_{ij} - \sum_{j \in \mathcal{V}_d, (i,j) \in \mathcal{E},} x_{ji} = 0 \quad \forall i \in \mathcal{V}_d, i \neq s_1, i \neq s_d$$
(18)

$$\sum_{i \in \mathcal{V}_d, (i,j) \in \mathcal{E}_d} x_{ij} \ge 1 \quad \forall j \in \mathcal{A} \cap \mathcal{B}$$
(19)

$$\sum_{j \in \mathcal{V}_d, (j, i_1) \in \mathcal{E}_d} x_{ji_1} \ge 1 \quad \forall i \in \mathcal{A} \setminus \mathcal{B}$$
(20)

$$\sum_{j \in \mathcal{V}_d, (j, i_d) \in \mathcal{E}_d} x_{ji_d} \ge 1 \quad \forall i \in \mathcal{A} \setminus \mathcal{B}$$
(21)

$$\tau_{i_k} + T\left(1 - \sum_{j \in \mathcal{V}_d, (j,i) \in \mathcal{E}_d} x_{ji_k}\right) \ge \tau_{i_1} \quad \forall i \in \mathcal{A} \setminus \mathcal{B}, k = 1, 2, \dots, d$$
(22)

$$\tau_{i_k} - T\left(1 - \sum_{j \in \mathcal{V}_d, (j,i) \in \mathcal{E}_d} x_{ji_k}\right) \le \tau_{i_d} \quad \forall i \in \mathcal{A} \setminus \mathcal{B}, k = 1, 2, \dots, d$$
(23)

$$\sum_{j \in \mathcal{B}_d} y_{ij} \ge 1 \quad \forall i \in \mathcal{A} \setminus \mathcal{B}$$
(24)

$$\sum_{i \in \mathcal{V}_d, (i,j) \in \mathcal{E}_d} x_{ij} \ge y_{kj} \quad \forall j \in \mathcal{B}_d, k \in \mathcal{A} \setminus \mathcal{B}$$
(25)

$$\tau_{i_1} - T(1 - y_{ij}) \le \tau_j \le \tau_{i_d} + T(1 - y_{ij}) \quad i \in \mathcal{A} \setminus \mathcal{B}, j \in \mathcal{B}_d$$
(26)

$$\tau_i + t_{ij} x_{ij} - T(1 - x_{ij}) \le \tau_j \quad (i, j) \in \mathcal{E}_d \tag{27}$$

$$\sum_{j \in \mathcal{V}_d, (j,s_1) \in \mathcal{E}_d} x_{s_1 j} = 1 \tag{28}$$

$$\sum_{j \in \mathcal{V}_d, (j, s_d) \in \mathcal{E}_d} x_{js_d} = 1 \tag{29}$$

$$\tau_{s_1} - T\left(1 - \sum_{i \in \mathcal{V}_d, (i,j) \in \mathcal{E}_d} x_{ij}\right) \le \tau_j \le \tau_{s_d} + T\left(1 - \sum_{i \in \mathcal{V}_d, (i,j) \in \mathcal{E}_d} x_{ij}\right)$$
(30)
$$j \in \mathcal{V}_d, j \ne s_1, j \ne s_d$$

$$x_{ij} \in \{0,1\} \quad (i,j) \in \mathcal{E}_d \tag{31}$$

$$y_{ij} \in \{0,1\} \quad i \in \mathcal{A} \setminus \mathcal{B}, j \in \mathcal{B}_d \tag{32}$$

$$\tau_i \in [0, +\infty) \quad i \in \mathcal{V}_d \tag{33}$$

The model can be interpreted as follows. The minimization of the time 311 required to complete the tour is expressed in (16). The constraints (17) state 312 that any dummy node should be visited at most once. Constraints (18) establish 313 flow conservation by requiring that the number of incoming arcs of each node 314 (except the vertices s_1 and s_d) is equal to the outgoing ones. Constraints (19) 315 enforce that each node of type AB is visited at least once. The constraints (20)-316 (23) handle the two visits needed in each node of type A where data transmission 317 is not allowed. In fact $\forall i \in \mathcal{A} \setminus \mathcal{B}$ it is required that the associated dummy nodes 318 i_1 and i_d should be visited. Furthermore, the first and the last visit in i should 319 occur using respectively i_1 and i_d . Constraints (24)-(25) ensure that the data 320 collected in each node $i \in \mathcal{A} \setminus \mathcal{B}$ are transmitted from some node $j \in \mathcal{B}_d$ before the 321 last visit to node i occurs, while (26) models the relation between the variables 322 $y_{ij}, \tau_j, \tau_{i_1}, \tau_{i_d}$. Time constraints are expressed in (27). The constraints (28)-323 (29) enforce the connectivity between the starting location and the other nodes. 324 Constraints (30) require that the tour starts and ends at the depot s. Finally, 325 (31)-(33) determine the domain of the decision variables. 326

327 5. Proposed solution

To avoid an unreasonable increase in the number of decision variables, we 328 propose to employ a graph visiting algorithm coupled with a branch and bound 329 approach so that all possible solutions are explored while complexity is mini-330 mized. For the sake of convenience we remind the goal of the problem. Starting 331 from the depot s, the UGV must visit every node $a \in \mathcal{A}$, acquire some data, 332 find a location $b \in \mathcal{B}$ where the data can be sent, receive instructions, go back 333 to a, perform the required task and, at the end, return back to s spending the 334 lowest possible amount of time. 335

Let us denote by a_1, a_2, \ldots, a_A the A nodes to be visited. While the UGV is moving through the graph, the state of the UGV itself is completely defined by

$$\mathbb{S} = (n_1, n_2, \dots, n_A, u, c)$$

where u is the node the UGV is visiting, c is the total cost of the edges traversed so far and

$$n_{i} = \begin{cases} 0 & \text{if the node } a_{i} \text{ has not yet been visited} \\ 1 & \text{if the node } a_{i} \text{ has been visited} \\ 2 & \text{if the data associated with node } a_{i} \text{ has been transmitted} \\ 3 & \text{if the task required for the node } a_{i} \text{ has been performed} \end{cases}$$

Definition 5.1. The state $\mathbb{S} = (n_1, n_2, \dots, n_A, u, c)$ is said to be a feasible solution of the problem if u = s and

$$n_i = 3 \quad \forall i \in \{1, 2, \dots, A\}$$

344

The state $S = (n_1, n_2, \dots, n_A, u, c)$ is branched taking into consideration all possible displacements of the UGV from the node u to any other node connected to u by one edge and this leads to the construction of the state tree as illustrated in Example 5.1.

Example 5.1. Consider the network in Fig. 1 in which the node $a_1 \in \mathcal{A} \setminus \mathcal{B}$, $a_2b_2 \in \mathcal{A} \cap \mathcal{B}$, $s \in \mathcal{B}$ and $b_1 \in \mathcal{B}$. The network shows two nodes of interest



Figure 1: Network of the example 5.1



Figure 2: State tree of the example 5.1

 a_1 and a_2b_2 and hence the generic state is given by (n_1, n_2, u, c) in which n_1 351 and n_2 are associated with a_1 and a_2b_2 (see Eq. (5)) In Fig. 2 we represent 352 part of the tree of the states that are generated by the algorithm. The root of 353 the tree is the state (0,0,s,0) corresponding to the initial location. From node 354 s, if the UGV moves to the node a_1 , n_1 takes value 1 and the new state will 355 be $(1, 0, a_1, 2)$; if the UGV moves to node b_1 , the new state will be $(0, 0, b_1, 5)$. 356 Finally, if the UGV moves to node a_2b_2 , since communication is possible there, 357 the UGV will send the corresponding information and then it will perform the 358 required task. Hence n_3 will assume value 3 and the state will be $(0, 3, a_2b_2, 10)$. 359 Such reasoning is applied recursively to all the newly generated states. Figure 2 360 shows also the next expansion for the next level of the visit. 361



first one (branch) consists in expanding the state tree by generating new states from its leaves, as illustrated in Example 5.1, and the second one (bound) consists in reducing the state tree by means of a pruning procedure that terminates the visit on the leaves of the state tree that can not lead to the optimal solution. The decision on what leaf can or can not lead to an optimal solution, i.e. pruning, is based on the following two rules:

If the lower bound associated with any state among those to be expanded
 (the leaves) is greater than the cost of any already computed feasible
 solution, such state is closed, i.e., it is no more considered for branching.

2. If any state among those to be branched differs from a state S already
reached just for its cost, such state is closed if its cost is greater than the
one of S.

In order to derive a lower bound associated with a state $\mathbb{S} = (n_1, n_2, \dots, n_A, u, c)$, first let us consider the set

$$\mathcal{V}_{\mathbb{S}} = \{ a \in \mathcal{A} \mid n_a \neq 3 \}$$

37

At the state S, each node $a \in \mathcal{V}_{S} \setminus \mathcal{B}$ such that $n_a = 0$ is still to be visited 378 at least twice and those with $n_a = 1$ or $n_a = 2$ should still be visited at least 379 once. Thus, in (34), each node $a \in \mathcal{V}_{\mathbb{S}} \setminus \mathcal{B}$ has still to be visited at least 380 $2 - \min(n_a, 2)$ times. Furthermore, each node $b \in \mathcal{V}_{\mathbb{S}} \cap \mathcal{B}$ still has to be visited 381 at least once. The information acquired in each node $a \in \mathcal{V}_{\mathbb{S}} \setminus \mathcal{B}$ has then 382 to be transmitted by visiting some of the nodes $b \in \mathcal{B}$ and, finally, the UGV 383 has to go back to the depot s. Since we are searching for a lower bound of 384 the time needed to perform the aforementioned operations, we relax the flow 385 conservation constraints, hence we assume that: i) the UGV visits and leaves 386 each node $j \in \mathcal{V}_{\mathbb{S}}$ using the cheapest edge connected to it $(\min_{i \in \mathcal{V}} t_{ij})$; ii) the 387 transmissions occurs only once at some node $b \in \mathcal{V}_{\mathbb{S}} \cap \mathcal{B}$ if there is any, otherwise 388 it occurs at the node $b \in \mathcal{B}$ that is reachable from any node that is still to be 389 visited assuming to spend the smallest possible time $(\min_{i \in \mathcal{V}_{S}, j \in \mathcal{B}} t_{ij})$; iii) the 390 UGV will go back to the depot using the lowest amount of time needed to reach 391

it from a node $i \in \mathcal{V}_{\mathbb{S}}$ (min_{$i \in \mathcal{V}_{\mathbb{S}}$} t_{is}). Such relaxations clearly lead to a total time that is smaller than or equal to the time that is actually needed and hence any solution derived from the state \mathbb{S} is lower bounded by

$$LB_{\mathbb{S}} = c + \left(\sum_{a \in \mathcal{V}_{\mathbb{S}} \setminus \mathcal{B}} \left(2 - \min(n_a, 1)\right) \min_{i \in \mathcal{V}} t_{ia}\right) + \left(\sum_{b \in \mathcal{V}_{\mathbb{S}} \cap \mathcal{B}} \min_{i \in \mathcal{V}} t_{ib}\right) + (1 - \min(1, |\mathcal{V}_{\mathbb{S}} \cap \mathcal{B}|)) * \min_{i \in \mathcal{V}_{\mathbb{S}}, j \in \mathcal{B}} t_{ij} + \min_{i \in \mathcal{V}_{\mathbb{S}}} t_{is}.$$
(34)

It is worth noting that the lower bound proposed in (34) is not particularly complex nor computationally expensive. Therefore it can be considered a reasonably simple example of how the proposed algorithm can be improved. Nevertheless, the use of such lower bound can systematically reduce the time needed to find an optimal solution, as shown in Section 6.

Another example about how the proposed algorithm can be improved is to make available to the algorithm itself an initial feasible solution so that the lower bound can be effectively used right from the start. Such initial feasible solution is computed using a nearest neighbour algorithm that implements the steps in Algorithm 1.

Algorithm 1 Compute Initial Feasible Solution

- 1: run the Bellman Ford algorithm to find the shortest path between each pair of nodes $(i, j), j \in \mathcal{V}, i \in \mathcal{V}$ and its total cost c_{ij} ;
- $2: \ \mathcal{T} = \mathcal{A}, \ C=0;$

395

- 3: find the node $\hat{a} \in \mathcal{T}$ such that $c_{s\hat{a}} = \min_{a \in \mathcal{A}} c_{sa}$ and set $C = C + c_{s\hat{a}}$
- 4: $\mathcal{T} = \mathcal{T} \setminus \{\hat{a}\}$
- 5: if $\mathcal{T} \neq \emptyset$, find the node $b \in \mathcal{T}$ such that $c_{\hat{a}b} = \min_{i \in \mathcal{T}} c_{\hat{a}i}$; set $C = C + c_{\hat{a}b}$, let $\hat{a} = b$, go back to step 4.
- 6: find the node $i \in \mathcal{B}$ such that $c_{\hat{a}i} = \min_{j \in \mathcal{B}} c_{\hat{a}j}$; set $C = C + c_{\hat{a}i}$
- 7: C = 2C
- 8: return C

The underlying idea is the following: starting from the depot s, the UGV to chooses and visits the closest type A node (i.e., the one reachable in the lowest

possible amount of time); then, from that node, it selects the next node in the 408 set of those to be visited using the same criteria; this process is repeated until 409 all nodes of type A are visited. Then, a type B node is visited to transmit the 410 acquired data. Finally, all the nodes are visited again following back the path 411 used for data collection, i.e., the cost is doubled. Clearly, this is not the best 412 solution, but it is a value that can be immediately compared with the computed 413 lower bound also in the initial phases when a feasible solution would not be 414 available otherwise. 415

Also, another important factor to consider in order to make the algorithm faster is that while expanding the state tree, leaves can be considered in several different orders by means of a priority value. The simplest strategy to assign such a value is to use a monotonically decreasing value each time a new leaf is created. In such a way leaves are considered in creation order. Other more advances strategies are possible and they will be presented and investigated in Section 6.

When branching a leaf S, it may happen that all the states generated from Sare canceled when the two pruning rules are applied. In this case S is closed and no more considered. Note that closed states also include feasible solutions which clearly do not need to be further expanded. The expansion of the state tree and the pruning procedure are then iteratively applied until all the leaves of the state tree are closed. In order to formally present the algorithm we introduce the following definitions:

- $\bullet \mathcal{G}$ the graph associated with the problem;
- \mathcal{S}_T the state tree;
- N_U the number of nodes adjacent to node U in the graph \mathcal{G} ;
- \mathcal{L}_g the list of all new states generated when branching the leaf having highest priority;
- \mathcal{L}_p the list of all the states in \mathcal{L}_g that are not pruned after the function Prune() has been called;

- \bullet \mathcal{F} the list containing the closed states;
- {} the empty list
- Pop(L) a function that takes as input a list of states and returns the state
 S with highest priority;
- Branch(\mathbb{S}, \mathcal{G}) a function that takes as input a state $\mathbb{S} = (n_1, n_2, \dots, n_A, U, C)$ and the graph, branches the received state and returns a list of N_U states, one for each of the adjacent nodes to U (see Example 5.1);
- $Insert(\mathcal{L}_p, \mathcal{S}_T)$ a function that inserts in the state tree the list of states received as input;
- $Prune(\mathcal{S}_T, \mathcal{L}_g)$ a function that takes as input the state tree \mathcal{S}_T and the list of newly generated states \mathcal{L}_g then eventually prunes some states from \mathcal{S}_T and \mathcal{L}_g according to the aforementioned pruning rules and returns the list of remaining states from \mathcal{L}_g ;
- $GetSolution(S_T)$ a function that extracts from the state tree the branch that leads to the leaf state that is a feasible solution with the minimum cost.
- ⁴⁵³ The algorithm consists of the steps defined in Algorithm 2.

It is worth noting that the proposed algorithm solves the problem *without making any assumption* neither on the value of K nor on the number of dummy vertices associated with each node. The value of K can be simply computed at the end by just counting the number of displacements performed by the UGV.

Algorithm 2 Compute Optimal Path

 $S_{T} = \{(0, 0, \dots, S, 0)\}$ $\mathcal{F} = \{\}$ while $((\mathbb{S} = Pop(\mathcal{S}_{T} \setminus \mathcal{F})) \neq \{\})$ do $\mathcal{L}_{g} = Branch(\mathbb{S}, \mathcal{G})$ $\mathcal{L}_{p} = Prune(\mathcal{S}_{T}, \mathcal{L}_{g})$ if $(\mathcal{L}_{p} == \{\})$ then $\mathcal{F} = \mathcal{F} \cup \mathbb{S}$ else $Insert(\mathcal{L}_{p}, \mathcal{S}_{T})$ end if
end while $Sol = GetSolution(\mathcal{S}_{T})$

458 6. Numerical results and discussion

The numerical experiments conducted in this section aim at comparing the complexity of solving the problems using both the integer and mixed integer linear programming problems represented respectively by Model I and Model II and our proposed algorithm.

Two sample graphs are shown respectively in Fig. 3 and Fig. 4. A different labelling has been adopted for these graphs in order to make them more readable when superimposed on a map. Here we use symbols instead of letters:

- the squares represent nodes of type A
- the diamonds represent nodes of type B
- the stars represent nodes of type AB

While Fig. 3 is a simple example drawn up from scratch, the graph in Fig. 4 represents a more realistic case that has been obtained by mapping a vineyard of the Langhe area in Piedmont (Italy) on an undirected graph with 33 nodes and 39 edges representing the paths across the 9 rows of the vineyard.

First, in order to quantify how the effectiveness and efficiency of Model I is 473 affected by the value of K, we first find the lowest value of K that yields the 474 optimal solution in Model I by running our branch and bound algorithm. Then, 475 Model I is solved by means of the integer linear programming solver available in 476 the CPLEX software [48] using different estimations of K including the previous 477 one. We remind that the number K of displacements of the UGV required to 478 perform all the tasks in the shortest possible time is dependent on the topology 479 of the considered graph. Choosing K according to (15) guarantees to find the 480 optimal solution. However, to explore the performance provided by commercial 481 solvers such as CPLEX, we tested different K values. To better quantify the 482 different amount of time required as a function of K, we propose to evaluate 483 the following ratio: 484

$$\rho = \frac{T_K}{T_{opt}}$$

485

where T_K and T_{opt} are the times required by CPLEX to solve the problem using 486 a certain value of K and the lowest K value that yields the optimal solution, 487 respectively. Table 1 presents the results obtained for the graph in Fig. 3. 488 The experiments have been performed using IBM ILOG CPLEX Optimization 489 Studio 12.9.0.0 on a Dell PowerEdge T640 with an Intel Xeon 4114 2.2 GHz 64 490 bit deca-core processor and 32 GB of DDR4 2400 MT/s memory. If K is less 491 than 18, i.e., the lowest value that allows to compute the optimal solution, the 492 CPLEX solver either finds a cost higher than the optimal one or no solution at 493 all. When K = 18, the CPLEX solver finds the optimal solution, as expected. 494 A rather small overestimation of K, e.g., K = 22 instead of K = 18, makes 495 CPLEX about 7 times slower, and even worse for higher values. 496

To present a more comprehensive set of results, we now focus our attention on the topology in Fig. 4 which is far more complex than the one in Fig. 3. Due to our interest in the agricultural application, this topology will be taken as the basis to randomly generate many different graphs with a variable number of nodes to visit and thus different complexity. In practice, starting from the full graph, some of its type A nodes are transformed into "transit" nodes in



Figure 3: Sample graph drawn from scratch to mimic a possible field. Here the nodes represent: POIs that need to be visited (squares), places where it is possible to communicate (diamonds), or locations that are both (stars).

| | K | T_K (s) | ρ | Cost |
|--------------------------|------|-----------|---------|------|
| underestimation | 10 | 2.16 | 0.12 | _ |
| of the K value | 12 | 3.88 | 0.22 | _ |
| | 15 | 6.17 | 0.36 | 100 |
| ${\cal K}$ optimum value | 18 | 17.33 | 1.00 | 94 |
| overestimation | 20 | 40.05 | 2.31 | 94 |
| of the K value | 22 | 120.01 | 6.92 | 94 |
| | > 25 | > 1000 | > 57.70 | 94 |

Table 1: Time (T_K) taken by the CPLEX solver for the graph in Fig. 3 for different K values in Model I.

which the UGV must not perform any operation (they are neither of type A or ⁵⁰⁴ B or AB). Also, in all experiments, the time required to cover an edge (t_{ij}) is ⁵⁰⁵ directly proportional to the length of the edge itself.

More precisely, Table 2 and 3 report the time taken by CPLEX to find the optimal solution using Model I and Model II, for different combinations of number of labelled nodes N (hence the sum of A, B and AB nodes) and position of such nodes. When CPLEX exceeded the given time limit of two hours, we report a percentage in brackets (instead of the duration of the computation) that represents how much the best solution found by the model exceeds the



Figure 4: Undirected graph that represents the available paths for the UGV on a real vineyard. Nodes are marked as described in Fig. 3 and represent the type A,B,AB nodes with the addition of some "transit" nodes marked as empty circles.

optimal solution. Also, note that all the values shown in the tables have been obtained in the best possible conditions, i.e., when the value of K leading to the optimal solution is known (for Model I) and when the number of dummy vertices to use is known (for Model II).

The results in Table 2 and Table 3 show that Model II performs better than Model I almost all the times. For Model I the execution time is quite large and can exceed the given time limit also for relatively small graph instances. Both models, when interrupted after two hours of computation, are able to provide an intermediate best solution that in some cases is very close to the optimal one, but that tends to derive more and more for complex graph instances. Our proposal, instead, is always able to compute a solution in the given time limit

| Ν | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 | #10 |
|----|----------|---------|----------|----------|----------|----------|----------|----------|----------|----------|
| 11 | 14.76 | 493.69 | 0.77 | 83.87 | 0.96 | 469.49 | 3.82 | (7.30%) | (3.11%) | (1.02%) |
| 12 | 2319.38 | 2.60 | 1647.31 | 4.35 | 1647.31 | 4.35 | 3.52 | (7.30%) | (0.00%) | (2.18%) |
| 13 | 7.61 | 40.61 | 4090.54 | 823.92 | 346.69 | 7.34 | (7.47%) | (4.57%) | (0.54%) | (0.00%) |
| 14 | 4670.32 | 18.86 | 856.01 | (7.59%) | (0.61%) | (1.56%) | (8.47%) | (14.97%) | (3.39%) | (5.26%) |
| 15 | 1723.71 | 2057.74 | 4100.92 | 429.9 | (0.00%) | (0.00%) | (0.00%) | (14.33%) | (30.49%) | (5.72%) |
| 16 | 2700.44 | 3920.32 | (3.83%) | (3.38%) | (0.00%) | (7.02%) | (2.25%) | (11.34%) | (8.18%) | (6.87%) |
| 17 | (34.07%) | (3.33%) | (14.74%) | (38.41%) | (6.06%) | (35.94%) | (37.23%) | (27.65%) | (39.01%) | (53.03%) |
| 18 | (34.11%) | (8.96%) | (9.21%) | (25.69%) | (11.24%) | (28.14%) | (36.87%) | (55.70%) | (29.28%) | (18.50%) |

Table 2: Time (s) required to solve Model I with a time limit of two hours. For each number of nodes N, ten problem instances (#1 ... #10) have been randomly generated. When the time limit has been reached we report, in brackets, how much the best solution found by the model exceeds the optimal one.

| N | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 | #10 |
|----|--------|---------|---------|----------|---------|----------|----------|----------|----------|----------|
| 11 | 5.73 | 8.79 | 4.78 | 40.25 | 3.56 | 12.55 | 1.47 | 32.55 | 30.45 | 12.77 |
| 12 | 21.04 | 7.58 | 8.90 | 27.78 | 8.90 | 27.78 | 8.40 | 17.34 | 43.07 | 19.14 |
| 13 | 10.93 | 38.17 | 44.28 | 12.31 | 7.40 | 7.32 | 49.01 | 137.86 | 5.31 | 39.72 |
| 14 | 24.87 | 483.22 | 14.37 | 82.69 | 46.27 | 114.50 | 116.87 | 54.01 | 32.55 | 597.56 |
| 15 | 10.63 | 133.05 | 138.65 | 2570.43 | 234.08 | 770.36 | 97.99 | 217.67 | 532.98 | 1327.23 |
| 16 | 27.95 | 21.52 | 454.72 | 436.22 | 474.42 | 262.23 | 1757.05 | 1229.74 | 1237.98 | (0.00%) |
| 17 | 207.71 | 3261.04 | 626.26 | 1653.33 | 2485.79 | (11.45%) | (18.56%) | (7.34%) | (22.15%) | (8.51%) |
| 18 | 412.53 | 1058.85 | 2258.76 | (11.02%) | (9.62%) | (12.41%) | (10.20%) | (12.41%) | (9.47%) | (22.74%) |

Table 3: Time (s) required to solve Model II with a time limit of two hours. For each number of nodes N, the same ten instances of Table 2 have been used. When the time limit has been reached we report, in brackets, how much the best solution found by the model exceeds the optimal one.

and, when the CPLEX Model II is able to find a solution, performs 20 to 30 times faster than that.

Regarding our proposed algorithm, four cases have been considered. They are named *normal*, *reduced*, *advanced*, and *lower bound* in the following. The *normal* approach corresponds to the algorithm that runs on the graph as built from the real sample vineyard shown in Fig. 4. In such a graph we have reported also some additional nodes (marked as circles in the figure) to represent the path of the UGV in the vineyard and not only the connections between the nodes of type A, B, AB. We define such nodes as "transit" nodes because in our algorithm, when such nodes are visited, the state remains unchanged except
for the cost of the path that is updated consequently. Clearly, introducing new
nodes into a graph increases the total number of nodes which has a direct impact
on the problem's complexity.

Therefore, in order to minimize the complexity, the graph from Fig. 4 can be reduced by producing an equivalent "reduced" graph that has less nodes because all the nodes that are just transit nodes have been substituted by direct edges between the adjacent nodes, so that the same topology is maintained but the number of nodes is reduced. This is referred to as the *reduced* approach in the latter, which always runs on the "reduced" graph version.

The *advanced* approach is similar to the *reduced* approach, i.e., it runs on 542 the "reduced" graph, but the priority value described in Section 5 has been 543 changed. The priority is set to the inverse of the cost of the state (named 544 costonly in the following) thus the state tree is always expanded from the leaf 545 state with the lower cost. Following the minimum cost path, and not a path 546 given by the ordering of the nodes, the pruning function is more efficient and a 547 larger number of states are discarded by the prune rules because the algorithm 548 already found an equivalent state with a lower cost. 549

Finally, the *lower bound* approach works as the *advanced* one but at each new state it computes a lower bound, as previously described in Section 5, to prune branches faster, hence to reduce execution time.

Results are reported in Table 4 as the average on 50 different random graphs, 553 for different numbers of nodes N. As it might be expected, the complexity scales 554 exponentially with the number of nodes to be visited (i.e., transmitted and re-555 visited). However, the cost of finding the optimal solution is heavily reduced 556 if the *advanced* approach is employed. In fact, in the *reduced* approach, every 557 time a new node to be visited is added to the graph, the size of the state tree is 558 increased by a factor of 4.40, while in the *advanced* approach only by a factor 559 of 3.71. Here the S_T size corresponds to the number of nodes in the state 560 tree as defined in Section 5. This value equals the number of times the loop 561 in Algorithm 2 is run, thus it is directly proportional to the average execution 562

| | norm | nal | reduc | ced | advan | ced | lower bound | | |
|----|----------------------|--------|----------------------|--------|----------------------|--------|----------------------|--------|--|
| N | \mathcal{S}_T size | time | |
| 11 | 52529 | 3.0 | 5879 | 0.5 | 1121 | 0.1 | 852 | 0.1 | |
| 12 | 258049 | 15.2 | 34623 | 3.7 | 4275 | 0.5 | 3222 | 0.4 | |
| 13 | 979446 | 60.4 | 158299 | 16.8 | 16957 | 1.9 | 12842 | 1.7 | |
| 14 | 4018175 | 267.7 | 715881 | 76.2 | 65044 | 8.1 | 45478 | 6.3 | |
| 15 | 14663488 | 1014.4 | 2789828 | 279.7 | 245102 | 30.1 | 164629 | 21.8 | |
| 16 | 55298726 | 4135.3 | 12438650 | 1308.6 | 988588 | 128.0 | 625578 | 86.7 | |
| 17 | - | - | 57007645 | 5906.8 | 3631947 | 475.6 | 2866574 | 407.4 | |
| 18 | | - | - | - | 10687414 | 1388.2 | 7998251 | 1127.1 | |

Table 4: Complexity of the various approaches, in terms of number of nodes in the state tree $(S_T \text{ size})$ and execution time (seconds). Each row reports the average on 50 different random instances, for different numbers of nodes N.

time, also shown in Table 4. Additional time reductions are possible if lower bound techniques are used to prune states faster while exploring them, as done by the *lower bound* approach, or if an initial feasible solution is provided right at the start of the algorithm. While the lower bound consistently allows to reduce the computation time with all the instances, the initial solution does not always provide a significant gain because the priority used to visit the nodes is already based on a minimum cost criterion.

To better visualize the data, we also plot the time (on a logarithmic scale) as 570 a function of N in Figure 5. It is clear that the best approach is the *lower bound* 571 one, which can consistently outperform the others, keeping approximately the 572 same distance from the *advanced* one in terms of relative time reduction, which 573 is about 20%. We did not test graphs with more than N = 18 because in 574 that case also our algorithm would have rapidly exceeded the two hours time 575 bound we used for the experiments. In fact, the exponential trend at which the 576 computation time increases can be clearly deduced from Figure 5. 577

Table 5 shows the individual times required by the best variant (*lower bound*)



Figure 5: Time required by the different approaches of our proposed algorithm, as a function of the number of nodes N.

of our proposed algorithm for each number of nodes N, on the same ten different 579 randomly generated graphs as in Table 2 and 3. It is clear that our algorithm 580 provides lower execution time. Moreover, it does not require to estimate any 581 other parameter to be fed as input as it happens for Model I and Model II. 582 Despite the execution time of the ILP formulations can, occasionally, be some-583 how close to the one of our algorithm, Table 5 clearly shows that our algorithm 584 exhibits much more consistency across different problem instances, which is not 585 the case for the ILP formulations. For example, the CPLEX solver using Model 586 II happens to exceed the given two hours time bound starting from N = 16587 and at more than half of the times for the more complex graph instances. For 588 the less complex instances we computed that, on average (over the 50 tested 589 instances), our proposed algorithm is about 20 to 30 times faster as it can be 590 seen also by comparing the results for the few instances presented in Table 3 591 and 5. 592

⁵⁹³ Note also that our algorithm has been implemented, to speed up develop-⁵⁹⁴ ment, in the *python* scripting language, which is not particularly optimized for ⁵⁹⁵ speed. Therefore the shown time difference could potentially be improved by

| Ν | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 | #10 |
|----|---------|---------|--------|---------|---------|---------|--------|---------|---------|---------|
| 11 | 0.09 | 0.12 | 0.09 | 0.06 | 0.10 | 0.07 | 0.08 | 0.11 | 0.11 | 0.12 |
| 12 | 0.38 | 0.37 | 0.45 | 0.46 | 0.45 | 0.46 | 0.36 | 0.42 | 0.48 | 0.48 |
| 13 | 1.25 | 1.07 | 1.65 | 2.03 | 1.88 | 1.54 | 1.41 | 1.82 | 2.36 | 2.19 |
| 14 | 6.94 | 7.67 | 6.73 | 8.13 | 5.09 | 7.41 | 6.42 | 9.97 | 6.43 | 4.60 |
| 15 | 21.65 | 16.52 | 17.45 | 22.96 | 19.00 | 21.05 | 17.31 | 22.87 | 29.73 | 29.83 |
| 16 | 83.43 | 66.34 | 82.52 | 107.13 | 114.55 | 76.65 | 81.24 | 94.96 | 76.73 | 86.66 |
| 17 | 411.03 | 507.35 | 352.81 | 328.25 | 391.64 | 446.76 | 409.63 | 303.40 | 404.87 | 449.02 |
| 18 | 1057.29 | 1067.60 | 905.80 | 1374.41 | 1371.11 | 1432.13 | 872.12 | 1200.19 | 1232.31 | 1054.64 |

Table 5: Time (s) required to solve the problem using the best variant of the proposed algorithm (with lower bounds). For each number of nodes N, the same ten instances of Table 2 have been used.

⁵⁹⁶ rewriting our algorithm in native code.

⁵⁹⁷ Note that in this work we always focused on finding the guaranteed opti-⁵⁹⁸ mal solution to the problem. In fact, also with the *advanced* and *lower bound* ⁵⁹⁹ approaches, the optimality of the solution is preserved.

In order to present a more comprehensive view of the algorithm, we also investigated how fast the algorithm converges towards the optimal solution. Therefore, knowing the cost value of the optimal solution, we plotted how far is the current solution at each step of the algorithm with respect to the optimal one. We compared two different priority functions, i.e., the *costonly* one which has just been described, and the *mixed* one where the priority is defined as follows

$$\pi_{\mathbb{S}} = \frac{\frac{1}{3A} \sum_{i=1}^{A} n_i}{c} \tag{35}$$

where the state is $\mathbb{S} = (n_1, n_2, \dots, n_A, u, c)$. The basic idea is that the numerator in (35) provides an estimation of how close a state is from a feasible solution. In fact, considering that n_i can assume increasing values, from 0 (the initial state) to 3 (the final state), the higher the value of the sum, the more the visit is near to its completion. The $\frac{1}{3A}$ coefficient normalizes the numerator for the feasible



Figure 6: Analysis of the optimality of the intermediate solutions found using two different priority measures. The *costonly* measure terminates earlier with a small total state tree size, but finds a feasible solution only at the very end of the execution. The *mixed* measure makes the algorithm move faster towards a (suboptimal) feasible solution, but delays its completion.

solution to 1, since n_i ranges from 0 to 3. If the next leaf to be branched is chosen according to (35), it is expected to reach a (somewhat "good") feasible solution faster than in the case we use the *costonly* priority measure. That is because the priority value defined in (35) takes in consideration both how much of the visit has already been completed and how much it costs, thus it may tend to favour a path that completes faster the visit of the nodes, than just a path with the minimum cost.

Figure 6 shows that, for the priority function in (35), with less than 1/10 of the size of the fully grown state tree, the current solution differs from the optimal one only for less than 3.5%. This is an encouraging result to be explored in future work where more efficient heuristics could be investigated.

623 7. Conclusions

In this work we proposed an algorithm to solve the navigation problem of a UGV that aims to minimize the time needed to perform repeated acquisition,

communication and action tasks on a predetermined set of positions. Starting 626 from the graph, we proposed both a modeling approach based on integer linear 627 programming, to be tackled through commercial solvers, and a branch and 628 bound algorithm specifically designed for the problem. The algorithm has been 629 explained in details by means of detailed pseudocode and examples. Results 630 showed that the proposed algorithm is able to provide an improvement of 20 to 631 30 times over commercial linear programming solvers when compared on many 632 instances that can be solved to optimality in reasonable computational time. 633 Tests have been conducted on both synthetically generated input data and data 634 extracted from a real world case. Moreover, faster variants of the algorithm 635 have also been proposed while solution optimality is maintained. While our 636 proposals can help to speed up computation compared to the ILP approach, 637 the complexity still increases exponentially with the number of nodes due to 638 the nature of the problem. Therefore, future work will be devoted to investigate 639 efficient heuristics able to find good solutions with acceptable response times, 640 maybe even able to run on the robots themselves. As a first step in this direction, 641 in this work we also investigated how fast the proposed algorithm converges to 642 the optimal solution, providing an insight into how it is possible to extend this 643 work by designing efficient heuristics. 644

645 8. Acknowledgements

This work has been supported in part by the Politecnico di Torino Interdepartmental Center for Service Robotics (PIC4SeR) https://pic4ser.polito.it.

648 9. References

649 References

- [1] C. Zecha, J. Link, W. Claupein, Mobile sensor platforms: Categorisation
- and research applications in precision farming, Journal of Sensors and Sen sor Systems 2 (1) (2013) 51–72.

- [2] S. Bonadies, A. Lefcourt, S. A. Gadsden, A survey of unmanned ground
 vehicles with applications to agricultural and environmental sensing, in:
 Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping, Vol. 9866, International Society for Optics and Photonics, 2016, p. 98660Q.
- [3] S. Candiago, F. Remondino, M. De Giglio, M. Dubbini, M. Gattelli, Eval uating multispectral images and vegetation indices for precision farming
 applications from UAV images, Remote Sensing 7 (4) (2015) 4026–4047.
- [4] V. Lukas, J. Novák, L. Neudert, I. Svobodova, F. Rodriguez-Moreno,
 M. Edrees, J. Kren, The combination of UAV survey and landsat imagery
 for monitoring of crop vigor in precision agriculture, ISPRS-International
 Archives of the Photogrammetry, Remote Sensing and Spatial Information
 Sciences 41 (2016) 953–957.
- ⁶⁶⁶ [5] Q. Vu, M. Raković, V. Delic, A. Ronzhin, Trends in development of UAV⁶⁶⁷ UGV cooperation approaches in precision agriculture, in: A. Ronzhin,
 ⁶⁶⁸ G. Rigoll, R. Meshcheryakov (Eds.), Interactive Collaborative Robotics,
 ⁶⁶⁹ Springer International Publishing, Cham, 2018, pp. 213–221.
- [6] A. Vasudevan, D. A. Kumar, N. Bhuvaneswari, Precision farming using
 unmanned aerial and ground vehicles, in: Technological Innovations in ICT
 for Agriculture and Rural Development (TIAR), IEEE, 2016, pp. 146–150.
- ⁶⁷³ [7] A. Bechar, C. Vigneault, Agricultural robots for field operations: Concepts
 ⁶⁷⁴ and components, Biosystems Engineering 149 (2016) 94 111.
- [8] S. Nandi, S. Thota, A. Nag, S. Divyasukhananda, P. Goswami, A. Aravindakshan, R. Rodriguez, B. Mukherjee, Computing for rural empowerment:
 enabled by last-mile telecommunications, IEEE Communications Magazine
 54 (6) (2016) 102–109.
- [9] G. R. MacCartney, Jr., S. Sun, T. S. Rappaport, Y. Xing, H. Yan, J. Koka,
 R. Wang, D. Yu, Millimeter wave wireless communications: New results

- for rural connectivity, in: Proceedings of the 5th Workshop on All Things
 Cellular: Operations, Applications and Challenges, ATC '16, ACM, New
 York, NY, USA, 2016, pp. 31–36.
- [10] L. Fotio Tiotsop, A. Servetti, E. Masala, Optimally scheduling complex
 logistics operations involving acquisition, elaboration and action tasks, in:
 Proceedings of the 5th International Forum on Research and Technologies
 for Society and Industry (RTSI), Florence, Italy, 2019, to be published.
- [11] A. Castellini, A. Farinelli, G. Minuto, D. Quaglia, I. Secco, F. Tinivella,
 EXPO-AGRI: Smart automatic greenhouse control, in: IEEE Biomedical
 Circuits and Systems Conference (BioCAS), 2017, pp. 1–4.
- [12] J. Dong, J. G. Burnham, B. Boots, G. Rains, F. Dellaert, 4D crop monitoring: Spatio-temporal reconstruction for agriculture, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017,
 pp. 3878–3885.
- [13] A. Chang, J. Jung, M. M. Maeda, J. Landivar, Crop height monitoring
 with digital imagery from unmanned aerial system (UAS), Computers and
 Electronics in Agriculture 141 (2017) 232–237.
- [14] S. Khanal, J. Fulton, S. Shearer, An overview of current and potential
 applications of thermal remote sensing in precision agriculture, Computers
 and Electronics in Agriculture 139 (2017) 22–32.
- [15] L. Comba, P. Gay, J. Primicerio, D. R. Aimonino, Vineyard detection from
 unmanned aerial systems images, Computers and Electronics in Agriculture
 114 (2015) 78–87.
- [16] J. Primicerio, G. Caruso, L. Comba, A. Crisci, P. Gay, S. Guidoni, L. Genesio, D. Ricauda Aimonino, F. P. Vaccari, Individual plant definition and
 missing plant characterization in vineyards from high-resolution UAV imagery, European Journal of Remote Sensing 50 (1) (2017) 179–186.

- [17] A. Matese, S. F. Di Gennaro, Technology in precision viticulture: A state
 of the art review, International Journal of Wine Research 7 (2015) 69–81.
- [18] J. Das, G. Cross, C. Qu, A. Makineni, P. Tokekar, Y. Mulgaonkar, V. Kumar, Devices, systems, and methods for automated monitoring enabling
 precision agriculture, in: IEEE International Conference on Automation
 Science and Engineering (CASE), IEEE, 2015, pp. 462–469.
- [19] W. Pei, Y. Lan, L. Xiwen, Z. Zhiyan, Z. Wang, Y. Wang, Integrated sensor
 system for monitoring rice growth conditions based on unmanned ground
 vehicle system, International Journal of Agricultural and Biological Engineering 7 (2) (2014) 75.
- [20] Y. Marinakis, A. Migdalas, P. M. Pardalos, A new bilevel formulation for
 the vehicle routing problem and a solution method using a genetic algorithm, Journal of Global Optimization 38 (4) (2007) 555–580.
- [21] Z. Huang, Q. P. Zheng, E. Pasiliao, V. Boginski, T. Zhang, A cutting plane
 method for risk-constrained traveling salesman problem with random arc
 costs, Journal of Global Optimization (Sep 2018).
- [22] E. Fadda, R. Tadei, G. Perboli, L. F. Tiotsop, The multi-path traveling
 salesman problem with dependent random cost oscillations, in: Seventh
 Intl. Workshop on Freight Transportation and Logistics (ODYSSEUS),
 Cagliari, Italy, 2018, pp. 368–371.
- [23] G. Laporte, The vehicle routing problem: An overview of exact and approximate algorithms, European Journal of Operational Research 59 (1992)
 345–358.
- [24] B. Eksioglu, A. Volkan. Vural, A. Reisman, The vehicle routing problem:
 A taxonomic review, Computers & Industrial Engineering 57 (2009) 1472–
 1483.

- [25] M. Salavati-Khoshghalb, M. Gendreau, O. Jabali, W. Rei, A hybrid re course policy for the vehicle routing problem with stochastic demands,
 EURO Journal on Transportation and Logistics (2017) 1–30.
- [26] H. Seyyedhasani, J. S. Dvorak, Using the vehicle routing problem to reduce
 field completion times with multiple machines, Computers and Electronics
 in Agriculture 134 (2017) 142 150.
- [27] H. Seyyedhasani, J. S. Dvorak, Dynamic rerouting of a fleet of vehicles in
 agricultural operations through a dynamic multiple depot vehicle routing
 problem representation, Biosystems Engineering 171 (2018) 63 77.
- [28] D. Bochtis, C. Sørensen, The vehicle routing problem in field logistics: Part
 II, Biosystems Engineering 105 (2) (2010) 180 188.
- [29] K. Zhou, A. L. Jensen, C. Sørensen, P. Busato, D. Bothtis, Agricultural
 operations planning in fields with multiple obstacle areas, Computers and
 Electronics in Agriculture 109 (2014) 12 22.
- [30] M. Schiffer, M. Schneider, G. Walther, G. Laporte, Vehicle routing and
 location routing with intermediate stops: A review, Transportation Science
 53 (2) (2019) 319–343.
- ⁷⁵¹ [31] B. Crevier, J.-F. Cordeau, G. Laporte, The multi-depot vehicle routing
 ⁷⁵² problem with inter-depot routes, European Journal of Operational Re⁷⁵³ search 176 (2) (2007) 756–773.
- [32] B.-I. Kim, S. Kim, S. Sahoo, Waste collection vehicle routing problem with
 time windows, Computers & Operations Research 33 (12) (2006) 3624–
 3642.
- ⁷⁵⁷ [33] T. Bousonville, A. Hartmann, T. Melo, H. Kopfer, Vehicle routing and
 ⁷⁵⁸ refueling: the impact of price variations on tour length, in: Logistikman⁷⁵⁹ agement, University of Bamberg Press, 2011, pp. 83–101.

- [34] M. Schiffer, G. Walther, The electric location routing problem with time
 windows and partial recharging, European Journal of Operational Research
 260 (3) (2017) 995–1013.
- [35] P. Vansteenwegen, W. Souffriau, K. Sörensen, The travelling salesperson
 problem with hotel selection, Journal of the Operational Research Society
 63 (2012) 207–217.
- [36] M. Drexl, Synchronization in vehicle routing—a survey of VRPs with multi ple synchronization constraints, Transportation Science 46 (3) (2012) 297–
 316.
- ⁷⁶⁹ [37] A. Gasparetto, P. Boscariol, A. Lanzutti, R. Vidoni, Path planning and tra⁷⁷⁰ jectory planning algorithms: A general overview, in: Motion and operation
 ⁷⁷¹ planning of robotic systems, Springer, 2015, pp. 3–27.
- [38] L. C. Basaca-Preciado, O. Y. Sergiyenko, J. C. Rodríguez-Quinonez,
 X. Garcia, V. V. Tyrsa, M. Rivas-Lopez, D. Hernandez-Balbuena, P. Mercorelli, M. Podrygalo, A. Gurko, I. Tabakova, O. Starostenko, Optical 3D
 laser measurement system for navigation of autonomous mobile robot, Optics and Lasers in Engineering 54 (2014) 159–169.
- [39] K. Ferentinos, K. Arvanitis, N. Sigrimis, Heuristic optimization methods
 for motion planning of autonomous agricultural vehicles, Journal of Global
 Optimization 23 (2) (2002) 155–170.
- [40] O. Contente, N. Lau, F. Morgado, R. Morais, A path planning application
 for a mountain vineyard autonomous robot, in: Robot 2015: Second Iberian
 Robotics Conference, Springer, 2016, pp. 347–358.
- [41] H. Zhang, W. Tong, Y. Xu, G. Lin, The Steiner traveling salesman problem
- with online advanced edge blockages, Computers & Operations Research
 70 (2016) 26 38.
- [42] F. Nex, F. Remondino, UAV for 3D mapping applications: a review, Applied geomatics 6 (1) (2014) 1–15.

- [43] J. Torres-Sánchez, J. M. Peña, A. I. de Castro, F. López-Granados, Multitemporal mapping of the vegetation fraction in early-season wheat fields
 using images from UAV, Computers and Electronics in Agriculture 103
 (2014) 104–113.
- [44] G. Sona, D. Passoni, L. Pinto, D. Pagliari, D. Masseroni, B. Ortuani,
 A. Facchi, UAV multispectral survey to map soil and crop for precision
 farming applications, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 41 (2016) 1023–1029.
- [45] J. Primicerio, G. Caruso, L. Comba, A. Crisci, P. Gay, S. Guidoni, L. Genesio, D. Ricauda Aimonino, F. P. Vaccari, Individual plant definition and
 missing plant characterization in vineyards from high-resolution UAV imagery, European Journal of Remote Sensing 50 (1) (2017) 179–186.
- ⁸⁰⁰ [46] W. E. Wilhelm, A technical review of column generation in integer programming, Optimization and Engineering 2 (2) (2001) 159–200.
- [47] Y. Zhao, T. Larsson, E. Rönnberg, P. M. Pardalos, The fixed charge transportation problem: a strong formulation based on lagrangian decomposition and column generation, Journal of Global Optimization 72 (3) (2018)
 517–538.
- ⁸⁰⁶ [48] IBM, CPLEX.
- 807 URL https://www.ibm.com/products/ilog-cplex-optimization-studio