Mixed-Integer and Constraint Programming Techniques for Mobile Robot Task Planning

Kyle E. C. Booth, Tony T. Tran, Goldie Nejat, *Member, IEEE*, and J. Christopher Beck

***Abstract*—We investigate the use of optimization-based techniques to model and solve two real-world single robot task planning problems. In the first problem, a robot must plan a set of tasks, each with different temporal constraints. In the second problem, a socially interacting robot must plan a set of tasks while considering the schedules of multiple human users, as well as physical constraints including battery level. We apply existing mixed-integer programming and constraint programming techniques, yielding two exact methods for each problem. Numerical experiments show that both approaches produce better solutions in less time compared to techniques previously proposed for these problems. In order to confirm their physical utility, we implement the plans for the second problem in both simulated and real environments on Tangy, a socially interacting robot. We conclude that both mixed-integer programming and constraint programming are promising general approaches to robot task planning that should be considered when solving these problems.**

***Index Terms*—Companion robots, coordination, planning, scheduling.**

I. INTRODUCTION

As autonomous robots take on everyday applications, the demand increases for efficient techniques for autonomous decision making. Automated reasoning for the planning and scheduling of tasks is a core component of intelligent behavior and, as such, is fundamental to the design of autonomous mobile robots [1]. The promise of societal and economic benefits from robot technology will be substantially muted if robots must be provided with an explicit plan of tasks they need to undertake.

Task planning and scheduling for mobile robotics has been studied in a variety of applications, including container transportation [2], office assistance [3], [4], planetary rovers [5], warehouse management [6], hospital assistance [7], and human care [7]–[10]. In Coltin et al. [4], they study task planning for a mobile robot that receives task requests from users via an online web utility. Upon receiving a collection of task requests, the robot makes inferences about the relationship between task time windows and then generates a plan using a mixed-integer programming (MIP) solver. Mudrova and Hawes [10] address the same single-robot task allocation (SRTA) problem with a customized interval-algebra-based algorithm to make implied and heuristic sequencing decisions before assigning start-times via a MIP solver. In our previous work [11], we used a forward-chaining temporal planner for a SRTA problem in a multi-user environment, where the socially interactive robot builds plans to facilitate a number of interaction activities while considering user timetables.

In this paper, we focus on the application of existing, general-purpose scheduling technologies from the operations research (e.g., [12], [13]) and artificial intelligence (e.g., [14], [15]) literatures to robot task planning problems. Our hypothesis is that high-performance optimization techniques from these communities can be used to produce efficient, complete plans in run-times that are realistic for these applications. While generic optimization technologies, notably mixed-integer programming, have been applied to robotics applications [4], [16], [17], they have not been exploited to their full potential for mobile robotics applications.

We consider two SRTA problems from the literature. In the first [4], a robot must plan a set of tasks, each with different temporal constraints, with the goal of finding a feasible plan that minimizes the sum of task completion times. In the second problem [11], a socially interacting robot must plan a set of tasks while considering multi-user timetables and physical constraints. The problem involves reasoning about which activities the robot should implement, as well as when these activities should occur, while taking into account the individual availabilities of users, robot battery level, energy consumption, and location transition times.

We investigate the modeling and solving of robot task planning problems with two existing optimization formalisms: mixed-integer programming and constraint programming. Our contributions center on the use of these optimization technologies to quickly find high-quality task plans. On the problems studied, we show our methods overall determine better solutions in shorter run-time than the approaches presented in previous works. For the second problem, we implement simulated and real-environment experiments on the mobile social robot Tangy, demonstrating that our techniques yield implementable, high-quality task plans for mobile robots in a human-robot interaction setting.

II. BACKGROUND

In this section we introduce mixed-integer programming (MIP) and constraint programming (CP), illustrating that

Manuscript received: August, 31, 2015; Revised November, 26, 2015; Accepted January, 5, 2016.

This paper was recommended for publication by Editor J. Li upon evaluation of the Associate Editor and Reviewers' comments. This research has been funded by the Natural Sciences and Engineering Council of Canada (NSERC), Dr. Robot Inc., and the Canada Research Chairs (CRC) Program.

K.E.C. Booth, T. Tran, G. Nejat, and J.C. Beck are with the Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario, Canada. kbooth@mie.utoronto.ca, tran@mie.utoronto.ca, nejat@mie.utoronto.ca, jcb@mie.utoronto.ca

Digital Object Identifier (DOI): see top of this page.

although these methods can often be applied to the same problems, they represent fundamentally different methodologies for dealing with the combinatorial explosion inherent in many scheduling problems. MIP and CP represent mature technologies that have benefited from many years of research, development, and application to $\mathcal{NP}$-complete problems [12], [18].

A. Mixed-Integer Programming

Mixed-integer programming (MIP) is a mathematical optimization approach for problems modeled as a set of decision variables taking on either continuous or integer values (hence "mixed"), constrained by linear constraints and with the goal of optimizing a linear objective function. MIP has been used widely for optimization since the 1950s, including many scheduling problems [12], [13], [19].

The standard solution approach for MIP is *branch-and-bound* tree search [20]. The key to (often) avoiding the worst-case exponential search is the fact that ignoring the integrality requirement on the integer variables results in a polynomial solvable linear programming (LP) *relaxation* of the MIP. The LP is solved at each node in the branch-and-bound tree to provide a bound on the objective function and, once a feasible solution is found, the ability to prune sub-trees that are proved not to contain solutions better than the current best solution. The solution to the LP relaxation is also extensively used heuristically to drive the order in which sub-trees are explored.

Modern-day MIP solvers combine branch-and-bound search with techniques such as cutting planes [21] to solve large, complex problems. Along with a description of the historical development of MIP solving, Bixby [22] shows a machine-independent speed-up of over 400,000 times in commercial MIP solvers from the early 1990s to 2012.

B. Constraint Programming

Constraint programming (CP) is more general than MIP, allowing variable types beyond integer and continuous (e.g., interval [23] and set variables [24]), and dropping the restriction of linearity in the constraints and objective function. Problem modeling focuses on combining *global constraints* [25], encapsulations of frequently recurring combinatorial substructure. Developed primarily within the artificial intelligence community, CP has also been applied to a wide range of combinatorial optimization problems with scheduling being one of the most successful commercial areas [18].

In contrast to MIP, CP reduces search effort through logical *inference* [26]. Each constraint has an inference algorithm that performs *domain filtering*: removing possible values from the domains of its variables by proving that a value cannot satisfy the constraint itself and, therefore, cannot participate in a global solution. Given that variables typically participate in multiple constraints, value removal by one constraint often triggers subsequent removals from the domains of other variables in neighboring constraints, resulting in the *propagation* of inferences through the problem representation. The overall solution process is again a tree search but at each node, each constraint performs inference, removing values that are no longer locally consistent. The results of the propagation are then used to inform heuristics which choose the order in which the search tree is explored.

CP solvers have seen significant advances in efficiency in recent decades, transitioning from logic programming into more optimization-based paradigms and becoming an alternative to MIP-based approaches [27].

III. ROBOT TASK PLANNING PROBLEM #1

We consider a problem in which a robot must autonomously plan a set of tasks where each task has its own temporal constraints. We present a formal definition of the problem, propose two scheduling models, and compare these models to existing solution techniques. We identify situations where the existing techniques do not provide a plan due to their incompleteness.

A. Problem Definition

Given a set of tasks, $j \in J$, with time windows and sequence-dependent setup times, the goal is to find a feasible plan, if one exists, over a planning horizon, H, such that the sum of completion times across all tasks is minimized.

Following standard scheduling terminology, this problem can be represented as $1|r_j, d_j, \delta_{jk}| \sum C_j$, where 1 refers to the single robot resource, r_j is the release time of task j, d_j is the deadline time for task j, δ_{jk} is the minimum "setup" time that must occur between pairs of tasks j, k, and $\sum C_j$ is the sum of completion times objective function. The setup time between two tasks, j preceding k, represents robot travel time and is defined as the non-negative time that must elapse between the end of task j and the start of task k. This relation may be asymmetric such that $\delta_{jk} \neq \delta_{kj}$ holds. We assume that setup times follow the triangle inequality, namely $\delta_{jl} + \delta_{lk} \geq \delta_{jk}$.

A solution is a set of start times, s_j, for each task j such that the start and end time, e_j, adhere to its time window (i.e., $[s_j, e_j] \in [r_j, d_j]$), the setup time between each task pair is respected, and the sum of the completion times is minimized.

B. Proposed Scheduling Models

We propose two optimization models for solving this SRTA problem using MIP and CP, respectively.

1) MIP Model: We propose a disjunctive MIP model as is common in the literature [21]. Eqs. (1) through (6) define the model. The objective function (1) represents the minimization

$$\min \quad \sum_j C_j \tag{1}$$

$$\text{s.t.} \quad C_j = s_j + p_j, \qquad \forall j \tag{2}$$

$$C_j + \delta_{jk} \leq s_k + (H + \delta_{jk})(1 - x_{jk}), \qquad \forall j, k \tag{3}$$

$$C_k + \delta_{kj} \leq s_j + (H + \delta_{kj})(x_{jk}), \qquad \forall j, k \tag{4}$$

$$x_{jk} \in \{0, 1\} \qquad \forall j, k \tag{5}$$

$$s_j \in [r_j, d_j - p_j] \qquad \forall j \tag{6}$$

of the sum of completion times over all tasks. Constraint (2) defines the completion time of a task $j \in J$ as the sum of the start time of the task, s_j, and its processing time, p_j. Constraints (3) and (4) represent disjunctive sequencing constraints for tasks that prevent them from overlapping: x_{jk} is a binary decision variable that is assigned a value of 1 if task j precedes k and 0 otherwise. When $x_{jk} = 1$, the second summand on the right hand side of (3) evaluates to 0, forcing the start time of task k to be greater than the completion time of j plus the setup time from j to k. Similar reasoning applies for (4) when x_{jk} is assigned a value of 0. Thus, these two constraints enforce the disjunctive meaning of the x_{jk} variables. Eqns (5) and (6) define the continuous positive and binary domains for s_j and x_{jk}, respectively.

2) CP Model: Expressions (7) through (10) define the CP model. Eq. (7) is the objective function and constraint (8) defines the completion time of each task. Constraint (9) is the global constraint that ensures that the tasks do not overlap. Constraint (10) defines the domain for the start time of the tasks. The CP model is very similar to the

$$\begin{aligned} \min \quad & \sum_j C_j && && (7)\\ \text{s.t.} \quad & C_j = s_j + p_j, && \forall j && (8)\\ & NoOverlap(s_j, p_j, \delta_{jk}), && \forall j,k && (9)\\ & s_j \in [r_j, d_j - p_j] && \forall j && (10) \end{aligned}$$

MIP model and is implemented using interval variables [23], decision variables whose possible values are convex intervals: $\{[f,g) | f,g \in \mathbb{Z}, f \leq g\}$, where f and g are the start and end values of the interval. We introduce an interval variable for each $j \in J$, where each of these is defined by r_j, d_j, and p_j. We also utilize the *NoOverlap* global constraint in place of Constraints (3) and (4). This constraint performs efficient, incomplete domain filtering of the start-time variables by reasoning about the task time windows, processing times, and the relationship that no pair of tasks can overlap in time, including the setup times [15].

C. Existing Techniques

1) Dynamic User Task Scheduling: Coltin et al. [4] developed the dynamic user task scheduling (DUTS) technique using MIP. The approach introduces constraints (3') and (4') for all pairs of tasks that have overlapping time windows.

$$s_j + p_j + \delta_{jk} - s_k \leq |d_j - r_k|(1 - x_{jk}) \quad \forall j,k \qquad (3')$$

$$s_k + p_k + \delta_{kj} - s_j \leq |d_k - r_j| x_{jk} \quad \forall j,k \qquad (4')$$

We label these constraints (3') and (4') as they appear to serve the same disjunctive reasoning as constraints (3) and (4) in our proposed model. The rest of the DUTS model is the same as the model proposed above.[1] As shown in the following simple example, constraints (3') and (4') do not guarantee completeness: DUTS will fail to find a feasible solution in some situations where a feasible solution exists.

[1]Following [10], [28] we use d_j to denote the latest end time of task j. Similar conclusions can be easily shown if latest start time is used to define the end of the window. We also amend the indices in the $|d - r|$ terms in [4].

Consider an instance with two tasks, 1 and 2, with identical setups, processing times, and release and deadline times (e.g., $\delta_{1,2} = \delta_{2,1} = 1$, $p_1 = p_2 = 1$, $r_1 = r_2 = 0$, and $d_1 = d_2 = 3$). Both tasks can be completed within their time windows with either ordering. However, when task 1 is scheduled to precede 2, the binary decision variable $x_{1,2} = 1$, causing the constraints (3') and (4') to simplify to $s_1 \leq s_2 - 2$ and $s_1 \geq s_2 - 1$, which is infeasible. A similar infeasibility follows if we were to assign the opposite ordering.

The modeling of disjunctive constraints in MIP usually involves a large constant term (i.e., $(H + \delta_{jk})$ in our model). It may be that this pattern is what is attempted in [4], however, an insufficiently large constant term was chosen, cutting off some feasible solutions. Though such cases will not occur in all problem instances, DUTS is incomplete: it is not guaranteed to find a feasible solution, even if one exists.

2) Task Scheduling with Interval Algebra: Mudrova and Hawes [10] address the same problem with the task scheduling with interval algebra (TSIA) approach to heuristically order each task pair. First, an ordering is asserted for all pairs for which an ordering can be inferred based on reasoning about the pair of time windows and durations [29]. Second, if both orderings are possible, TSIA heuristically selects the one that either i) maximizes the time available for the tasks, or ii) locally minimizes the sum of completion times of those two tasks, depending on the relationship between the time windows of the two tasks. A complete pairwise ordering is, thus, imposed. A MIP solver, is then used to assign the start times of the tasks consistent with the ordering.

Note that as the pairwise task analysis results in a total ordering of the tasks, a MIP model is unnecessary. If a feasible start time assignment exists for a given total ordering, it is sufficient to sort the tasks according to the ordering and assign the start time of each task to the maximum of: i) its release time, and ii) the completion time plus setup time of the previous task in the ordering. This algorithm will minimize sum of completion times or show that no feasible assignment exists *for a given total ordering*.

However, the TSIA approach is also incomplete: it may produce pairwise orderings with no feasible start time assignment and so fail to find a feasible solution when one exists. Consider the problem instance with three tasks with identical processing times (e.g., $p_1 = p_2 = p_3 = 1$) and setup times ($\delta_{jk} = \delta_{kj} = 1, \forall j,k \in \{1,2,3\}$) and with time windows $[2,4]$, $[0,5]$, and $[1,6]$ for tasks 1, 2, and 3, respectively.

There exists a feasible solution: $2 \rightarrow 1 \rightarrow 3$. However, TSIA heuristically produces the following pairwise orderings: $\{2 \rightarrow 3,\ 2 \rightarrow 1,\ 3 \rightarrow 1\}$ which result in an infeasible plan: task 1 can start no earlier than time 4 but its time window constrains it to finish by time 4.

We note that TSIA is proposed as a heuristic approach and is not guaranteed to find a globally optimal solution. For large problems, heuristics may be preferred because global optimality may be unreachable in a reasonable time. Therefore, in our simulation experiments (Section V below), we evaluate the solution-quality vs. run-time trade-off of the different solution approaches.

IV. ROBOT TASK PLANNING PROBLEM #2

In this section we define the second robot task planning problem, propose MIP and CP models, and review the previous solution method [11].

A. Problem Definition

The objective of the Tangy robot is to plan and execute a set of recreational activities throughout the day while considering multiple user timetables [11]. Given a *planning horizon* of 8am to 7pm, a daily plan is to be generated for a single robot consisting of the tasks the robot will facilitate (bingo games and reminders) each with associated participants, location, start time, processing time, and end time. Each user has a *timetable* identifying the times in which he/she is available for robot-facilitated activities including three mandatory breaks (mealtimes) from 8am-9am, 12pm-1pm, and 5pm-6pm. There are also two or three other one-hour unavailable intervals for each user with varying times depending on user and problem instance. Each user has his/her own personal room where, it is assumed, he/she will be whenever available.

A set of required *bingo games* with specified participants are defined. The robot assists players during bingo by repeating missed numbers and reviewing player cards. The robot must also perform *reminder* tasks by navigating to each participant's room and communicating the reminder. In addition to bingo game and reminder tasks, the robot must execute a *move* task to navigate in the facility. We assume that the travel times between any pair of relevant points are known.

The instantaneous battery level of the robot is available and the problem instance defines maximum and minimum allowable battery levels. Each task type (bingo, reminder, move) has its own rate of energy consumption. If the battery level gets too low, the robot can perform a *recharging* task by navigating to the location of the *recharging station*. There is one recharging station, with a specified location. The robot recharge tasks can be either *absent* or *present*: they are optional tasks. We use a calculated upper bound for each instance to determine how many optional recharge tasks to include in the models.

Given the user timetables, bingo games, and parameter values, the planning problem is to determine the start times for each bingo game together with the robot tasks so that it can feasibly conduct each bingo game with the specified users while maintaining sufficient battery level and while also delivering a reminder to each user before his/her bingo game. These actions include the need for travel between locations within the facility.

B. Proposed Scheduling Models

We model the Tangy mobile robot task planning problem with MIP and CP technologies and solve these models to produce feasible plans.

1) Model Parameters: The parameters used in both the MIP and CP models are as follows:

U: Set of users,
G: Set of bingo game tasks,
M: Set of reminder tasks,
M_g: Subset of reminder tasks for each bingo game $g \in G$,
C: Set of charging tasks,
A: Set union of all tasks $G \cup M \cup C$,
$\dot{a}$: An auxiliary starting task (for sequencing),
$\ddot{a}$: An auxiliary ending task (for sequencing),
$\bar{A}$: Set A including both auxiliary tasks,
A_u: Subset of all tasks that involve user u,
$calendar_u$: Individual timetable of user u,
p_j: Duration of task j (not recharging tasks),
δ_{jk}: Transition time between task j and k, $\forall j, k \in \bar{A}$,
b_min, b_max: Minimum and maximum battery levels,
r_rate: Rate at which robot recharges,
$e\_move, e\_bingo, e\_remind$: Robot energy consumption rate for move, bingo and reminder tasks respectively,
H: The planning horizon for the instance,
M: A large positive number used in disjunctive reasoning.

We formulate each model to minimize a generic objective, where each task in the final plan is defined by a type, location, start time, duration and end time.

2) MIP Model: Expressions (11) through (32) define the MIP model. Decision variable w_j has a value of 1 if task j is scheduled, and a value of 0 otherwise. Decision variable z_{jk} has a value of 1 if task j occurs *immediately* before task k and 0 otherwise. Decision variable y_{jt} has a value of 1 if task j is scheduled at time t and a value of 0 otherwise. We use y_{jt} to relate scheduled tasks with user timetables, $calendar_u$. We then introduce decision variables ϵ_j and E_j which represent the energy consumption of task j, and the energy level of the robot after completing task j respectively. Finally, we use decision variable D_j to represent the duration of task j. $D_j = p_j$ for tasks with fixed duration (i.e., excepting recharging tasks).

Expression (11) represents a generic objective function. Constraint (12) sets the decision variable w_j for all tasks, except recharge tasks, to be equal to 1, indicating that these tasks are required. Constraint (13) forces recharging tasks to be used in lexicographic order, breaking some of the symmetry in the model. Constraints (14) and (15) ensure that the start times of the tasks adhere to the user and task schedules where T represents all time points t in $0 \leq t \leq H$ and T_j represents the time points during which task j can be scheduled to start. Constraint (16) sets the duration of fixed-length tasks (bingo games and reminders) to be equal to their defined processing time. Constraint (17) ensures that bingo tasks start after the corresponding reminders are executed, and (18) enforces a sequential relationship between a pair of tasks j, k where k immediately follows j. Constraints (19) and (20) ensure that the sequencing variable z_{jk} across all pairs of tasks is constrained properly to w_j. Constraint (21) links y_{jt} to s_j. Constraint (22) constrains the energy level of the robot at the end of task k to depend on its own energy consumption, as well as the energy consumption (and travel) of the task before it, j. Constraints (23) through (25) define the energy consumption of the various tasks. Eqs. (26) through (32) represent the domains for the decision variables as binary, positive continuous or real numbers.

$$\min \quad objective \tag{11}$$
$$\text{s.t.} \quad w_j = 1, \qquad \forall j \in \bar{A} \setminus C \tag{12}$$
$$w_j \geq w_k, \qquad \forall j, k \in C \mid j < k \tag{13}$$
$$\sum_{t \in T} y_{jt} = w_j, \qquad \forall j \in A \tag{14}$$
$$\sum_{t \in T \setminus T_j} y_{jt} = 0, \qquad \forall j \in A \tag{15}$$
$$D_j = p_j, \qquad \forall j \in \bar{A} \setminus C \tag{16}$$
$$s_g \geq s_j, \qquad \forall g \in G, j \in M_g \tag{17}$$
$$s_k \geq s_j + D_j + \delta_{jk} + H(z_{jk} - 1), \qquad \forall j, k \in \bar{A} \tag{18}$$
$$\sum_{j \in A \cup \dot{a}} z_{jk} = w_k, \qquad \forall k \in A \cup \ddot{a} \tag{19}$$
$$\sum_{k \in A \cup \ddot{a}} z_{jk} = w_j, \qquad \forall j \in A \cup \dot{a} \tag{20}$$
$$\sum_{t \in T} y_{jt} t = s_j, \qquad \forall j \in A \tag{21}$$
$$E_k \leq E_j - z_{jk}\delta_{jk} e\_move - \epsilon_k + M(1 - z_{jk}), \qquad \forall j, k \in \bar{A} \tag{22}$$
$$\epsilon_j = D_j \times e\_bingo, \qquad \forall j \in G \tag{23}$$
$$\epsilon_j = D_j \times e\_remind, \qquad \forall j \in M \tag{24}$$
$$\epsilon_j = -D_j \times r\_rate, \qquad \forall j \in C \tag{25}$$
$$w_j, \in \{0, 1\}, \qquad \forall j \in C \tag{26}$$
$$z_{jk}, \in \{0, 1\}, \qquad \forall j, k \in \bar{A} \tag{27}$$
$$y_{jt}, \in \{0, 1\}, \qquad \forall j \in A, t \in T \tag{28}$$
$$s_j \in [0, H], \qquad \forall j \in \bar{A} \tag{29}$$
$$0 \leq D_j \leq \frac{b\_max}{r\_rate}, \qquad \forall j \in C \tag{30}$$
$$\epsilon_j \in \mathbb{R}, \qquad \forall j \in \bar{A} \tag{31}$$
$$b\_min \leq E_j \leq b\_max \qquad \forall j \in \bar{A} \tag{32}$$

3) CP Model: The CP model is defined by Eqns. (33) to (47). The parameters used are the same as for the MIP model, presented above. The model uses optional interval variables [23], defined similarly to those used in Problem #1: $\{\perp\} \cup \{[f, g) | f, g \in \mathbb{Z}, f \leq g\}$, with the addition of $\perp$, a special value that indicates that the variable is not present in the solution. We introduce an optional interval variable a_j for each task $j \in A$, where each of these is defined by a start time, end time, and processing time.

The cumulative function, e_level, represents the battery level of the robot over time. The variables ϵ_j represent the energy consumption of task j, and θ_j represents the total energy consumption of task j including robot travel to its location. Eq. (33) represents the generic objective function for the minimization problem. Constraint (34) is the $NoOverlap$ global constraint, as defined previously. In constraint (35) we make use of the $ForbidExtent$ global constraint to ensure that none of the tasks for a user conflict with the timetable of that user. $ForbidExtent(a_j, F)$ prevents an interval variable a_j from overlapping a time point t where $F(t)$, an integer step function, is equal to 0 [23]. In our case, individual user calendars, $calendar_u$, detail the t values for which $F(t) = 0$.

In constraint (36) we constrain the start of a bingo game to be after the end of the reminders associated with that bingo game. Constraint (37) sets the length of all interval variables (not including recharging tasks) to be equal to the processing time of the associated task. Constraint (38) sets the domain for the length of the charging tasks to be between zero and an upper bound defined as maximum battery capacity divided by the recharge rate of the robot. Constraint (39) ensures that each recharging task is used in the proper order.

Constraints (40) through (42) set the energy consumption of task j to be proportional to its duration and consumption parameter. We note that a recharging task has a negative energy consumption. Expression (43) constrains θ_j to be equivalent to the base energy consumption of the task, ϵ_j, plus additional energy consumption for robot travel to the location of task j from the preceding task. To accomplish this, we use the function $preLoc(a_j)$ that returns the location of the task directly before a_j in the solution sequence. Constraint (44) represents a step function constraint on the robot energy level to adjust the robot energy level as each task is completed. The $StepAtStart$ is a cumulative function expression, representing the individual contribution of an interval variable to a cumulative value [23]. In our model, each time an interval variable a_j is executed, it has an instantaneous impact of energy decrease, θ_j, on the cumulative value, e_level, at its start time.

Expression (45) constrains the start time of all interval tasks to be within the allotted time horizon and Expression (46) uses $PresenceOf$ to establish recharging tasks as optional. $PresenceOf(a_j)$ represents whether an interval variable a_j is present in the solution, and thus, whether the constraints should apply to it [23]. If it is decided that $PresenceOf(a_j) = 0$, then the task is not present, and thus, not constrained or constraining. Eq. (47) sets bounds for the energy level to be between the maximum and minimum battery levels.

Recall that each of these global constraints represents a relationship amongst a set of variables and encapsulates an inference algorithm that, during search, remove values from the variable domains when it can be inferred that the value no longer satisfies the global constraint.

C. Existing Techniques

In our previous work [11], we used the Planning Domain Definition Language (PDDL), the standard problem definition language in AI planning [30], to define the domain and problem instances. Space limitations prevent a full presentation of the PDDL model here (see Louie et al. [11]). We used OPTIC [31], a forward-chaining partial order temporal planner, to generate a feasible plan. OPTIC was tested on five scenarios that varied in the number of users and bingo activity requests.

$$\min \quad objective \tag{33}$$
$$\text{s.t.} \quad NoOverlap(a_j, \delta_{jk}), \quad \forall j,k \in A \tag{34}$$
$$ForbidExtent(a_j, calendar_u), \quad \forall j \in A_u, u \in U \tag{35}$$
$$Start(a_g) \geq End(a_j), \quad \forall g \in G, j \in M_g \tag{36}$$
$$Length(a_j) = p_j, \quad \forall j \in A \setminus C \tag{37}$$
$$0 \leq Length(a_j) \leq \frac{b\_max}{r\_rate}, \quad \forall j \in C \tag{38}$$
$$PresenceOf(a_j) \geq PresenceOf(a_k), \quad \forall j,k \in C \mid j < k \tag{39}$$
$$\epsilon_j = Length(a_j) \times e\_bingo, \quad \forall j \in G \tag{40}$$
$$\epsilon_j = Length(a_j) \times e\_remind, \quad \forall j \in M \tag{41}$$
$$\epsilon_j = -Length(a_j) \times r\_rate, \quad \forall j \in C \tag{42}$$
$$\theta_j = \epsilon_j + \delta_{preLoc(a_j),j} e\_move, \quad \forall j \in A \tag{43}$$
$$e\_level = \sum_{j \in A} StepAtStart(a_j, -\theta_j), \quad \forall j \in A \tag{44}$$
$$Start(a_j) \in [0, H], \quad \forall j \in A \tag{45}$$
$$PresenceOf(a_j) \in \{0,1\}, \quad \forall j \in C \tag{46}$$
$$b\_min \leq e\_level \leq b\_max \tag{47}$$

V. IMPLEMENTATION

In this section we present simulation results for both task planning problems. We define algorithm performance based on computational run-time and solution quality, quantified by the optimality gap of the produced plan. Given the need for a robot to plan in a real environment, the time taken to solve a problem is particularly important. It is often the case that quickly-found, high-quality solutions are more valuable than optimal solutions found more slowly. Therefore, one of our primary interests is in the evaluation of the trade-off between solution quality and algorithm run-time.

All methods are implemented in C++ on a hexacore machine with a Xeon processor and 12GB of RAM running Linux. We use the IBM ILOG CPLEX 12.6.2 Optimization Studio single-threaded for all simulations. The CPLEX Optimization Studio includes multiple solvers. In particular, our MIP models are solved with the CPLEX MIP solver while the CP models are solved with CPLEX CP Optimizer, a wholly different solver.

Problem #1 is simulated using a random instance generation method developed in C++, similar to that described by Mudrova and Hawes [10]. Problem #2 is simulated using the scenarios of Louie et al. [11]. We produce plans with our methods and then implement them in an example retirement home facility with ROS Visualization [32]. As a proof of concept, we also implement the task plan resulting from the CP model for Scenario 1 on the mobile robot Tangy, using OpenSlam's (openslam.org) GMapping to create our environment map via simultaneous localization and mapping (SLAM).

A. Robot Task Planning Problem #1 - Simulation Results

Table I shows the mean relative error (MRE) of the best solution found by the given technique at the run-time corresponding to the column. For example, the performance of CP for $P40$, the set of five problem instances with 40 tasks, at the run-time 0.01 seconds is calculated as follows:

$$MRE_{(CP,P40,0.01)} = \sum_{p \in \mathcal{F}} \frac{c(CP,p,0.01) - c^*(p)}{|\mathcal{F}| \times c^*(p)} \times 100 \tag{48}$$

where $\mathcal{F} \subseteq P40$ is the set of instances in $P40$ with feasible solutions at 0.01 seconds using CP, $c(CP,p,0.01)$ is the best solution found by CP in problem instance p at 0.01 seconds, and $c^*(p)$ is the optimal solution, if known, or the best known lower bound for instance p found by running MIP with an 18,000 second time-out. A value of '-' indicates that the technique failed to find a feasible plan for all five instances at that run-time. MRE values with a '†' are calculated from the subset of instances for which the method found a feasible plan at the corresponding run-time. The '# Inf.' column represents the number of instances for which no feasible plan was found after 100 seconds of run-time for a technique. Bolded values indicate the technique found the most feasible plans at the corresponding run-time with ties decided by the lowest MRE.

TABLE I
PROBLEM #1 SIMULATION: MEAN RELATIVE ERROR (%) OVER TIME

		Run-time (s)					
Tasks	Method	0.01	0.1	1	10	100	# Inf.
40	**CP**	**0.33**	**0.08**	**0.00**	**0.00**	**0.00**	0
	MIP	-	7.93	0.13	**0.00**	**0.00**	0
	DUTS	22.43†	13.10	0.06	0.02	0.02	0
	TSIA	0.98	0.98	0.98	0.98	0.98	0
80	**CP**	**0.37**	**0.32**	**0.15**	**0.10**	**0.10**	0
	MIP	-	9.02	1.38	0.11	0.11	0
	DUTS	-	10.23	4.49	0.15	0.12	0
	TSIA	0.45†	0.45†	0.45†	0.45†	0.45†	**2**
120	**CP**	**0.56**	**0.37**	**0.34**	**0.25**	**0.24**	0
	MIP	-	6.60†	3.67	**0.25**	0.25	0
	DUTS	-	7.06†	4.48	0.28	0.25	0
	TSIA	0.40†	0.40†	0.40†	0.40†	0.40†	**4**
160	**CP**	-	**0.33**	**0.30**	**0.23**	**0.22**	0
	MIP	-	-	4.07	1.13	0.23	0
	DUTS	-	4.74†	3.08	0.85	0.23	0
	TSIA	**0.33**†	0.33†	0.33†	0.33†	0.33†	**4**
200	**CP**	-	**0.26**	**0.25**	**0.20**	**0.18**	0
	MIP	-	-	3.56	1.63	**0.18**	0
	DUTS	-	4.77	3.83	1.93	**0.18**	0
	TSIA	-	-	-	-	-	**5**

Overall, the results indicate that CP very quickly finds high quality solutions without sacrificing completeness. CP outperforms all approaches at each run-time, except P160 at 0.01 seconds where TSIA is able to find a solution for a single instance. Due to its heuristic ordering, TSIA cannot improve upon initial feasible solutions, nor does it find any solutions for the largest problems and a portion of medium-sized problems. The DUTS method, as a MIP model very similar to our own proposed technique, is overall slightly outperformed by our MIP model for these instances.

TABLE II
PROBLEM #2 SIMULATION: TIME TO FIRST FEASIBLE PLAN

Scenario	Users	Bingo	**CP**	MIP	OPTIC
Scenario 1	4	1	$<$ **0.01**	0.01	0.54
Scenario 2	8	2	$<$ **0.01**	0.36	9.13
Scenario 3	12	3	**0.04**	1.30	13.09
Scenario 4	16	4	**0.01**	-	-
Scenario 5	20	5	**0.08**	-	-

TABLE III
PROBLEM #2 SIMULATION, CP: SENT ACTIVITY COMMAND RESULTS

	Expected Activity from Robot				
Commands	Remind	Play Bingo	Recharge	Total	Success (%)
Remind	58	0	0	60	96.7
Play Bingo	0	15	0	15	100.0
Recharge	0	0	3	3	100.0

B. *Robot Task Planning Problem #2 - Simulation Results*

Following Louie et al., we evaluate the performance of our models on our second task planning problem as a feasibility problem (i.e., no objective function) in a realistic environment. The problem differs from that solved by Louie et al. as we revised the parameter values to provide better estimates for battery capacities, consumption rates and robot location transition times. We use run-time as our performance metric and enforce a 10-minute time limit on all experiments.

In Table II, the CP and MIP models produce feasible plans much faster than OPTIC. CP is able to generate feasible plans three orders of magnitude faster than OPTIC and roughly two orders of magnitude faster than MIP. The use of a feasibility problem favours OPTIC over the optimization techniques given the relative focus on optimization versus feasibility in the respective literatures. Our experiments with various objective functions (not shown) indicate that OPTIC cannot typically improve its initial feasible solution while both CP and MIP are able to find improving solutions.

Due to the extended nature of these full-day task planning problems, we use ROS Visualizer to conduct experiments on Scenarios 1-5 to ensure the plans being generated are implementable based on robot and user constraints. These experiments involve simulating the robot and the users within an example retirement home facility layout as seen in Figure 1. We ran the simulation on all scenarios with the plans produced by the CP technique.

The top of Figure 1 illustrates the layout of the facility, the users (red) and the robot (green). This image, captured during simulation, depicts a bingo game reminder in a user's personal room. The lower portion of the figure represents the robot and user timetables and identifies the plan progress as it is executed.

Table III presents the results of our implementation of the CP-produced plans within the simulated environment; we experiment with all five scenarios. After some small adjustments to account for user movement within the environment, we achieved nearly a 100% success rate on the proper command/activity from the robot for reminder, bingo game and recharge activities. The success rate is defined as the number of correct activity actions implemented by the robot over the total commands received by the robot from the planner. Two reminder tasks failed to execute properly due to an insufficient transition time allotted for the robot to arrive at its planned reminder location.

As a proof of concept, we conducted a real environment test with the social robot Tangy. We successfully implemented Scenario 1 using a plan produced by CP. The real environment test consisted of robot navigation, four bingo reminders and a bingo game activity. In Figure 2 we can see the social robot Tangy involved in a user reminder and autonomously navigating out of their personal room. These experiments, both simulated and real-world, are significant as they verify the utility and feasibility of our task planning methods in realistic environments, while considering real implementation situations.

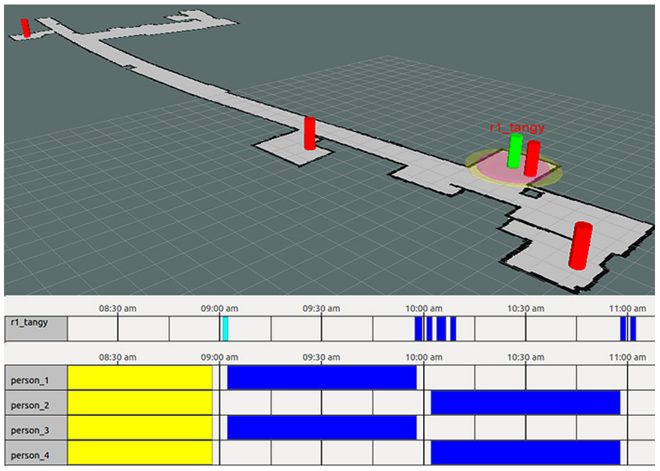

Fig. 1. ROS task planning simulation: a) Retirement home facility layout. b) Corresponding robot and user schedules. Blue bars represent tasks, cyan represents a currently active task, and yellow represents a mealtime.

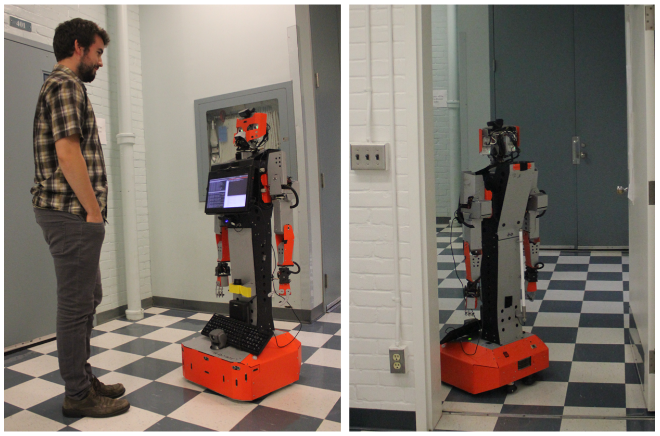

Fig. 2. Physical experiment, Scenario 1: a) Bingo game reminder in personal room. b) Robot autonomously navigating out of the personal room.

VI. CONCLUSION

We investigated modeling and solving two mobile robot task planning problems with existing optimization formalisms:

mixed-integer programming (MIP) and constraint programming (CP). The first problem requires a robot to plan a set of tasks, each with different temporal constraints. The second problem requires a robot to plan tasks while considering physical limitations, including battery level, as well as the timetables of multiple human users.

We compare the performance of these scheduling techniques to those in the literature. Constraint programming provides the best results by a substantial margin, indicating that the inference-based search of CP is superior to the relaxation-based MIP techniques for the problems studied. This will not always be the case; determining problem characteristics that favor one approach over the other is an active pursuit in the optimization literature (e.g., [33]). We also implemented the plans for the second problem in simulated and real environments on the social robot Tangy.

We believe that optimization-based techniques are a promising technology for mobile robot task planning problems. The flexibility of the "model-and-solve" paradigm eliminates the need for algorithmic development while exploiting the ongoing advances in these technologies. A primary direction for our future work is to understand the scale and complexity of robot task planning problems that these approaches can solve "off-the-shelf", without sophisticated techniques such as problem-specific decompositions (e.g., [16], [34]).

VII. Acknowledgment

We would like to thank M. Schwenk for the design of the simulation environment and S. Mohamed for the robot navigation and mapping modules utilized in the experiments.

References

[1] M. Ghallab, D. Nau, and P. Traverso, *Automated planning: theory & practice*. Elsevier, 2004.

[2] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, "An architecture for autonomy," *The International Journal of Robotics Research*, vol. 17, no. 4, pp. 315–337, 1998.

[3] M. Beetz and M. Bennewitz, "Planning, scheduling, and plan execution for autonomous robot office couriers," in *Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments, volume Workshop Notes*, pp. 98–02, 1998.

[4] B. Coltin, M. M. Veloso, and R. Ventura, "Dynamic user task scheduling for mobile robots.," in *Automated Action Planning for Autonomous Mobile Robots, AAAI Workshops*, vol. WS-11-09., AAAI, 2011.

[5] R. Castano, T. Estlin, R. C. Anderson, D. M. Gaines, A. Castano, B. Bornstein, C. Chouinard, and M. Judd, "Oasis: Onboard autonomous science investigation system for opportunistic rover science," *Journal of Field Robotics*, vol. 24, no. 5, pp. 379–397, 2007.

[6] B.-I. Kim, S. S. Heragu, R. J. Graves, and A. S. Onge, "A hybrid scheduling and control system architecture for warehouse management," *Robotics and Automation, IEEE Transactions on*, vol. 19, no. 6, pp. 991–1001, 2003.

[7] A. Cesta, G. Cortellessa, R. Rasconi, F. Pecora, M. Scopelliti, and L. Tiberio, "Monitoring elderly people with the robocare domestic environment: Interaction synthesis and user evaluation," *Computational Intelligence*, vol. 27, no. 1, pp. 60–82, 2011.

[8] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun, "Towards robotic assistants in nursing homes: Challenges and results," *Robotics and Autonomous Systems*, vol. 42, no. 3, pp. 271–281, 2003.

[9] A. Tapus, M. J. Mataric, and B. Scasselati, "Socially assistive robotics [grand challenges of robotics]," *Robotics & Automation Magazine, IEEE*, vol. 14, no. 1, pp. 35–42, 2007.

[10] L. Mudrova and N. Hawes, "Task scheduling for mobile robots using interval algebra," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 383–388, 2015.

[11] W.-Y. G. Louie, T. Vaquero, G. Nejat, and J. C. Beck, "An autonomous assistive robot for planning, scheduling and facilitating multi-user activities," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 5292–5298, 2014.

[12] M. L. Pinedo, *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media, 2012.

[13] S. Heinz, W.-Y. Ku, and J. C. Beck, "Recent improvements using constraint integer programming for resource allocation and scheduling," in *Proceedings of the Ninth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR2012)*, pp. 12–27, 2013.

[14] J. C. Beck, A. J. Davenport, E. D. Davis, and M. S. Fox, "The ODO project: Toward a unified basis for constraint-directed scheduling," *Journal of Scheduling*, vol. 1, no. 2, pp. 89–125, 1998.

[15] P. Baptiste, C. Le Pape, and W. Nuijten, *Constraint-based Scheduling*. Kluwer Academic Publishers, 2001.

[16] G. A. Korsah, B. Kannan, B. Browning, A. Stentz, and M. B. Dias, "xbots: An approach to generating and executing optimal multi-robot plans with cross-schedule dependencies," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 115–122, IEEE, 2012.

[17] N. Atay and O. B. Bayazit, "Emergent task allocation for mobile robots.," in *Robotics: Science and Systems*, Citeseer, 2007.

[18] P. Baptiste, C. Le Pape, and W. Nuijten, *Constraint-based scheduling: applying constraint programming to scheduling problems*, vol. 39. Springer Science & Business Media, 2012.

[19] W.-Y. Ku and J. C. Beck, "Revisiting off-the-shelf mixed integer programming and constraint programming models for job shop scheduling," tech. rep., Department of Mechanical & Industrial Engineering, University of Toronto, 2014. MIE-OR-TR2014-01.

[20] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica: Journal of the Econometric Society*, pp. 497–520, 1960.

[21] J. Błażewicz, W. Domschke, and E. Pesch, "The job shop scheduling problem: Conventional and new solution techniques," *European journal of operational research*, vol. 93, no. 1, pp. 1–33, 1996.

[22] R. E. Bixby, "A brief history of linear and mixed-integer programming computation," *Documenta Mathematica, Extra Volume: Optimization Stories*, pp. 107–121, 2012.

[23] P. Laborie, "IBM ILOG CP Optimizer for detailed scheduling illustrated on three problems," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pp. 148–162, Springer, 2009.

[24] C. Gervet, "Constraints over structured domains," in *The Handbook of Constraint Programming* (F. Rossi, P. Van Beek, and T. Walsh, eds.), pp. 605–638, Elsevier, 2006.

[25] W.-J. van Hoeve and I. Katriel, "Global constraints," in *Handbook of Constraint Programming* (F. Rossi, P. van Beek, and T. Walsh, eds.), ch. 6, pp. 169–208, Elsevier, 2006.

[26] J. Jaffar and M. J. Maher, "Constraint logic programming: A survey," *The journal of logic programming*, vol. 19, pp. 503–581, 1994.

[27] F. Rossi, P. Van Beek, and T. Walsh, *Handbook of constraint programming*. Elsevier, 2006.

[28] B. Coltin and M. Veloso, "Towards replanning for mobile service robots with shared information," in *Proceedings of ARMS Workshop, AAMAS*, vol. 11, 2013.

[29] J. Erschler, F. Roubellat, and J. P. Vernhes, "Finding some essential characteristics of the feasible solutions for a scheduling problem," *Operations Research*, vol. 24, pp. 772–782, 1976.

[30] M. Ghallab, C. Knoblock, D. Wilkins, A. Barrett, D. Christianson, M. Friedman, C. Kwok, K. Golden, S. Penberthy, D. E. Smith, *et al.*, "PDDL-the planning domain definition language," 1998.

[31] J. Benton, A. J. Coles, and A. Coles, "Temporal planning with preferences and time-dependent continuous costs.," in *ICAPS*, vol. 77, p. 78, 2012.

[32] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, 2009.

[33] S. Heinz and J. C. Beck, "Reconsidering mixed integer programming and MIP-based hybrids for scheduling," in *Proceedings of the Ninth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR2012)*, pp. 211–227, 2012.

[34] T. T. Tran, A. Araujo, and J. C. Beck, "Decomposition methods for the parallel machine scheduling problem with setups," *INFORMS Journal on Computing*, 2015. in press.