Integer Programming as a General Solution Methodology for Path-Based Optimization in Robotics: Principles, Best Practices, and Applications

Shuai D. Han   Jingjin Yu

Abstract— Integer programming (IP) has proven to be highly effective in solving many path-based optimization problems in robotics. However, the applications of IP are generally done in an ad-hoc, problem specific manner. In this work, after examined a wide range of path-based optimization problems, we describe an IP solution methodology for these problems that is both easy to apply (in two simple steps) and high-performance in terms of the computation time and the achieved optimality. We demonstrate the generality of our approach through the application to three challenging path-based optimization problems: *multi-robot path planning* (MPP), *minimum constraint removal* (MCR), and *reward collection problems* (RCPs). Associated experiments show that the approach can efficiently produce (near-)optimal solutions for problems with large state spaces, complex constraints, and complicated objective functions. In conjunction with the proposition of the IP methodology, we introduce two new and practical robotics problems: *multi-robot minimum constraint removal* (MMCR) and *multi-robot path planning* (MPP) with *partial solutions*, which can be quickly and effectively solved using our proposed IP solution pipeline.

I. INTRODUCTION

The study of robot task and motion planning problems aims at finding a path (resp., paths) for the robot (resp., robots) to optimize certain cumulative cost or reward. While some settings admit efficient search-based algorithmic solutions, e.g., via dynamic programming, such problems are frequently computationally intractable [1], [2]. In such cases, two approaches are often employed: *(i)* designing polynomial-time algorithms that compute approximately optimal solutions, and *(ii)* applying greedy search, assisted with heuristics. Both approaches have their fair share of drawbacks in practical applications: the former does not always ensure good optimality and the later often does not scale well as the problem instance becomes larger.

In this paper, we describe an *integer programming* (IP) methodology as a third general solution approach toward challenging path-based optimization problems. The key to building an IP-based solution is the construction of a *model* constituting of variables and inequalities that encodes all constraints of the target problem. For optimizing over paths, we make the important observation that it is natural to partition the model construction process into a *path-encoding* step followed by a second step that adds the *optimization constraints*. Following this methodology, we can readily solve many distinct and challenging path-based optimization problems including multi-robot path planning (MPP), minimum constraint removal (MCR), and reward collection

S. D. Han and J. Yu are with the Department of Computer Science, Rutgers, the State University of New Jersey, Piscataway, NJ, USA. E-Mails: {shuai.han, jingjin.yu} @ rutgers.edu.

problems (RCPs). As shown with extensive evaluation, the IP approaches often come with competitive performance in terms of both computation time and solution optimality. In conjunction with the proposition of the IP methodology, we introduce two new robotics problems: *multi-robot minimum constraint removal* (MMCR) and *multi-robot path planning* (MPP) with *partial solutions*. These problems are natural generalizations of MCR and MPP, respectively, that are practical but can be more challenging computationally.

Related Work. Integer programming (IP) methods are widely used to tackle combinatorial optimization challenges since their inception [3], [4], with applications to a variety of problems spanning the traveling salesperson problem (TSP) [5], network flow [6], multi-target tracking [7], etc. More recent studies have applied IP on path optimization problems in robotics including multi-robot path planning [8]–[10] and robotic manipulation [11], [12], to list a few.

This work is motivated by and builds on a long line of work that used IP, starting with the surprising initial success as IP was applied to multi-robot path planning (MPP) [10], which achieved a leap in performance in optimally solving MPP. MPP is an important problem that finds applications in a diverse array of areas including evacuation [13], formation [14], [15], localization [16], microdroplet manipulation [17], object transportation [18], search and rescue [19], and human robot interaction [20]. In the past decade, significant progress has been made on optimally solving MPP problems in discrete, graph-base environments. Algorithmic solutions for such a discrete MPP are often through reduction to other problems [10], [21], [22]. Decoupling-based heuristics are also proven to be useful [23]–[25]. Similar to the partial solution aspect examined in this paper, a recent work [26] provides a search-based solver which optimizes the number of robots that reach goals in a limited time horizon.

To demonstrate the generality and ease of application of our methodology, we also examined the minimum constraint removal (MCR) and a class of robotic reward collection problems (RCPs). MCR, which requires finding a path while removing the least number of blocking obstacles, is relevant to constraint-based task and motion planning [27], [28], object rearrangement [12], [29], and control strategy design [30]. Two search-based solvers are provided in [1] that extend to weighted obstacles [31]. Methods exist that balance between optimality, path length and computation time [32]. Recent studies on MCR reduce the gap between lower and upper bounds regarding optimality [33]. Reward collection problems (RCPs) are generally concerned with gathering rewards

without exceeding some (e.g., time or distance) budget. There is a variety of such problems including the classical traveling salesperson problem (TSP) [34] and the orienteering problem (OP) [35]. Our focus here is geared toward the more complex variations [36], [37] involving non-additive optimization objectives. Such problems model challenging information gathering tasks, e.g., precision agriculture [38], monitoring environmental attributes of the ocean [39], and infrastructure inspection [40]. These problems are generally at least NP-hard [1], [35], [41].

Contributions. This study brings two main contributions:

- Building on previous studies, we propose a general integer programming (IP) solution framework for path-based optimization problems in robotics. The two-step pipeline of the framework is easy to apply and also frequently produces highly optimal solutions.
- We formulate the multi-robot minimum constraint removal (MMCR) problem and the multi-robot path planning (MPP) with partial solutions problem, as practical generalizations of MCR and MPP, respectively. We show that the IP approach can effectively solve these new problems.

In addition to the main contributions, the study provides many additional, problem-specific heuristics that significantly enhance the performance of the baseline IP formulation; some of these heuristics are also generally applicable. Unless explicitly mentioned and referenced, the enhancements (e.g., heuristics) described in the paper are also presented here for the first time. While IP-based methods have already been used in robotics, to the best of our knowledge, this work is the first one that summarizes a general IP framework that can be readily applied to a variety of path-based optimization problems, backed by thorough simulation-based experimental evaluations.

Organization. The rest of this paper is structured as follows. In Section II, we formally define MPP, multi-robot MCR, and RCP. In Section III, we outline the general IP solution methodology and describe two tried-and-true approaches for path encoding. In Sections IV–VI, we demonstrate how the IP model may be completed for the three diverse robotics problems and introduce many best practices along the way. We conclude in Section VII. In Appendix, we provide a guidance to IP implementation.

II. PRELIMINARIES

A. Path-Based Optimization Problems

For stating path-based optimization problems, we adopt the standard graph-theoretic encoding of paths. Consider a connected undirected graph $G(V, E)$ with vertex set V and edge set E. For $v_i \in V$, let $N(v_i) = \{v_j \mid (v_i, v_j) \in E\}$ be the *neighborhood* of v_i. For a robot with initial and goal vertices $x^I, x^G \in V$, a *path* is defined as a sequence of vertices $P = (p^0, \ldots, p^T)$ satisfying: *(i)* $p^0 = x^I$; *(ii)* $p^T = x^G$; *(iii)* $\forall 1 \leq t \leq T$, $p^{t-1} = p^t$ or $(p^{t-1}, p^t) \in E$. For n robots with their initial and goal configurations given as $X^I = \{x_1^I, \ldots, x_n^I\}$ and $X^G = \{x_1^G, \ldots, x_n^G\}$, the paths are then $\mathcal{P} = \{P_1, \ldots, P_n\}$, where $P_i = (p_i^0, \ldots, p_i^T)$. These paths are not necessarily collision-free. We now outline three diverse classes of path-based optimization problems.

1) Multi-Robot Path Planning (MPP): The main task in multi-robot path planning (MPP) is routing robots to their goals while avoiding robot-robot collisions, which happen when two robots meet at a vertex or an edge. Note that the graph-theoretic formulation already considers static obstacles. For $\mathcal{P}$ to be collision-free, $\forall 1 \leq t \leq T$, $P_i, P_j \in \mathcal{P}$ must satisfy: *(i)* $p_i^t \neq p_j^t$; *(ii)* $(p_i^{t-1}, p_i^t) \neq (p_j^t, p_j^{t-1})$. The objective for MPP is to minimize the *makespan T*, which is the time for all the robots to reach the goal vertices.

In this paper, we also introduce a practical MPP generalization that allows *partial solutions*, i.e., only $k \leq n$ robots are required to reach their pre-specified goals. This formulation models diminishing reward scenarios where the payoff stops accumulating after a certain amount of targets are reached. Here, the ending vertices for the other $(n - k)$ paths can be arbitrary vertices in V. Note that $\mathcal{P}$ must still be collision-free. Being more general, the problem is also much more difficult to solve. We note that this setting is different from the case where the robots are indistinguishable, which is much simpler and admits fast polynomial time algorithms.

Problem 1 (Generalized Time-Optimal MPP). *Given $\langle G, X^I, X^G, k \rangle$, find a collision-free path set $\mathcal{P}$ that routes at least k robots to the goals and minimizes T.*
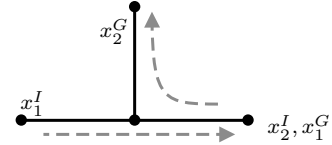

Fig. 1. A MPP example with V and E colored in black. When $k = n = 2$, a 3 step collision-free min-makespan solution requires robot 1 to stay still in the beginning as robot 2 moves to the middle vertex. When $k = 1$, the 2 step optimal solution only moves robot 2 to its goal.

An example for (partial) MPP is provided in Fig. 1.

2) Multi-Robot Minimum Constraint Removal (MMCR): Given a graph (V, E), let an *obstacle $O \subset V$* be a subset of vertices in V. Given a finite set of obstacles $\mathcal{O} = \{O_1, \ldots\}$, the multi-robot minimum constraint removal problem seeks a solution $\mathcal{P}$ and a set of obstacles to be removed $\mathcal{O}_r \subset \mathcal{O}$, such that paths in $\mathcal{P}$ do not traverse through any obstacles in $\mathcal{O} \backslash \mathcal{O}_r$. The objective is to minimize the number of obstacles to be removed, i.e., $|\mathcal{O}_r|$. More formally:

Problem 2 (MMCR). *Given $\langle G, X^I, X^G, \mathcal{O} \rangle$, find $\mathcal{P}$ and $\mathcal{O}_r$ which minimizes $|\mathcal{O}_r|$, and for all $P_i \in \mathcal{P}, O \in \mathcal{O} \backslash \mathcal{O}_r, 0 \leq t \leq T$: $p_i^t \notin O$.*

An illustration of MMCR is provided in Fig. 2.

3) Reward Collection Problem (RCP): Denote $\mathbb{R}_{\geq 0}$ as the set of non-negative real numbers. In a reward collection problem (RCP), a robot is tasked to travel on a graph G for a limited amount of time $c^* \in \mathbb{R}_{\geq 0}$, while maximizing the reward it collects[1]. Such a setup defines two functions: a cost function $C : P \rightarrow \mathbb{R}_{\geq 0}$ specifies the time already spent,

---

[1]Multi-robot RCP can be readily defined; we omit the such discussions given the already extensive multi-robot coverage and the page limit.
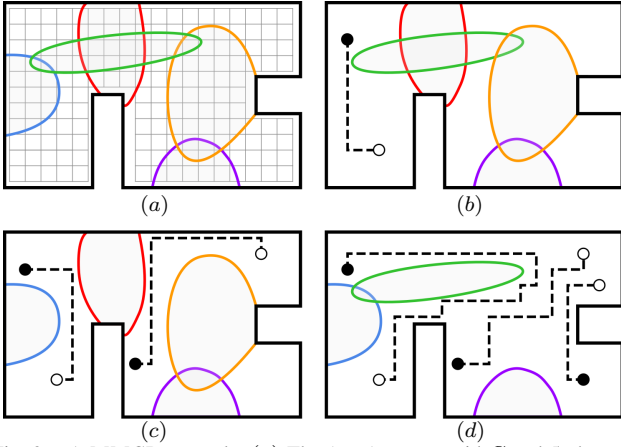
Fig. 2. A MMCR example. (a) The (gray) square grid G and 5 obstacles colored in red, blue, green, orange and purple. G is not drawn in the other sub-graphs for better visibility. (b) An optimal solution which removes the blue obstacle when $n = 1$. The dashed line shows a feasible path for the robot to move from its start (the circle) to the goal (the black dot). An alternative optimal solution is to remove the green obstacle. (c) The only optimal solution when $n = 2$. (d) An optimal solution when $n = 3$; alternative optimal solutions are also avaliable.

and a reward function $R : P \rightarrow \mathbb{R}_{\geq 0}$ tracks the reward collected along P. Obviously, *time* here may be replaced by other types of bounded resources, e.g., fuel.

Problem 3 (RCP). *Given* $\langle G, x^I, x^G, C, R, c^* \rangle$, *find* P *that maximize* $R(P)$ *under the constraint* $C(P) \leq c^*$,

We work with two variations of RCP in this paper: the quadratic correlated orienteering problem (QCOP) and the optimal tourist problem (OTP).

In a *Quadratic Correlated Orienteering Problem (QCOP)*, each vertex $v_i \in V$ is associated with a reward $r_i \in \mathbb{R}_{\geq 0}$. If $v_i \in P$, not only r_i but also partial rewards from vertices in $N(v_i)$ are collected if P does not contain these vertices. Denote the correlated weights as $\{w_{ij} = 1/|N(v_j)| \ |v_j \in N(v_i)\}$, the maximum possible reward collected from v_i is $r_i + \sum_{v_j \in N(v_i)} w_{ij} r_j$. Suppose $x_i \in \{0, 1\}$ indicates whether $v_i \in P$, the total reward collected is

$$R_{QCOP}(P) = \sum_{i=1}^{|V|} (r_i x_i + \sum_{v_j \in N(v_i)} r_j w_{ij} x_i (x_j - x_i)). \quad (1)$$

In QCOP, each edge $(v_i, v_j) \in E$ is associated with a time cost $c_{ij} \in \mathbb{R}_{\geq 0}$. The time constraint of QCOP requires that

$$C_{QCOP}(P) = \sum_{t=1}^{T} c_{p^{t-1}, p^t} \leq c^*. \quad (2)$$

In an *Optimal Tourist Problem (OTP)*, the reward collected at vertex $v_i \in V$ is described by a non-decreasing function $R_i(t_i)$, where t_i is the length of time that the robot spends at v_i. The total reward is then expressed as

$$R_{OTP}(P) = \sum_{v_i \in P} R_i(t_i). \quad (3)$$

Similar to QCOP, the time constraint of OTP requires that

$$C_{OTP}(P) = \sum_{t=1}^{T} c_{p^{t-1}, p^t} + \sum_{v_i \in P} t_i \leq c^*. \quad (4)$$

It has been shown that optimal MPP, QCOP, and OTP are all NP-hard [2], [36], [37]. MMCR is a multi-robot generalization of the single-robot MCR problem, which is

known to be NP-hard [1]. As a consequence, MMCR is also computationally intractable.

B. Integer Programming Basics

Integer programming (IP), roughly speaking, addresses a class of problems that optimize an objective function subject to integer constraints. A typical IP sub-class is integer linear programming (ILP). Given a vector $\mathbf{x}$ consists of integer variables, a general ILP model is expressed as:

$$\text{minimize } \mathbf{c}^T \mathbf{x} \quad \text{subject to } A\mathbf{x} \leq \mathbf{b}.$$

Here, $\mathbf{c}, \mathbf{b}$ are vectors and A is a matrix. In general, solution to $\mathbf{x}$ must contain only integers. Note that the formulation is compatible with equality constraints since an equality constraint can be interpreted as two inequality constraints. As will be demonstrated, due to IP's rather straightforward formulation, the reduction from path-planning problems to IP are often not hard to achieve in low polynomial time using our methodology. Apart from the unsophiscated reductions, IP is also a well-known fundamental methematical problem that has been studied extensively. Thus, an IP model can often be solved quickly and optimally using solvers like Gurobi [42], Cplex [43] and GLPK [44].

Remark. As a note on scope, our work studies IP formulations and path-based optimization problems from an algorithmic perspective. We do not handle execution details such as workspace discretization, uncertainties, and communication issues. These items are eventually to be manged by other parts in the system. For example, with proper synchronization and feedback based control, solutions generated by the ILP-based MPP solver [10] can be readily executed on multi-robot hardware platforms [45].

III. GENERAL METHODOLOGY AND PATH ENCODING

Our methodology for path-based optimization problems generally follows a straightforward two-step process. In the first step, a basic IP model is constructed that ensures only feasible paths are produced. This is the *path encoding* step. For example, in the case of a multi-robot path planning problem, the IP model must produce multiple paths that connect the desired start and goal configurations. In the second step, the optimization criteria and additional constraints, e.g., for collision avoidance, is enforced. In our experience, the path encoding step has a limited variations whereas the second step often has more variations and requires some creativity.

In this section, we provide detailed descriptions of two IP models for encoding paths: a *base-graph encoding* that works with the original edge set E, and a *time-expanded-graph encoding* that encodes paths on a new graph generated by making multiple time-stamped copies of the vertex set V. Feasible assignments to the integer variables in a given model correspond (in a one-to-one manner) to all possible paths in the original problem. The optimization step will be introduced in the next section.

A. Base-Graph Encoding

We define a *non-cyclic path* to be a path P that goes through any vertex at most once, i.e., $\forall 0 \leq i < j \leq T, p^i \neq$

p^j. Given $\langle G, x^I, x^G \rangle$, the base graph encoding introduces a binary variable x_{v_i,v_j} for each edge $(v_i, v_j) \in E$ to indicate whether P uses (v_i, v_j). The following constraints must be satisfied:

$$\sum_{v_i \in N(x^I)} x_{x^I, v_i} = \sum_{v_i \in N(x^G)} x_{v_i, x^G} = 1; \quad (5)$$

$$\sum_{v_i \in N(x^I)} x_{v_i, x^I} = \sum_{v_i \in N(x^G)} x_{x^G, v_i} = 0; \quad (6)$$

$$\sum_{v_j \in N(v_i)} x_{v_i, v_j} = \sum_{v_j \in N(v_i)} x_{v_j, v_i} \le 1, \forall v_i \in V \backslash \{x^I, x^G\}. \quad (7)$$

Here, constraint (5) and (6) make P starts from x^I and ends at x^G. Constraint (7) ensures that for each vertex, one outgoing edge is used if and only if an incoming edge is used. It prevents the path from creating multiple branches, and also forces each vertex to appear in P at most once.

A solution from this formulation could contain *subtours*, which are cycles formed by edges that are disjoint from P (see Fig. 3). As introduced in [36], these subtours can be eliminated by creating integer variables $3 \le u_i \le |V|$ for each $v_i \in V \backslash \{x^I, x^G\}$, and adding the following constraint for each pair of vertices $v_i, v_j \in V \backslash \{x^I, x^G\}$:

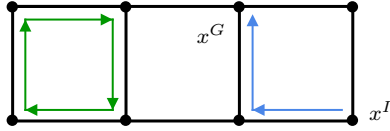$$u_i - u_j + 1 \le (|V| - 3)(1 - x_{v_i, v_j}). \quad (8)$$

Fig. 3. Illustration of a subtour in a base-graph encoding formulation. The blue lines show a feasible path from x^I to x^G. The green cycle is a possible undesirable disjoint subtour.

Proposition III.1. *Given $\langle G, x^I, x^G \rangle$, there exists a bijection between solutions to the base-graph encoding IP model and all non-cyclic paths in G from x^I to x^G.*

Proof. ($\Rightarrow$) Given a feasible solution of the base-graph encoding IP model, a non-cyclic path starts from x^I is constructed by following positive edge variables until reaching x^G. The path is guaranteed to be feasible since *(i)* by constraint (5) and (6), the path can only start at x^I and end at x^G, *(ii)* by constraint (7) and (8), the path is composed of non-repetitive vertices in V connected by edges in E.

($\Leftarrow$) A non-cyclic path $P = (p^0, \ldots, p^T)$ can be translated to a feasible solution to the IP model by assigning the corresponding edge variables to 1 and all others to 0. These values satisfies constraint (5) (6) (7) (8) because P is a sequence of vertices starts from x^I, ends at x^G, connected by edges, and has no subtours. $\square$

When $x^I = x^G$, we split $x^I(x^G)$ into two vertices v_{in}, v_{out} and connect them to all the vertices in $N(x^I)$. A path can then be formulated on the graph with vertex set $(V \backslash \{x^I\}) \cup \{v_{in}, v_{out}\}$. The resulting path is acyclic in the new graph, but interpreted as a cycled path in G and contains $x^I(x^G)$.

B. Time-Expanded-Graph Encoding

This path encoding method uses a time-expanded-graph instead of the original graph $G(V, E)$. Given a fixed time
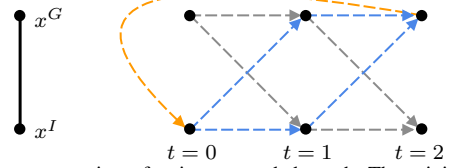
Fig. 4. The construction of a time-expanded graph. The original graph is shown on the left, with two vertices connected by an undirected edge. The time-expanded-graph with $T = 2$ is shown on the right. Directed edges (blue and grey dashed lines) are added to the adjacent vertices in the three copies of V. The feedback edge is shown as an orange dashed line. The grey edges are removed by reachability tests.

horizon $T \in \mathbb{N}$, we make $T + 1$ copies of V, namely $V^0, \ldots, V^T$. Neighboring vertices in adjacent time steps are then connected by *directed edges*: for $1 \le t \le T$, the edge set between V^{t-1} and V^t is $\{(v_i^{t-1}, v_j^t) | \forall v_i \in V, v_j \in N(v_i) \cup \{v_i\}\}$. We then add a directed *feedback edge* connecting $x^{G,T} \in V^T$ to $x^{I,0} \in V^0$. An example is provided in Fig. 4. Similar to base-graph encoding, we use a binary variable x_{v_i, v_j}^t to indicate whether (v_i^{t-1}, v_j^t) is used. The variable associated with the feedback edge is x_{x^G, x^I}^0. By denoting the outgoing (resp. incoming) edges of v_i in this time-expanded graph as $N^-(v_i)$ (resp. $N^+(v_i)$), $\forall 1 \le t \le T$, constraints (5) (6) (7) are now expressed as:

$$\sum_{v_i \in N^+(x^G)} x_{v_i, x^G}^T = x_{x^G, x^I}^0 = 1; \quad (9)$$

$$\sum_{v_j \in N^-(v_i)} x_{v_i, v_j}^t = \sum_{v_j \in N^+(v_i)} x_{v_j, v_i}^{t-1}, \ \forall v_i \in V^1 \cup \cdots \cup V^T. \quad (10)$$

The representation of a feasible path in this time-expanded graph is a sequence of directed edges that starts from $x^{I,0}$, travel through exactly one vertex in each of $V^0, \ldots, V^T$, and finally goes back to $x^{I,0}$ uses the feedback edge. Similarly, a 1-1 solution correspondence exists in this case [10].

Proposition III.2. *There is a bijection function between feasible paths in G with length less than $T + 1$, and feasible paths in the T-step time-expanded-graph.*

In a time-expanded-graph encoding, the length of the path generated is limited by the time horizon T. The path can contain any vertex for multiple times. The number of variables in the time-expanded-graph encoding can be reduced by performing *reachability tests* (see Fig. 4), which remove edges not reachable from x^I or x^G. Reachability tests do not affect completeness and optimality. Scalability can be further improved using *k-way split* [10], a divide-and-conquer heuristic that splits a problem into smaller sub-problems. The sub-problems require much shorter time to solve together than solving a single large problem.

C. Basic Extensions of the Two Encoding

Both encoding can handle the case when a robot has multiple initials or goals to choose. In base-graph encoding, this is done by adding one virtual vertex, and connecting it to all possible initials and goals. In the time-expanded-graph encoding formulation, the same objective can be achieved by adding multiple feedback edges.

Both encoding can also be extended to the multi-robot scenario. Which is achieved by simply making one copy of

all variables for each robot. The paths can then be planned disregarding mutual collisions. As we will show soon, with additional constraints, the time-expanded-graph encoding is able to generate collision-free paths.

Here we note that our IP models are solved using Gurobi Solver [42]. All experiments are executed on an Intel® Core™ i7-6900K CPU with 32GB RAM at 2133MHz.

IV. TIME-OPTIMAL MULTI-ROBOT PATH PLANNING

A high-performance time-optimal MPP IP model was proposed in [10] using the time-expanded-graph encoding. We review this baseline model and introduce two generic new heuristics for trimming the state space. We then introduce an updated IP model for MPP with partial solution.

The basic method [10] first calculates an underestimated T by routing each robot to its goal without considering robot-robot collisions. This is achieved by running n A* searches and taking the maximum path length. Then, an IP model is built and the feasibility is checked: for each robot $1 \leq r \leq n$, a set of variables $\{x^t_{v_i,v_j}\}$ satisfying constraints (9) and (10) is created and renamed as $\{x^t_{r,v_i,v_j}\}$. The method tries to find a feasible variable value assignment with the following additional constraints: for all $0 \leq t \leq T, v_i \in V^t$,

$$\sum_{r=1}^{n} \sum_{v_j \in N^+(v_i)} x^t_{r,v_j,v_i} \leq 1, \qquad (11)$$

$$\sum_{r=1}^{n} x^t_{r,v_i,v_j} + \sum_{r=1}^{n} x^t_{r,v_j,v_i} \leq 1, \forall v_j \in N^+(v_i). \qquad (12)$$

Here, constraint (11) prevents robots from simultaneously occupying the same vertex, while constraint (12) eliminates head-to-head collisions on edges in E. By Proposition III.2, an infeasible model indicates that no feasible solution exists in makespan T. Then, with T incremented by 1, the model is re-constructed and solved again. Once the model has a feasible solution, a solution to MPP with optimal makespan is then extracted.

Effective new heuristics. Intuitively, the time to solve an IP is often negatively correlated with the number of variables in the model. This suggests that the computation time may be reduced if we remove vertices in $V^0, \ldots, V^T$ which are not likely to be part of the solution path. Recall that when calculating the underestimated makespan, for each robot r, a shortest path P^*_r from x^I_i to x^G_i is obtained. When the graph is not densely occupied, we are likely to find a solution by make some minor modifications to these initial candidate paths. Building on the analysis, we propose two new heuristics, based on reachability analysis, to reduce the number of variables in the IP model.

1) Tubular Neighborhood: Fixing some parameter $h_t \in \mathbb{N}$, for robot r, the time-expanded-graph includes v_i only if v_i is within h_t distance from some vertices in P^*_r. The reachability region in this case mimics a *tube* around the candidate path. The rationale behind the heuristic is that, in general, the actual path a robot takes is not likely to significantly deviate from the reference path P^*_r.

2) Reachability Sphere: Fixing some parameter $h_s \in \mathbb{N}$, for robot r, the time-expanded-graph includes v_i only if v_i is within h_s distance from the $\lfloor t|P^*_r|/T \rfloor$-th element in P^*_r. The basic principle behind the reachability sphere heuristic is similar to that for the tubular neighborhood heuristic.

The effect of tubular neighborhood and reachability sphere heuristics are visualized on the left and right of Fig. 5, respectively. In this example, we visualize the variables when $t = 0.4T$ and assume for robot r, $|P^*_r| = 0.7T$. The dashed straight lines indicate P^*_r, with their left ends x^I_r and right ends x^G_r. For clearness, all other vertices are not shown. If no heuristic is applied, V^t contains all the vertices in the entire canvas. Through reachability tests, a vertex is removed from V^t if it is not reachable from x^I in time t, or from x^G in time $T - t$. Reachability tests remove all the vertices not in the intersection of the two arcs centered at x^I and x^G, making V^t contain only the vertices in the region with red boundaries. The tubular neighborhood heuristic with $h_t = 0.12T$ on the left sub-graph and the reachability sphere heuristic with $h_s = 0.18T$ on the right then removes all the vertices out of the shapes with blue borders. The vertices that are copied to V^t are colored in orange.
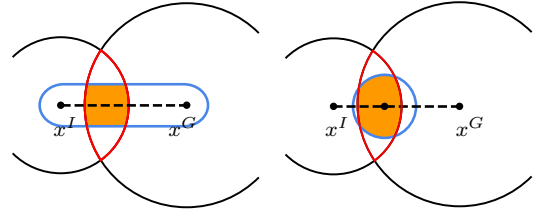

Fig. 5. Illustration of tubular neighborhood (left) and reachability sphere (right) heuristics.

We evaluate how the heuristics influence the performance using randomly generated test cases on a 24×18 grid. As it is shown in Fig. 6, the number of variables is reduced by more than 70% when $h_t \leq 2$ or $h_s \leq 2$; the computation time is reduced by at least 60% when $n \leq 60$. In the best case (the red curve), the reduction in computation time is over ten fold, which is very significant. Somewhat to our surprise, for more than 60 robots, reduced variable count in the IP model does not always translate to faster solution time. After digging in further, this seems to be related to how the Gurobi solver works: we could verify that our heuristics interfere with Gurobi's own heuristics when there are too many robots. We report that the achieved optimality is not affected by the new heuristics for all the test cases.

Partial Solutions. To accommodate the $k \leq n$ constraint in Problem 1, we prioritize using the individual path lengths in determining the time expansion parameter T. That is, we pick T to the maximum length of the shortest k robot paths out of the n paths. For the rest of the $n - k$ robots, we allow their goals in the IP model to be in a neighborhood of their respective specified goals. This is achieved through the heuristics of adding additional feedback edges (see Fig. 7) for these $(n - k)$ robots. Note that now the reachability test should not be applied to the goals of these $(n - k)$ robots.

In evaluating the general MPP solver, we use a 24×18 grid with 10% vertices removed to simulate obstacles (see
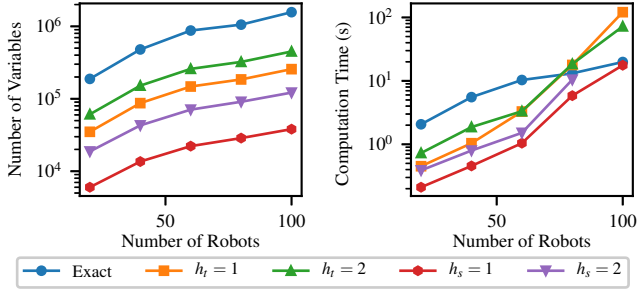
Fig. 6. Evaluation result of the proposed new heuristics for MPP. (left) The number of vertices in the IP models versus n, (right) computation time of the IP method versus n, with different heuristic parameters. The *Exact* entries indicate the original IP method.
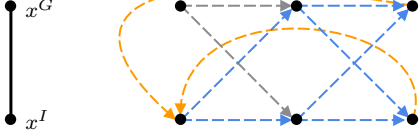

Fig. 7. Multiple feedback edges for the same example in Fig. 4 visualized using the same color and dash style.
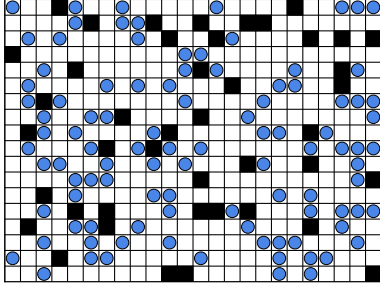

Fig. 8. An example of 24×18 grid with 10% vertices removed (visualized as black cells) and 100 randomly placed robots (visualized as blue circles).
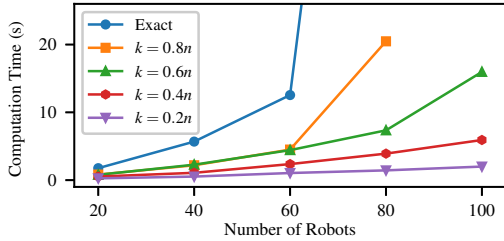

Fig. 9. Computation time of the MPP partial solver versus n, with different k values. The *Exact* entry indicates the original IP method.

Fig. 8 for an example). As shown in Fig. 9, running time is significantly reduced when we only need a partial solution.

Support for a non-fully labeled problem. We can also adapt the existing IP model to resolve the problem in which a group of robots are indistinguishable, i.e., they are assigned a set of goals without individual targets. The final configuration of these robots in a solution can be an arbitrary permutation of the goals. This is also known as the *k-colored* motion planning problem [46]. To update the model, suppose a group of m robots share m goal vertices, we create a single copy of $\{x_{v_i,v_j}^t\}$ for this group, and add m^2 feedback edges between all the pairs of initials and goals. The summation of the variables associated with these edges is set to be m.

V. MULTI-ROBOT MINIMUM CONSTRAINT REMOVAL

As implicitly stated in [1], the state space of MCR can be reduced by building a graph reflecting the coverage of differ-

ent combinations of obstacles. Here, we explicitly construct this graph $G_{\mathrm{MCR}}(V_{\mathrm{MCR}}, E_{\mathrm{MCR}})$ such that each element in V_{MCR} is a set of connected vertices that exist in the same set of obstacles. Specifically, $V_i \in V_{\mathrm{MCR}}$ satisfies the following constraints: *(i)* $V_i \subset V$; *(ii)* $\forall v_j, v_k \in V_i, O \in \mathcal{O}$, $v_j \in O$ if and only if $v_k \in O$; *(iii)* $\forall v_j, v_k \in V_i$, there exists a path P from v_j to v_k, and formed by elements in V_i. Under this definition, $\bigcup_{V_i \in V_{\mathrm{MCR}}} V_i = V$, and $\forall V_i, V_j \in V_{\mathrm{MCR}}$, $V_i \cap V_j = \emptyset$. In our implementation, V_{MCR} is built by *iteratively* selecting a random $v \in V$ and run *breadth first search* from v until all the leaf nodes collide with different sets of obstacles from v. An undirected edge (V_i, V_j) is added to the edge set E_{MCR} if there exist $v_i \in V_i$ and $v_j \in V_j$, such that $(v_i, v_j) \in E$. An example G_{MCR} is shown in Fig. 10.
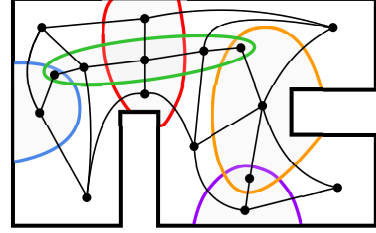

Fig. 10. G_{MCR} for the problem instance in Fig 2 illustrated using black dots and black lines. Note that each vertex can be considered as a high level abstraction of a set of vertices in a more detailed roadmap (i.e. G).

For each robot $1 \leq r \leq n$, we denote $V_r^I \in V_{\mathrm{MCR}}$ as the set contains x_r^I and $V_r^G \in V_{\mathrm{MCR}}$ as the set contains x_r^G. We assume $V_r^I \neq V_r^G$ since otherwise the set of obstacles that need to be removed for robot r is trivially $\{O \in \mathcal{O} | O \cap V_r^I \neq \emptyset\}$. With slight abuse of notation, we denote $N(V_i)$ as the neighborhood of V_i. Following base-graph encoding, for each robot $1 \leq r \leq n$, a set of variables $\{x_{V_i,V_j}\}$ satisfying constraints (5) (6) (7) are created and renamed as $\{x_{r,V_i,V_j}\}$.

In the rest of this section, unless otherwise stated, a *path* refers to a path in G_{MCR}. We introduce binary variables $\{x_{V_i} | V_i \in V_{\mathrm{MCR}}\}$ to indicate if at least one of n paths contains V_i, and $\{x_{O_i} | O_i \in \mathcal{O}\}$ to indicate whether O_i must be removed to make the paths do not collide with obstacles. The constraints for these variables are:

$$L\, x_{V_i} \geq \sum_{r=1}^{n} \sum_{V_j \in N(V_i)} (x_{r,V_i,V_j} + x_{r,V_j,V_i}); \quad (13)$$

$$L\, x_{O_i} \geq \sum_{V_i \in V_{\mathrm{MCR}}, V_i \subset O_i} x_{V_i}. \quad (14)$$

Here, L is a large constant. Constraint (13) assigns positive values to x_{V_i} only if V_i is in any path. Constraint (14) ensures that $\forall V_i \in V_{\mathrm{MCR}}$, if $x_{V_i} = 1$, then $\{x_{O_j} | O_j \in \mathcal{O}, V_i \subset O_j\}$ are all assigned positive values. The objective for this model is to minimize $\sum_{O_i \in \mathcal{O}} x_{O_i}$, i.e., the number of obstacles to be removed. Note that subtour elimination is unnecessary in MMCR since neither a subtour nor duplicate elements in a path reduce the objective value. Robot-robot collision is not considered (but can be if needed). After solving this IP, the obstacles to be removed can be extracted from the model, after which an actual robot trajectory can be found easily.

We compare the IP approach to two search-based solvers from [1]. For MCR, a search node in these solvers is a tuple $\langle v, \mathcal{O}_r \rangle$, where v is the robot's current location, and $\mathcal{O}_r$ is the

set of all the obstacles encountered from v tracing back to x^I. Different pruning methods are then applied to reduce the search space, giving rise to exact and greedy solvers. Both the exact and greedy solvers can be extended to MMCR via maintaining locations for all robots in the search state.

The comparison results are given in Fig. 11. The left figure is for MCR evaluated on a 100×100 grid and the right one for MMCR with $n = 2$ on a 10×10 grid. All test cases are generated by randomly placing arbitrary sized rectangular obstacles in the grids.
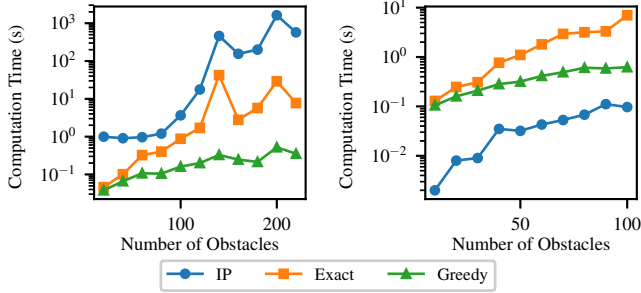

Fig. 11. Computation time of different MCR and MMCR methods versus the number of obstacles. The left subfigure shows the result for MCR, while the right subfigure shows the result for MMCR. The *Exact* and *Greedy* entries indicate the exact and greedy search-based solvers, respectively.

We observe that the exact IP method runs slower than the exact discrete search-based method for MCR but significantly outperforms both search-based methods on MMCR. We believe the reason is that with a single robot in a grid-based environment, the problem is not "complex" enough for IP-based method to utilize its structural advantages. Further evaluation shows that our IP method can solve a problem on a 50×50 grid with 100 robots and 100 obstacles in around 5 seconds, while the search-based solvers would fail to complete in such a case.

As may be observed in Fig. 11, left, the computation time does not grow monotonically as the number of obstacles increases, even though the data is already averaged over 20 instances. This is due to the NP-hardness of MCR and there could be some random instances that are particularly hard to solve. However, note that the relative performance difference between methods remains consistent.

VI. REWARD COLLECTION PROBLEMS

In this section, we examine some complex reward collection problems (RCPs) to further demonstrate the flexibility of our IP methodology. The initial work on QCOP [36] and OTP [37] used base-graph encoding. We show here these problems can also be solved using time-expanded-graph encoding to gain better computational performance.

In QCOP, the initial and goal vertices can be freely chosen from fixed sets $X^I \subset V, X^G \subset V$. To handle this, we create a virtual vertex u and add directed edges from u to all the candidacy initial vertices in V^0, and from all the possible goal vertices in $V^0, \ldots, V^T$ to u. Specifically, these directed edges are added to the time-expanded-graph: $\{(u, v^0) | v \in X^I\} \cup \{(v^t, u) | v \in X^G, 0 \le t \le T\}$. For a fixed time

horizon T, constraint (9) is now expressed as

$$\sum_{v_i \in N^-(u)} x_{u,v^i}^0 = \sum_{t=0}^{T} \sum_{v_i \in N^+(u)} x_{v_i,u}^t = 1. \quad (15)$$

Using a binary variable x_{v_i} to indicate whether $v_i \in V$ is in the path, the constraint to assign a correct value to x_{v_i} is

$$x_{v_i} \le \sum_{t=0}^{T} \sum_{v_j \in N^+(v_i^t)} x_{v_j,v_i}^t. \quad (16)$$

With binary variables indicating whether vertices and edges are used, the objective value (1) (to be maximized) is directly encoded into the IP model. The time consumption requirement (2) is added as a constraint.

The time-expanded model for OTP is similar to QCOP with one extra set of constraints $x_{u,v^i}^0 = \sum_{t=0}^{T} x_{v_i,u}^t, \forall v_i \in X^I$ to ensure $x^I = x^G$, and an additional set of non-integer non-negative variables t_i to indicate how long the robot stays at v_i. The constraint $t_i \le L x_{v_i}$ ensures that t_i is a positive value only if v_i is in the path where L is a large constant.

We use variable size grid settings similar to those from [36], [37] for evaluation. We let X^I contain 2 randomly selected vertices. In QCOP, $x^G = V$; a random reward weight r is assigned to each vertex. In OTP, we assume $R_i(t_i)$ are linear functions with random positive coefficients. We observe from the result (Fig. 12) that the time-expanded-graph encoding is always competitive and performs significantly better as the size of the problem gets larger.
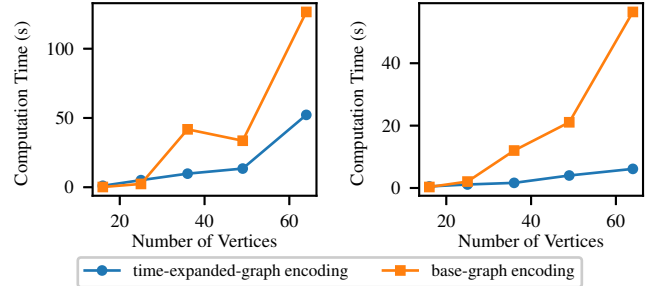

Fig. 12. Computation time of the two path encoding methods versus the number of vertices in G: (left) QCOP result, the fluctuation for base-graph encoding is due to its sensitivity to grid aspect ratio; (right) OTP result.

VII. DISCUSSION AND CONCLUSION

In this work, building on previous efforts, we propose a two-step integer programming (IP) methodology for solving path-based optimization problems. The approach is applicable to a variety of computationally hard problems in robotics involving filtering through a huge set of candidate paths. Although simple to use, harnessing the power of heavily optimized solvers, the IP method comes with performance that is often competitive or even beats conventional methods. We point out two major strengths that come with the IP solution method: *(i)* due to its ease of use, the time that is required for developing a solution can be greatly reduced, and *(ii)* the method can provide reference optimal solutions to help speed up the design of conventional algorithms. With the study, which provides principles and many best practices for IP model construction for path-based optimization, we hope to promote the adoption of the method as a first choice

when practitioners in robotics attack a new problem.

REFERENCES

[1] K. Hauser, "The minimum constraint removal problem with three robotics applications," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 5–17, 2014.

[2] J. Yu, "Intractability of optimal multi-robot path planning on planar graphs," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 33–40, 2016.

[3] G. Dantzig, *Linear programming and extensions*. Princeton university press, 2016.

[4] G. L. Nemhauser and L. A. Wolsey, "Integer programming and combinatorial optimization," *Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin*, vol. 20, pp. 8–12, 1988.

[5] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *Journal of the ACM (JACM)*, vol. 7, no. 4, pp. 326–329, 1960.

[6] T. C. Hu, "Integer programming and network flows," WISCONSIN UNIV MADISON DEPT OF COMPUTER SCIENCES, Tech. Rep., 1969.

[7] C. Morefield, "Application of 0-1 integer programming to multitarget tracking problems," *IEEE Transactions on Automatic Control*, vol. 22, no. 3, pp. 302–312, 1977.

[8] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *Control Conference (ECC), 2001 European*. IEEE, 2001, pp. 2603–2608.

[9] J. Peng and S. Akella, "Coordinating multiple robots with kinodynamic constraints along specified paths," *The International Journal of Robotics Research*, vol. 24, no. 4, pp. 295–310, 2005.

[10] J. Yu and S. M. LaValle, "Optimal multi-robot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, 2016.

[11] H. Ding, G. Reißig, D. Groß, and O. Stursberg, "Mixed-integer programming for optimal path planning of robotic manipulators," in *Automation Science and Engineering (CASE), 2011 IEEE Conference on*. IEEE, 2011, pp. 133–138.

[12] S. D. Han, N. M. Stiffler, A. Krontiris, K. E. Bekris, and J. Yu, "Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps," *The International Journal of Robotics Research*, p. 0278364918780999, 2017.

[13] S. Rodriguez and N. M. Amato, "Behavior-based evacuation planning," in *Proceedings IEEE International Conference on Robotics & Automation*, 2010, pp. 350–355.

[14] S. Poduri and G. S. Sukhatme, "Constrained coverage for mobile sensor networks," in *Proceedings IEEE International Conference on Robotics & Automation*, 2004.

[15] B. Smith, M. Egerstedt, and A. Howard, "Automatic generation of persistent formations for multi-agent networks under range constraints," *ACM/Springer Mobile Networks and Applications Journal*, vol. 14, no. 3, pp. 322–335, June 2009.

[16] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, pp. 325–344, June 2000.

[17] E. J. Griffith and S. Akella, "Coordinating multiple droplets in planar array digital microfluidic systems," *International Journal of Robotics Research*, vol. 24, no. 11, pp. 933–949, 2005.

[18] D. Rus, B. Donald, and J. Jennings, "Moving furniture with teams of autonomous robots," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots & Systems*, 1995, pp. 235–242.

[19] J. S. Jennings, G. Whelan, and W. F. Evans, "Cooperative search and rescue with a team of mobile robots," in *Proceedings IEEE International Conference on Robotics & Automation*, 1997.

[20] R. A. Knepper and D. Rus, "Pedestrian-inspired sampling-based multi-robot collision avoidance," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2012, pp. 94–100.

[21] P. Surynek, "Towards optimal cooperative path planning in hard setups through satisfiability solving," in *Proceedings 12th Pacific Rim International Conference on Artificial Intelligence*, 2012.

[22] E. Erdem, D. G. Kisa, U. Öztok, and P. Schueller, "A general formal framework for pathfinding problems with multiple agents." in *AAAI*, 2013.

[23] T. Standley and R. Korf, "Complete algorithms for cooperative pathfinding problems," in *Proceedings International Joint Conference on Artificial Intelligence*, 2011, pp. 668–673.

[24] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, 2015.

[25] E. Boyarski, A. Felner, R. Stern, G. Sharon, O. Betzalel, D. Tolpin, and E. Shimony, "Icbs: The improved conflict-based search algorithm for multi-agent pathfinding," in *Eighth Annual Symposium on Combinatorial Search*, 2015.

[26] H. Ma, G. Wagner, A. Felner, J. Li, T. K. S. Kumar, and S. Koenig, "Multi-agent path finding with deadlines," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 417–423. [Online]. Available: https://doi.org/10.24963/ijcai.2018/58

[27] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 3684–3691.

[28] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "An incremental constraint-based framework for task and motion planning," *The International Journal of Robotics Research*, p. 0278364918761570, 2018.

[29] A. Krontiris and K. E. Bekris, "Efficiently solving general rearrangement tasks: A fast extension primitive for an incremental sampling-based planner," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3924–3931.

[30] L. I. R. Castro, P. Chaudhari, J. Tumova, S. Karaman, E. Frazzoli, and D. Rus, "Incremental sampling-based algorithm for minimum-violation motion planning," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*. IEEE, 2013, pp. 3217–3224.

[31] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Ffrob: An efficient heuristic for task and motion planning," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 179–195.

[32] A. Krontiris and K. Bekris, "Computational tradeoffs of search methods for minimum constraint removal paths," in *Eighth Annual Symposium on Combinatorial Search*, 2015.

[33] E. Eiben, J. Gemmell, I. A. Kanj, and A. Youngdahl, "Improved results for minimum constraint removal." in *AAAI*, 2018.

[34] E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys, "The traveling salesman problem. 1985," *John Wiley&Sons, Essex, England*, 1985.

[35] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

[36] J. Yu, M. Schwager, and D. Rus, "Correlated orienteering problem and its application to persistent monitoring tasks," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1106–1118, 2016.

[37] J. Yu, J. Aslam, S. Karaman, and D. Rus, "Optimal tourist problem and anytime planning of trip itineraries," *arXiv preprint arXiv:1409.8536*, 2014.

[38] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic uav and ugv system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[39] K.-C. Ma, L. Liu, and G. S. Sukhatme, "An information-driven and disturbance-aware planning method for long-term ocean monitoring," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 2102–2108.

[40] C. Papachristos, K. Alexis, L. R. G. Carrillo, and A. Tzes, "Distributed infrastructure inspection path planning for aerial robotics subject to time constraints," in *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*. IEEE, 2016, pp. 406–412.

[41] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[42] Gurobi Optimization, Inc., "Gurobi optimizer reference manual," 2014. [Online]. Available: http://www.gurobi.com

[43] "Ibm ilog cplex optimization studio." [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer

[44] "Gnu linear programming kit." [Online]. Available: https://www.gnu.org/software/glpk/

[45] S. D. Han, E. J. Rodriguez, and J. Yu, "Sear: A polynomial-time multi-robot path planning algorithm with expected constant-factor optimality guarantee," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[46] K. Solovey and D. Halperin, "k-color multi-robot motion planning," in *Proceedings Workshop on Algorithmic Foundations of Robotics*, 2012.

APPENDIX:
DETAILED IP MODEL CONSTRUCTION PRACTICE

In this section, we provide implementation details of our IP formulations using some simple problem examples. To better show the IP structures, we

- do not fully apply the reachability test, and
- do not remove redundant variables and constraints.
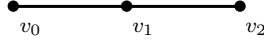
A. MPP *with Partial Solution*

Fig. 13.   A simple MPP instance with three verices $\{v_0, v_1, v_2\}$.

Consider G as shown in Fig 13, with problem parameters $n = 2, k = 1, x_1^I = v_0, x_1^G = v_1, x_2^I = v_1, x_2^G = v_2$. Using a time-expanded encoding method, the binary variables in our IP formulation for $T = 1$ are:

$$x_{1,v_0,v_0}^0, x_{1,v_1,v_0}^0, x_{2,v_0,v_1}^0, x_{2,v_1,v_1}^0, x_{2,v_2,v_1}^0,$$
$$x_{1,v_0,v_0}^1, x_{1,v_0,v_1}^1, x_{2,v_1,v_0}^1, x_{2,v_1,v_1}^1, x_{2,v_1,v_2}^1,$$

where the first 5 variables denote the usage of feedback edges, and the last 5 variables denote the usage of time-expanded edges generated from G.

The objective function is:

$$\text{maximize} \quad x_{1,v_1,v_0}^0 + x_{2,v_1,v_0}^0.$$

The constraints in the IP formulation are:
By constraint (9),

$$x_{1,v_1,v_0}^0 = x_{1,v_0,v_1}^1 = 1, x_{2,v_2,v_1}^0 = x_{2,v_1,v_2}^1 = 1.$$

By constraint (10),

$$x_{1,v_0,v_0}^1 + x_{1,v_0,v_1}^1 = x_{1,v_0,v_0}^0 + x_{1,v_1,v_0}^0,$$
$$x_{2,v_1,v_2}^1 + x_{2,v_1,v_1}^1 + x_{2,v_1,v_0}^1 = x_{2,v_2,v_1}^0 + x_{2,v_1,v_1}^0 + x_{2,v_0,v_1}^0.$$

By constraint (11),

$$x_{1,v_0,v_0}^0 + x_{1,v_1,v_0}^0 \leq 1,$$
$$x_{2,v_0,v_1}^0 + x_{2,v_1,v_1}^0 + x_{2,v_2,v_1}^0 \leq 1,$$
$$x_{1,v_0,v_0}^1 + x_{2,v_1,v_0}^1 \leq 1,$$
$$x_{1,v_0,v_1}^1 + x_{2,v_1,v_1}^1 \leq 1,$$
$$x_{2,v_1,v_2}^1 \leq 1.$$

By constraint (12):

$$x_{1,v_0,v_0}^0 \leq 1,$$
$$x_{2,v_0,v_1}^0 + x_{1,v_1,v_0}^0 \leq 1,$$
$$x_{1,v_1,v_1}^0 \leq 1,$$
$$x_{2,v_2,v_1}^0 \leq 1,$$
$$x_{1,v_0,v_0}^1 \leq 1,$$
$$x_{1,v_0,v_1}^1 + x_{2,v_1,v_0}^1 \leq 1,$$
$$x_{2,v_1,v_1}^1 \leq 1.$$
$$x_{2,v_1,v_2}^1 \leq 1.$$

Note that for a partial solution, the objective function does not need to be optimized; we can terminate the IP solution process when

$$x_{1,v_1,v_0}^0 + x_{2,v_1,v_0}^0 \geq k = 1.$$

B. MPP *Heuristics*

Continuing with the previous example, when we use tubular neighborhood with $h_t = 0$, variables x_{2,v_0,v_1}^0 and

x_{2,v_1,v_0}^1 will both be set to 0 since the shortest path for robot 2 does not include v_0. When rechability spheres with $h_s = 0$ is used, 4 more variables $x_{1,v_0,v_0}^0, x_{1,v_0,v_0}^1, x_{2,v_1,v_1}^0, x_{2,v_1,v_1}^1$ will be set to 0. For this specific problem instance, these modifications do not affect the solution feasibility and optimality.

Note that although a good reachability test will remove all the 6 variables mentioned above, on a larger problem instance the variables affected by each heuristic are generally not subsets of each other.

C. MMCR

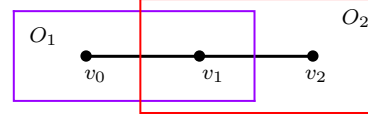Fig. 14.   A simple (M)MCR instance with three verices $\{v_0, v_1, v_2\}$ and two obstacles $O_1 = \{v_0, v_1\}, O_2 = \{v_1, v_2\}$.

As visualized in Fig 13, we add two obstacles into the previous MPP example to make it a MMCR example. With the same problem parameters $n = 2, x_1^I = v_0, x_1^G = v_1, x_2^I = v_1, x_2^G = v_2$. Using the base graph formulation, the binary variables are:

$$x_{1,v_0,v_1}, x_{1,v_1,v_0}, x_{1,v_1,v_2}, x_{1,v_2,v_1},$$
$$x_{2,v_0,v_1}, x_{2,v_1,v_0}, x_{2,v_1,v_2}, x_{2,v_2,v_1},$$
$$x_{v_0}, x_{v_1}, x_{v_2}, x_{O_1}, x_{O_2}.$$

The objective function is to minimize the number of obstacles that intersect with paths:

$$\text{minimize} \quad x_{O_1} + x_{O_2}.$$

By constraint (5),

$$x_{1,v_0,v_1} = x_{1,v_0,v_1} + x_{1,v_2,v_1} = 1,$$
$$x_{2,v_1,v_0} + x_{2,v_1,v_2} = x_{2,v_1,v_2} = 1.$$

By constraint (6),

$$x_{1,v_1,v_0} = x_{1,v_1,v_0} + x_{1,v_1,v_2} = 0,$$
$$x_{2,v_0,v_1} + x_{2,v_2,v_1} = x_{2,v_2,v_1} = 0.$$

By constraint (7),

$$x_{1,v_2,v_1} = x_{1,v_1,v_2} \leq 1,$$
$$x_{2,v_0,v_1} = x_{2,v_1,v_0} \leq 1.$$

Note that constraint (8) is not necessary in MMCR since additional subtours cannot decrease the value of the objective function.

By constraint (13),

$$Lx_{v_0} \geq x_{1,v_0,v_1} + x_{1,v_1,v_0} + x_{2,v_0,v_1} + x_{2,v_1,v_0},$$
$$Lx_{v_1} \geq x_{1,v_1,v_0} + x_{1,v_0,v_1} + x_{1,v_1,v_2} + x_{1,v_2,v_1} +$$
$$x_{2,v_1,v_0} + x_{2,v_0,v_1} + x_{2,v_1,v_2} + x_{2,v_2,v_1},$$
$$Lx_{v_2} \geq x_{1,v_2,v_1} + x_{1,v_1,v_2} + x_{2,v_2,v_1} + x_{2,v_1,v_2}.$$

By constraint (14),

$$Lx_{O_1} \geq x_{v_0} + x_{v_1},$$
$$Lx_{O_2} \geq x_{v_1} + x_{v_2}.$$

D. RCP

For this section, we only demonstrate an implementation example for OTP since the formulation can be roughly

considered as QCOP with extra variables and constraints. We continue to use the graph in Fig. 13. Consider the three vertices are associated with three linear reward functions R_0, R_1, R_2; the robot starts from vertex v_0; the cost of edge (v_0, v_1) and (v_1, v_2) are c_{01} and c_{12}; the maximum cost is c^*; $X^I = X^G = \{v_0, v_1\}$. Using a time-expanded encoding method and with the additional virtual vertex u, the binary variables in our IP formulation for $T = 2$ are:

$$x^0_{u,v_0}, x^0_{u,v_1}, x^2_{v_0,u}, x^2_{v_1,u},$$
$$x^1_{v_0,v_0}, x^1_{v_0,v_1}, x^1_{v_1,v_0}, x^1_{v_1,v_1}, x^1_{v_1,v_2}, x_{v_0}, x_{v_1}, x_{v_2}.$$

The three additional continuous variables are:

$$t_0 \geq 0, t_1 \geq 0, t_2 \geq 0.$$

The objective function is:

$$\text{maximize} \quad R_0(t_0) + R_1(t_1) + R_2(t_2).$$

By constraint (4),

$$c_{01} * (x^1_{v_0,v_1} + x^1_{v_1,v_0}) + c_{12} * x^1_{v_1,v_2} + t_0 + t_1 + t_2 \leq c^*.$$

By constraint (15) (instead of (9)),

$$x^0_{u,v_0} + x^0_{u,v_1} = x^2_{v_0,u} + x^2_{v_1,u} = 1,$$

By constraint (10),

$$x^1_{v_0,v_0} + x^1_{v_0,v_1} = x^0_{u,v_0},$$
$$x^1_{v_1,v_0} + x^1_{v_1,v_1} + x^1_{v_1,v_2} = x^0_{u,v_1},$$
$$x^2_{v_0,u} = x^1_{v_0,v_0} + x^1_{v_1,v_0},$$
$$x^2_{v_1,u} = x^1_{v_0,v_1} + x^1_{v_1,v_1}.$$

By constraint (16),

$$x_{v_0} \leq x^0_{u,v_0} + x^1_{v_0,v_0} + x^1_{v_1,v_0},$$
$$x_{v_1} \leq x^0_{u,v_1} + x^1_{v_0,v_1} + x^1_{v_1,v_1},$$
$$x_{v_2} \leq x^1_{v_1,v_2}.$$

Additional constraints for OTP are:

$$x^0_{u,v_0} = x^2_{v_0,u}, x^0_{u,v_1} = x^2_{v_1,u}$$
$$t_0 \leq Lx_{v_0}, t_1 \leq Lx_{v_1}, t_2 \leq Lx_{v_2}.$$