Bi-Objective Routing for Robotic Irrigation and Sampling in Vineyards

Thomas C. Thayer Stavros Vougioukas Ken Goldberg Stefano Carpin

Abstract— **Motivated by the use of robots in implementing new systems for precision irrigation, in this paper we consider a routing problem where the robot is tasked with collecting two unrelated rewards at once. While operating under a budget constraint limiting the number of locations it can visit, the robot needs to decide which locations to visit, to adjust water emitters and collect soil moisture samples to improve the inference model used to estimate soil water content. This problem can be cast as an instance of multi-objective orienteering, a computationally hard problem scarcely studied in the past. Building upon the heuristic solutions we recently developed for the single objective Orienteering Problem, in this paper we develop and compare various solutions for the case involving two distinct but concurrent objective functions. The goal is to develop algorithms that are both efficient and easy to tune for a non-expert human user. Extensive simulations informed by our field experience show the effectiveness of the proposed solutions.**

I. Introduction

Soil irregularity of moisture content is a poorly controlled but critical aspect of agriculture. Irrigation is used to partially control this variability, but is typically performed on the block level, where all crops on a large parcel of land will receive the same amount of water regardless of plant specific requirements and soil variation. This results in crops that may on average be watered properly, but with irregular results for individual plants. This lack of control introduces inefficiencies in the system, resulting in plants that are over or under-watered and the associated problems.

With some types of crops, such as wine grapes, special modalities of irrigation are necessary to produce the best product. Wine grapes require deficit irrigation, a.k.a. stress irrigation, where slightly under-watering vines yields grapes of higher quality. Deficit irrigation is difficult to correctly achieve on its own, but with current irrigation installations it is nearly impossible to accomplish for every vine in a block without harmful effects. To mitigate these problems, farmers often apply excess water to their fields, trading economically desirable grapes and water resources for crop security.

One way to remove these trade offs is to perform *precision irrigation*, meaning irrigating crops on a fine grain spatial and temporal basis, ideally at the individual plant level. Exactly watering each plant according to its needs using deficit irrigation means maximizing desirable crop production and minimizing water waste. This addresses variability caused by soil and ground conditions that normally effect the moisture available to plant roots. It also means upgrading current irrigation systems with an infrastructure capable of being adjusted at a higher resolution. To properly operate, such an infrastructure also requires a control system to regulate it.

RAPID (Robot Assisted Precision Irrigation Delivery) is a novel irrigation system in development by multiple University of California campuses that aims to combine robots, remote sensing, computational engines, and new irrigation hardware to provide scalable water delivery management on a per-plant level for vineyards, with the overall objective of helping farmers grow better grapes and reduce water expenditure. Data collection is performed by remote sensing with imagery and direct sensing with soil probing. Informed by this data, machine learning models are trained to infer soil conditions and necessary irrigation adjustments. Finally, robots are sent to regulate individual irrigation emitters, releasing the optimal amount of water to individual plants.

Our previous work on this project focused on routing algorithms for single and multiple robots working in vineyards adjusting irrigation emitters [13], [14], designing customized hardware to adjust variable rate emitters [1], [3], and inference [15]. In particular, the routing problem emerges when the robot needs to move from emitter to emitter adjusting them while subject to a preassigned travel budget (battery life). This is an optimization problem related to the *(Team) Orienteering Problem* ((T)OP) and is NP-hard. Because of the large size and specific structure of the typical vineyard, special heuristics were developed to compute routes to adjust a subset of possible emitters maximizing water utility. While these algorithms work well for their intended task, it is useful for a robot to perform additional tasks while in a vineyard. One such task is soil moisture probing, which augments data on the variability of soil moisture content within the vineyard. Figure 1 shows a prototype robot used by RAPID.

The additional information acquired on the fly can be used to check the accuracy of the computational engine and the effectiveness of previous irrigation adjustments. However, introducing another factor into the problem moves it into the domain of multi-objective optimization, because the spatial locations of interesting places to sample are unrelated to the positions of emitters that need adjusting. This problem is fundamentally more difficult and less studied.

This paper explores methods to expand the algorithms from prior work to perform bi-objective optimization, whereby the robot is tasked with finding a route that tries to optimize the set of emitters adjusted as well as the set of samples collected. In both instances, the values associated

T.C. Thayer and S. Carpin are with the University of California, Merced, CA, USA. S. Vougioukas is with the University of California, Davis, CA, USA. K. Goldberg is with the University of California, Berkeley, CA, USA.

This material is based upon work that is supported by the by USDA-NIFA under award number 2017-67021-25925 (NSF National Robotics Initiative). Thomas Thayer was also partially supported by the NSF under grant DGE-1633722. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the view of the USDA and NSF.

Fig. 1: The RAPID robot prototype, featuring a linear actuator mounted soil moisture probe and a data logger recording sampled values (visible on the right). This robot can collect soil moisture data while operating in the vineyard.

with emitters or samples is not constant, but rather are a spatial function of the region in which the robot operates.

The rest of this paper is organized in the following way. Section II succinctly discusses work related to this research, and section III formally details the problems being addressed. Viable procedures to solve the aforementioned problems, as well as the formerly developed heuristic method on which they are based, are outlined in section IV. Evaluations of the solution procedures, using multiple problem instances formulated from physical systems, are given in section V and final conclusions are presented in section VI.

II. Related Work

The problems investigated in this work are related to the OP. First discussed in [16], it was proven as NP-hard in [4], and APX-hard in [2]. The goal is to compute a path over a graph that maximizes reward gained for visiting each vertex while limited by a travel budget for costs associated with each edge. Rewards can only be collected once per vertex but costs are accumulated each time an edge is crossed. Solution approaches commonly involve heuristics due to the innate difficulty of the problem, but exact methods are also used when problem sizes are small enough (i.e. contain less than 1,000 vertexes). For large problem instances that contain tens of thousands of vertexes, domain specific heuristics are necessary to generate a useful solution, as evidenced by our prior work [14]. An abundance of variations exist for the OP, and the reader should consult [5] for a contemporary survey discussing some of the more common variants and recent solution procedures.

This paper deals with the *Bi-Objective Orienteering Problem* (BOOP), where two separate reward functions are considered. The problem is described in detail in [12], and because it is an extension of the OP it is also NP-hard. Unlike the OP, the BOOP and other multi-objective versions of the OP are sparsely studied, with most work motivated by specific applications such as tourism or work scheduling. A few solution methods have been created for it, mostly relying on multi-objective variations of prevalent orienteering heuristics. These include Pareto Ant Colony Optimization and Pareto Variable neighborhood search [12], Multi-Objective Artificial Bee Colony [7], and Greedy Randomized Adaptive Search Procedure with Path Relinking [6] for the standard BOOP. A few algorithms for extensions to the BOOP also exist, including Bi-Objective Large Neighborhood Search [8] for the BOOP with time windows, and Multi-Objective Memetic Algorithm and Multi-Objective Ant Colony System for the time-dependent BOOP [9].

III. Problem Definition

There are multiple types of the BOOP (two of which are studied in this paper), but they are all defined similarly and have the following arrangements in common. Let $G = (V, E)$ be a weighted and undirected graph. Let $c : E \to \mathbb{R}_{\geq 0}$ be the cost function for edges, and let $r_1 : V \to \mathbb{R}_{\geq 0}$ and $r_2 : V \to \mathbb{R}_{\geq 0}$ be two reward functions. On the graph a path P can be built following an ordered subset of connected vertexes $v \in V$ and edges $e \in E$. The path has an associated total cost $c(P)$ equal to the sum of costs $c(e)$ for edges in the path, and two total rewards, equal to the independent sum of rewards $r_1(v)$ and $r_2(v)$ for vertexes in the path. It is important to note that every time an edge is traversed in the path, the cost of that edge is added to the total cost, but the rewards for visiting a vertex can only be collected once, regardless of how many times the vertex is visited in the path. Additionally, let T_{max} be a travel budget.

The Dual Maximization BOOP aims at determining a path P of cost $c(P) \leq T_{max}$ that maximizes both total rewards. Because it is often the case that the two reward functions are not positively correlated with one another, it is likely that maximizing both total rewards together is impossible. As common in multi-objective optimization, different approaches can be followed to tackle this problem, including combining r_1 and r_2 into a single objective function (e.g. linear combination), studying Pareto dominant solutions [10], or characterizing situational relationships between the two reward functions that qualify reward gathering decisions.

Because the rewards we consider in our application are inherently heterogeneous (water variations and information gain), and considering in practical scenarios farmers may have individual preferences on what they value more, it is useful to study alternative approaches. The Objective Constraint BOOP aims to form a path P within the given budget satisfying a lower bound r_{min} for one of the rewards while maximizing the other. This makes the lack of a formally defined relationship between the two reward functions easier to handle, since the problem does not involve maximizing two independent or potentially anti-correlated variables.

For the special case of routing within vineyards, a class of graphs called *Irrigation Graphs* (IG) was introduced and formally defined in [14]. IGs, designated as $IG(m, n)$ where m and n are the respective number of rows and columns,

are planar graphs with degree at most 3 that delineate the movement restrictions placed on robots maneuvering through a vineyard. Each vertex symbolizes the position of an irrigation emitter (necessarily adjacent to a grapevine), and each edge illustrates viable movements between emitters. From an application perspective, desirable locations for collecting samples for soil moisture content are also co-located with vines. Therefore, vertices in the graph represent both the location of water emitters and possible points of interest for the samples. Figure 2 shows the specific arrangement of vertexes and edges for this class of graphs, where rows are only connected at their ends and movement between rows is prohibited from within.

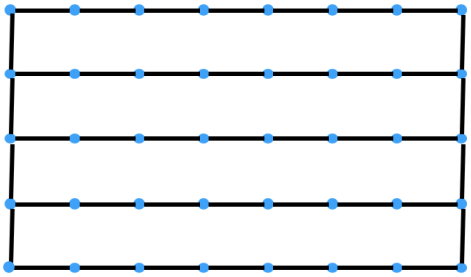

Fig. 2: An example of an IG, where each blue dot is a vertex and each black line is an edge.

Orienteering on an IG is a special case the OP, described in [14], as is team orienteering [13]. In this paper, we consider both types of the BOOP described above in the special case of IG, each defined as follows.

Irrigation Graph Dual Maximization Bi-Objective Orienteering Problem (IGDMBOOP): Given a graph $G(V, E) = IG(m, n)$ with a cost function c and two reward functions r_1, r_2, vertexes within the graph $v_1, v_n; \in V$, and a constant T_{max}, find a route over G that begins at v_1, ends at v_n, has cost no more than T_{max}, and independently maximizes the cumulative rewards $r_1(v), r_2(v);\ v \in V$, such that any reward for visiting a vertex is collected only once regardless of how many times the vertex is visited.

Irrigation Graph Objective Constraint Bi-Objective Orienteering Problem (IGOCBOOP): Given a graph $G(V, E) = IG(m, n)$ with a cost function c and two reward functions r_1, r_2, vertexes within the graph $v_1, v_n \in V$, and two constants T_{max}, r_{min}, find a route over G that begins at v_1, ends at v_n, has cost no more than T_{max}, has a cumulative reward $r_1(v) \geq r_{min}\ v \in V$, and maximizes the cumulative reward $r_2(v);\ v \in V$, such that any reward for visiting a vertex is collected only once regardless of how many times the vertex is visited.

Note that in some practical instances of these problems, based on real vineyards in regions like central California, the edge costs are safely assumed to be constant because of flat topography and homogeneous vine/emitter spacing. In [14] we proved that the single objective IG OP with constant cost is NP-hard. This can be described as a special case of either BOOP discussed here. For the IGDMBOOP, it is a case where one of the rewards r_1 or r_2 is equal to zero for all vertexes. For the IGOCBOOP, it is a case where $r_1 = 0$ for all vertexes and $r_{min} = 0$. Therefore, both the IGDMBOOP and the IGOCBOOP are NP-hard as well.

IV. Proposed Solution Methods

Because we attempt to address the BOOP in the domain of agriculture, one of the goals for our proposed solution methods is to create algorithms that have minimal input variables, so that non-expert human users can use them effectively. Therefore, our methods have been limited, beyond the specific problem inputs, to only a single tuning parameter that has an intuitive function.

A. Greedy Partial-Row Heuristic

The proposed solution methods for solving the BOOP on IGs all rely on the use of the single objective solver *Greedy Partial-Row* (GPR) introduced in [14]. Hence, we shortly outline the pseudo-code in algorithm 1.

$V_{tour} = \varnothing$
$T = 0$
$v = v_s$
for all $v(i,j) \in V$ **do**
  $R(i,j) \leftarrow \sum_{l=j}^{n} r(i,l)$
  $L(i,j) \leftarrow \sum_{l=1}^{j} r(i,l)$
  $feasible_{i,j} \leftarrow$ **true**
while there exists feasible vertexes **do**
  for all $v(i,j) \in V$ **do**
    if not feasible(i,j,T,v) **then**
      $feasible_{i,j} \leftarrow$ **false**
  for all feasible vertexes **do**
    $R'_{i,j} \leftarrow R_{i,j}/cost(v, v(i,j), v(i,n))$
    $L'_{i,j} \leftarrow L_{i,j}/cost(v, v(i,j), v(i,1))$
  for $i \leftarrow 1$ **to** m **do**
    if not feasible(i,T,v) **then**
      $feasible_i \leftarrow$ **false**
  for all feasible rows **do**
    $R'_{i,1} \leftarrow R_{i,1}/cost(v, v(i,1))$
    $L'_{i,n} \leftarrow R_{i,n}/cost(v, v(i,n))$
  if $v_j = n$ **then**
    $next \leftarrow \arg\max R'_{i,1}, R'_{i,j}$
  else
    $next \leftarrow \arg\max L'_{i,n}, L'_{i,j}$
  add path from v to $next$ to V_{tour}
  if $next_j \neq 1$ or n **then**
    add path from $next$ to $v(i,v_j)$ to V_{tour}
  update v and T
  $feasible_{pathtonext} \leftarrow$ **false**
  update $R(i,j)$ and $L(i,j)$ for relevant i
add path from v to v_s to V_{tour}
return V_{tour}

Algorithm 1: Greedy Partial-Row Heuristic (as proposed in [14])

GPR uses the vineyard's regular block structure to its advantage. The robot builds its tour iteratively by greedily adding to the tour the best full-row (complete traversal from

one side of the vineyard to the other along a horizontal row) or partial-row (traversal from one side of the vineyard along a horizontal row to a point within the row, then back to the same side). The best full-row or partial-row is the one with the highest heuristic value, defined by the sum of the collected rewards along the path divided by the total cost of the path from the current position in the graph. Every possible full-row and partial-row is checked for feasibility, which is the ability to add the path to the tour and return the robot to the start position without exceeding the overall budget. Once added to the tour, the rewards for each traversed vertex are zeroed so they cannot be collected again and their feasibility is marked as false to prevent unnecessary additions to the tour.

With a complexity of $\mathcal{O}(m^2n)$, the algorithm can efficiently compute solutions for large problem instances featuring graphs with more than 50,000 vertexes. In [14] we demonstrated that it performs better than other heuristics on IGs, especially general case heuristics. This makes it a good candidate for further development to solve bi-objective problems.

B. Dual Maximization Methods

Methods that solve the IGDMBOOP by attempting to balance the two objectives through assigning a weight or importance to each objective are included in this category. These are meta-methods, which can be used in combination with any heuristic algorithm that works on the OP for IGs, but in practice only the GPR algorithm is used.

1) Heuristic/Heuristic: The *Heuristic/Heuristic* method takes as input two reward maps, one for irrigation rewards, and one for sampling rewards. A fraction α of the overall budget is allocated to the sampling reward collection and the rest is assigned to the irrigation reward collection. In this way, budget allocation is the mechanism that determines which reward value is more important to collect. Then, the GPR algorithm is run in succession for each reward map and the tours are combined sequentially. Algorithm 2 shows this procedure.

1: V_{Stour} = GPR for sampling with allocated budget
2: Remove end of V_{Stour}
3: **for all** $v_{i,j} \in VStour$ **do**
4: $\quad r_I(i,j) = 0$
5: V_{Itour} = GPR for irrigation with remaining budget
6: $V_{tour} = V_{Stour} \cup V_{Itour}$
7: **return** V_{tour}

Algorithm 2: Heuristic/Heuristic

After running GPR for sampling reward collection with the allocated budget, the end of the produced tour V_{Stour}, which goes from the last chosen full-row or partial-row to the ending vertex, is removed from the tour. Then, the irrigation rewards for all visited vertexes are set to zero to prevent them from being collected twice, and GPR is run for irrigation reward collection with the remaining budget. Here, the starting vertex is the last vertex in V_{Stour} and the ending vertex is the original ending vertex. Finally, the two tours are combined sequentially and the total collected rewards for both irrigation and sampling are recalculated for the combined tour.

Since this method is comprised of two sequential executions of the GPR algorithm, its complexity is $\mathcal{O}(m^2n)$.

2) Heuristic/Knapsack: As with the method in section IV-B.1, the *Heuristic/Knapsack* method takes as input a reward map for both sampling and irrigation, and a fraction α of the total budget is allocated to collecting each type of reward. Then, GPR is run for irrigation rewards until the allocated budget is exhausted. With a tour in place, possible extensions to visit sampling locations are analyzed and added to the tour using dynamic programming by formulating the problem as a version of the 0-1 knapsack problem. Lastly, the GPR is run again for irrigation rewards to use up any remaining budget. Algorithm 3 outlines how this method works.

1: V_{itour} = GPR for irrigation with allocated budget
2: **for all** $v_{i,j} \in V_{itour}$ **do**
3: $\quad r_S(i,j) = 0$
4: **for all** $r_S(i,j) > 0$ **do**
5: $\quad$ Find minimal cost insertion to tour, add to prospect list
6: **while** true **do**
7: $\quad$ Perform DP 0-1 knapsack on prospect list
8: $\quad$ **if** Any chosen insertions overlap **then**
9: $\quad\quad$ Remove the one crossing less sampling locations
10: $\quad$ **else**
11: $\quad\quad$ Break
12: Insert prospects into V_{itour}
13: Remove end of V_{itour}
14: **for all** $v_{i,j} \in V_{itour}$ **do**
15: $\quad r_I(i,j) = 0$
16: V_{etour} = GPR for irrigation with remaining budget
17: $V_{tour} = V_{itour} \cup V_{etour}$
18: **return** V_{tour}

Algorithm 3: Heuristic/Knapsack

GPR is first run for irrigation rewards to build an initial tour. For each possible sampling location not already visited by the tour, the insertion (i.e. a detour that visits the sampling location beginning and ending at the same vertex in the initial tour) with the minimal cost is found and added to a list of prospects holding all such insertions. Then, dynamic programming finds which subset of prospects will maximize the sampling reward within the allocated sampling budget.

It is possible that the dynamic programming solver will choose multiple prospects that lie in the same row and overlap in their paths. This happens because each prospect is considered independently when solving the 0-1 knapsack problem, but in reality the detours for some prospects include visiting multiple sampling locations. When two overlapping prospects are chosen, the one that covers the least sampling locations is removed from the list of prospects and dynamic programming is run again. This continues until there are no more overlapping prospects in the solution, and then all the chosen prospects are added to the tour.

With the extended initial tour, it is possible that there is some remaining budget from the portion allocated to

collecting sampling rewards, so it should be used to collect more irrigation rewards. This is done in the same manner as algorithm 2, where the concluding leg of the tour is removed and previously collected rewards are zeroed out before running GPR and combining the results. Finally, the total collected irrigation and sampling rewards are recalculated and the algorithm terminates.

This algorithm runs with a complexity of $\mathcal{O}(2 \cdot m^2 n + 2mn + T_I \cdot mn + T_S \cdot m^2 n^2)$, where T_I and T_S are the budgets allocated to irrigation and sampling, respectively. The first term is due to running GPR twice, the second is due to zeroing rewards in lines 2-3 and 14-15, the third is from finding the minimal cost insertions to the tour (length of tour T_I times possible sampling locations), and he last term is due to the repeated use of the dynamic programming knapsack approach. Considering that T_S could be the entire budget T_{max}, the complexity reduces to $\mathcal{O}(T_{max} \cdot m^2 n^2)$.

3) Weighted Addition: The *Weighted Addition* approach attempts to optimize for both irrigation and sampling rewards by simply combining them as a single value over which GPR is run. Instead of splitting the total budget to identify when to collect either type of reward separately, each type of reward is normalized and assigned a weight, then the respective values are added together at each vertex in the graph.

$$r(v) = \alpha \cdot \frac{r_S(v)}{\sum_{i \in V} r_S(i)} + (1 - \alpha) \cdot \frac{r_I(v)}{\sum_{i \in V} r_I(i)} \quad \forall v \in V$$

Because there is no physical meaning behind the addition of these two rewards, the weights must be carefully chosen to reflect the desired proportional importance of each reward type. Since the combined rewards $r(v)$ are directly used by GPR, the only additional computation required is the calculation of the total collected irrigation and sampling rewards for the resulting tour.

C. Objective Constraint Methods

Methods that attempt to solve the IGOCBOOP are included in this category.

1) Heuristic/Heuristic: A simple method for maximizing one objective while satisfying the constraint on the other is to first prioritize meeting the constraint with all necessary budget resources, then focus on maximization. Because the two objectives are independent from each other, and there are no negative rewards in the IGOP problem, switching between the two objectives after one is satisfied will not cause problems later on. That is, once the minimum bound is satisfied by a tour, it will never become unsatisfied again while that tour is expanded. Therefore, it is possible to run GPR to collect rewards for the first objective until the minimum bound is met, then switch to maximizing the second objective until the budget is exhausted.

This method works the same way as the one described in section IV-B.1, except the first run of GPR terminates as soon as the minimum bound is fulfilled. Algorithm 4 shows how this method would work when a minimum sampling reward must be obtained. The complexity is the same as algorithm 2, i.e. $\mathcal{O}(m^2 n)$.

1: V_{Stour} = GPR for sampling until constraint is satisfied
2: **if** Constraint can not be satisfied **then**
3: Return empty tour
4: Remove end of V_{Stour}
5: **for all** $v_{i,j} \in VStour$ **do**
6: $r_I(i,j) = 0$
7: V_{Itour} = GPR for irrigation with remaining budget
8: $V_{tour} = V_{Stour} \cup V_{Itour}$
9: **return** V_{tour}

Algorithm 4: Objective Constraint Heuristic/Heuristic

2) Bisection GPR: It is possible that the method described in section IV-C.1 will collect additional rewards for the constrained objective while trying to maximize the other objective. Therefore it may not be necessary to satisfy the constraint immediately at the beginning of execution. It could be possible to satisfy the lower bound constraint while devoting limited resources to this goal, but it is impossible to know beforehand how to split the resources for optimally achieving both objectives. It is possible, however, to combine the methods in section IV-B with a bisection search to find the best split of parameters, whether it is how much budget to devote to collecting each type of reward or how much weight to assign to them. This approach is called *Bisection GPR* and is outlined in algorithm 5.

1: $\alpha_h = 1;\ \alpha_l = 0;\ \alpha = 0.5$
2: $V_{tour} = \varnothing$
3: **while** $\alpha_h - \alpha_l > \varepsilon$ **do**
4: $V_{tour2} \leftarrow$ Run bi-objective GPR with α and $1 - \alpha$
5: Calculate total collected rewards
6: **if** $r_1(V_{tour2}) \geq r_{min}$ and $r_2(V_{tour2}) > r_2(V_{tour})$ **then**
7: $V_{tour} = V_{tour2}$
8: $\alpha_h = \alpha$
9: $\alpha = (\alpha + \alpha_l)/2$
10: **else if** $r_1 \geq r_{min}$ **then**
11: $\alpha_h = \alpha$
12: $\alpha = (\alpha + \alpha_l)/2$
13: **else**
14: $\alpha_l = \alpha$
15: $\alpha = (\alpha + \alpha_h)/2$
16: **return** return V_{tour}

Algorithm 5: Bisection GPR

Bisection GPR can be used with the *Heuristic/Heuristic*, *Heuristic/Knapsack*, or *Weighted Addition* methods of dual maximization, and either sampling or irrigation rewards can be chosen as the constrained objective r_1 with minimal constraint r_{min}. If *Heuristic/Heuristic* or *Heuristic/Knapsack* is used, α is the fraction of total budget T_{max} devoted to gathering constrained rewards r_1 and $1 - \alpha$ is the fraction devoted to maximizing the other rewards r_2. If *Weighted Addition* is used, then α and $1 - \alpha$ are the weights w_1 and w_2, depending on which of the objectives are constrained and maximized.

Algorithm 5 will eventually terminate when the change in α is too small to cause significant changes in the tour, the limit of which is defined by ε. The complexity of this algorithm is $\mathcal{O}(X \cdot \log(1/\varepsilon))$, where X is the complexity of

the bi-objective GPR method used within.

V. Experimental Results

The methods presented in section IV were tested on multiple different routing problems built using data collected from a commercially operated vineyard in central California. A range of 13 budgets (one unit equal to the time to travel between two vines, adjusting irrigation and sampling soil moisture) and various values of α and r_{min} were used on these routing problems to thoroughly examine the effectiveness of each algorithm. The vineyard block in our experiments consisted of $m = 275$ rows with $n = 214$ columns (vines per row), for a total of 58,850 vine/emitter locations. On a vineyard of this size, with vine spacing of $6ft$ and row spacing of $8ft$ (as measured at the test site), a robot traveling at $3.3ft/s$ $(1m/s)$ would take approximately 30 hours to traverse the entire vineyard in a naive row by row fashion without making stops for irrigation adjustment and sample collection. This is well beyond the capabilities of typical battery powered robots and therefore the travel budget must be considered. Using a manual probe with GPS tagging capabilities (Hydrosense HS2P from Campbell Scientific), soil moisture data was manually collected at 72 locations uniformly spread within the block on multiple days throughout the growing season. Kriging [11] interpolation was used to obtain soil moisture and variance values for every vine location in the block.

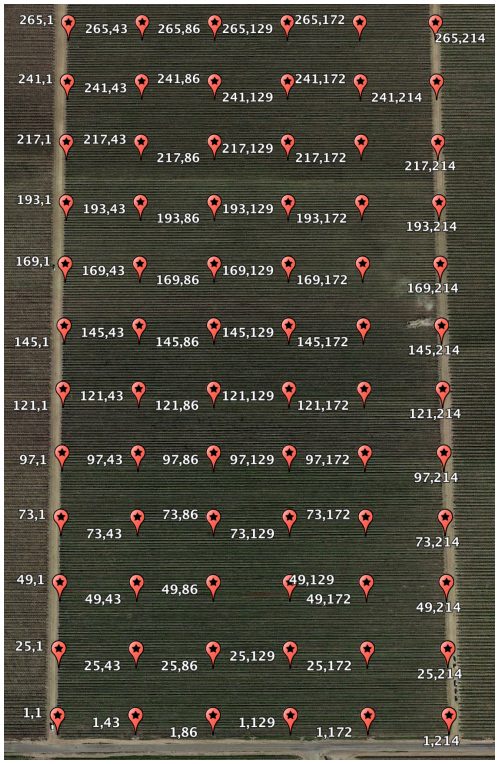

Fig. 3: Locations where soil moisture data was collected in a vineyard outside of Madera, California.

To define the rewards for the routing problems, the data was used to calculate potential rewards for visiting each vine vertex in the graph. Irrigation rewards were defined for every vine using $r_I(v) = |M - m(v)|$, where M is a constant target moisture value for the soil and $m(v)$ is the current moisture at vine vertex v. Sampling rewards $r_s(v)$ were defined as the variance values at the vines. Because soil moisture values are not expected to change much over short distances, it is unnecessary to sample at a high resolution. Therefore, only the local maxima variance values were kept and all other vertexes were given sampling rewards of 0. Noise was introduced to reduce regularity and study the algorithm under more realistic conditions. Figure 3 shows the vineyard and sampling locations, while figure 4 shows the computed irrigation rewards and sampling rewards for one day's worth of moisture data.

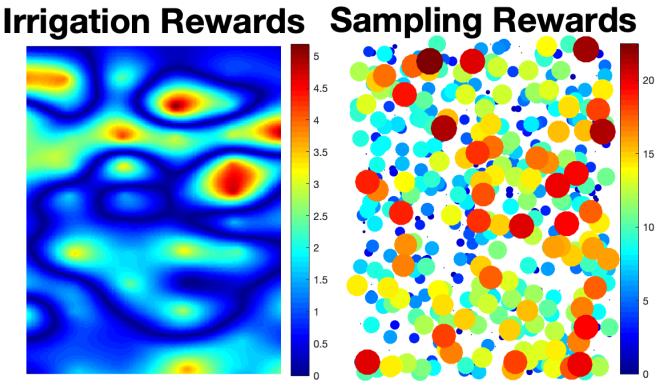

Fig. 4: A heatmap displaying the irrigation rewards for every vertex (left) and a scatter plot showing the sampling rewards for relevant vertexes (right). Note that each sampling reward is available at only a single vertex and large dots are used for an enhanced visual representation.

All of the results shown in this section are averages compiled from multiple independent tests using a different day's worth of moisture data for each test. This was done to clarify the results demonstrating which of the tested methods is better for the type of problem solved, as some variation emerges due to data fluctuation for different tests. For comparison, the original GPR algorithm with knowledge of only the irrigation rewards was run with all of the test data to see how well it performed on the bi-objective problems.

A. Dual Maximization

The methods solving the IGDMBOOP were run while varying α and the total budget T_{max}. Figures 5 and 6 show the results of this. The rewards were normalized, such that collecting 100% of rewards correlates with a value of 1 on the y-axes. The original GPR solver always received the full budget T_{max}, due to the lack of dependence on α.

Looking at the irrigation rewards in figure 5, there is a subtle trend. With smaller values for α, the irrigation rewards for all methods are nearly identical, and indeed when $\alpha = 0$ (not shown) they are the same. As α increases, the three proposed methods all diverge from GPR and collect less irrigation reward, generally with *weighted*

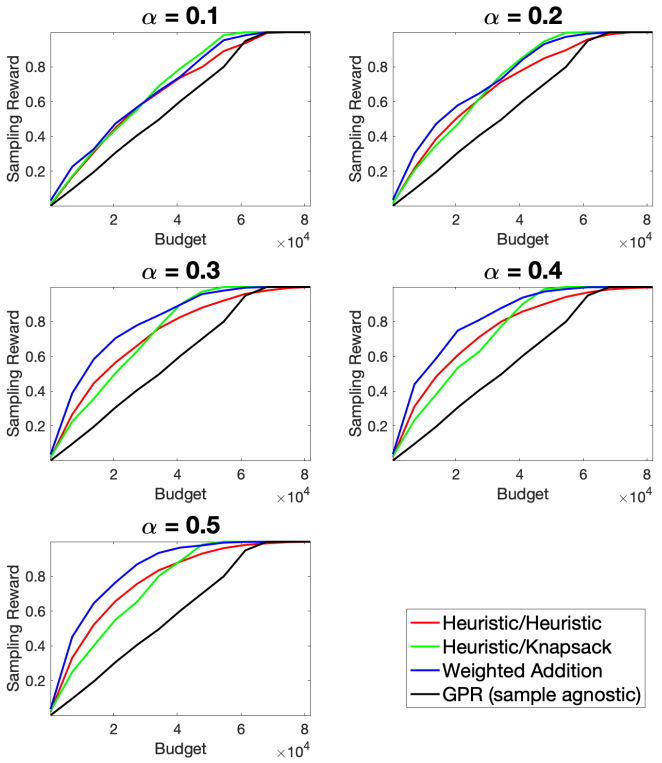

Fig. 5: Irrigation rewards for the proposed IGDMBOOP methods with different values of α. Note that GPR is consistent across each graph.

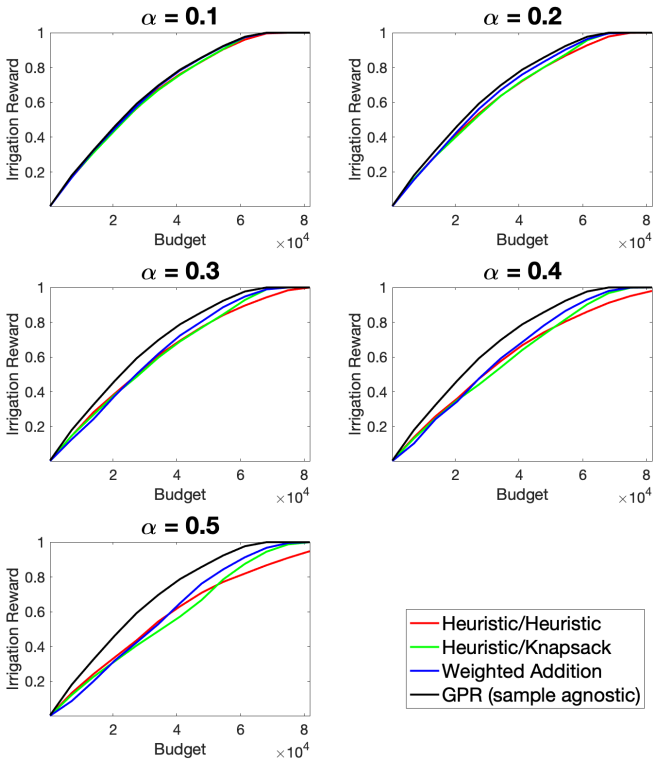

Fig. 6: Sampling rewards for the proposed IGDMBOOP methods with different values of α. Note that GPR is consistent across each graph.

addition collecting the most, and *heuristic/knapsack* trading off *heuristic/heuristic* for second.

There is a less subtle trend for sampling rewards in figure 6. At all shown values of α there is a clear separation between the three proposed methods and the original GPR, which is expected because the proposed methods all attempt to maximize sampling rewards while the GPR is unaware of these values. Larger values of α increase the gap. Interestingly, when $\alpha = 0.1$, the *heuristic/knapsack* method collects the most sampling reward, but as α increases, *weighted addition* takes the lead for most budgets. *Heuristic/heuristic* always collects more than GPR, but always less than *weighted addition*. With $\alpha = 0$ (not shown), the sampling reward is the same for all methods.

It is evident that when $\alpha \leq 0.1$, representative of when adjusting irrigation is of high importance, the *heuristic/knapsack* method is best because it collects nearly the same total irrigation reward as the original GPR but is also able to collect a substantial amount of sampling rewards more than GPR. In cases when $\alpha \geq 0.1$, where collecting soil samples is of greater importance, *weighted addition* is the clear choice.

B. Objective Constraint

The methods for solving the IGOCBOOP were run while varying the total budget T_{max}. Performance was tested for the case when sampling rewards were constrained while irrigation rewards were maximized (figure 7), and the case when irrigation rewards were constrained while sampling rewards were maximized (figure 8). In both cases, r_{min} was set to 50% of the total possible rewards. As before, the rewards were normalized, and the GPR results are shown for comparison purposes.

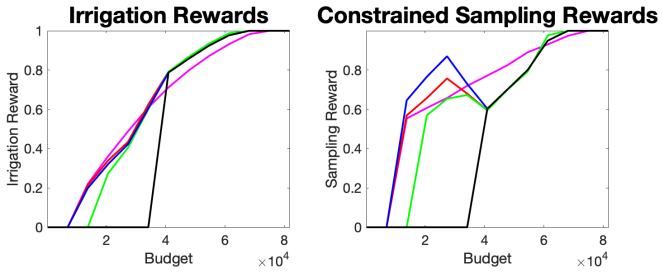

Fig. 7: The total irrigation and sampling rewards for each method, when a minimum of 50% of the sampling rewards must be collected. Magenta is *heuristic/heuristic*, red is *heuristic/heuristic bisection*, green is *heuristic/knapsack bisection*, blue is *weighted addition*, and black is GPR.

For the case when sampling rewards are constrained and irrigation rewards are maximized, the performance of the different proposed methods is very similar, while the GPR shows its deficiencies immediately. The *heuristic/knapsack bisection* method was not able to find a viable solution for the 2 smallest budgets tested, and the GPR method was unable to find solutions for half of the tested budgets, while the

other methods worked for 12 of 13 budgets. With larger budgets, *heuristic/heuristic bisection* and *heuristic/knapsack bisection* methods had the same performance at the top, but with mid-range budgets all the proposed methods had similar performance.

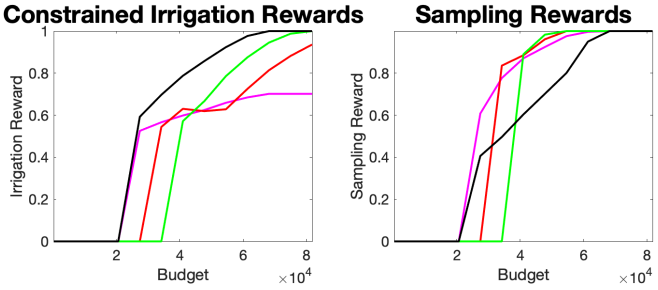

Fig. 8: The total irrigation and sampling rewards for each method, when a minimum of 50% of the irrigation rewards must be collected. Magenta is *heuristic/heuristic*, red is *heuristic/heuristic bisection*, green is *heuristic/knapsack bisection*, blue is *weighted addition*, and black is GPR. Note that lines for *weighted addition* and GPR are overlapping.

For the case when irrigation rewards are constrained and sampling rewards are maximized, there was more variability in the results. All methods lack the ability to meet the constraint with lower total budgets, but *heuristic/heuristic*, *heuristic/heuristic bisection*, *weighted addition*, and GPR were able to do it first. With regard to maximizing the sampling rewards, results are mixed with the *heuristic/heuristic*, *heuristic/heuristic bisection*, and *heuristic/knapsack* methods contending for the top spot. Surprisingly, the *weighted addition bisection* method performed poorly, showing that it is not very good at the IGOCBOOP when one of the rewards is too sparse.

VI. Conclusions

In this paper we studied vineyard routing problems for bi-objective task optimization, where a robot must adjust irrigation emitters and collect soil moisture samples simultaneously. Observing that there are various ways to formulate these problems, we describe two and present multiple solution approaches for each, based on the GPR algorithm from previous work. Then, we compared the approaches, examining how well they perform for both objectives in their respective problems.

For the IGDMBOOP, the *weighted addition* approach seemed to be the most useful for a range of inputs. For the IGOCBOOP, the *heuristic/heuristic bisection* approach was generally the better scheme. The *heuristic/knapsack* and *heuristic/knapsack bisection* methods also achieve good performance in limited situations, which suggests there might be room for improvement to make them more applicable in broader circumstances.

VII. Acknowledgments

We appreciatively acknowledge Luis Sanchez and Nick Dokoozlian from E&J Gallo Winery for allowing us access to their vineyards to collect data, and for their valuable insight provided during this project. We also thank Christine Breckenridge, Jonathan Garache, Andres Torres Garcia, and Jose Manuel Gonzalez for helping with data acquisition in the field.

References

[1] Ron Berenstein, Roy Fox, Stephen McKinley, Stefano Carpin, and Ken Goldberg. Robustly adjusting indoor drip irrigation emitters with the toyota hsr robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2236–2243, 2018.

[2] Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. *SIAM Journal on Computing*, 37(2):653–670, 2007.

[3] David V. Gealy, Stephen McKinley, Menglong Gou, Lauren Miller, Stavros Vougioukas, Joshua Viers, Stefano Carpin, and Ken Goldberg. Co-robotic device for automated tuning of emitters to enable precision irrigation. In *Proceedings of the IEEE Conference on Automation Science and Engineering*, pages 922–927, 2016.

[4] Bruce L. Golden, Larry Levy, and Rakesh Vohra. The orienteering problem. *Naval Research Logistics*, 34:307–318, 1987.

[5] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. Orienteering problem: A survey of recent variants, solution approaches, and applications. *European Journal of Operational Research*, 255(2):315–332, 2016.

[6] Rafael Mart, Vicente Campos, Mauricio G. C. Resende, and Abraham Duarte. Multiobjective grasp with path relinking. *European Journal of Operational Research*, 240(1):54–71, 2015.

[7] Rodrigo Martn-Moreno and Miguel A. Vega-Rodrguez. Multi-objective artificial bee colony algorithm applied to the bi-objective orienteering problem. *Knowledge-Based Systems*, 154:93–101, 2018.

[8] Piotr Matl, Pamela C. Nolz, Ulrike Ritzinger, Mario Ruthmair, and Fabien Tricoire. Bi-objective orienteering for personal activity scheduling. *Computers and Operational Research*, 82:69–82, 2017.

[9] Yi Mei, Flora D. Salim, and Xiaodong Li. Efficient meta-heuristics for the multi-objective time-dependent orienteering problem. *European Journal of Operational Research*, 254(2):443–457, 2016.

[10] Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

[11] M. A. Oliver and R. Webster. Kriging: A method of interpolation for geographical information systems. *International Journal of Geographical Information Systems*, 4(3):313–332, 1990.

[12] Michael Schilde, Karl F. Doerner, Richard F. Hartl, and Guenter Kiechle. Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence*, 3:179–201, 2009.

[13] Thomas C. Thayer, Stavros Vougioukas, Ken Goldberg, and Stefano Carpin. Multi-robot routing algorithms for robots operating in vineyards. In *Proceedings of the IEEE International Conference on Automation Science and Engineering*, pages 14–21, 2018.

[14] Thomas C. Thayer, Stavros Vougioukas, Ken Goldberg, and Stefano Carpin. Routing algorithms for robot assisted precision irrigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2221–2228, 2018.

[15] David Tseng, David Wang, Carolyn Chen, Lauren Miller, William Song, Joshua Viers, Stavros Vougioukas, Stefano Carpin, Juan Aparicio Ojea, and Ken Goldberg. Towards automating precision irrigation: Deep learning to infer local soil moisture conditions from synthetic aerial agricultural images. In *Proceedings of the IEEE International Conference on Automation Science and Engineering*, pages 284–291, 2018.

[16] Theodore Tsiligirides. Heuristic methods applied to orienteering. *Journal of Operational Research Society*, 35(9):797–809, 1984.