A Constraint Programming Approach to Multi-Robot Task Allocation and Scheduling in Retirement Homes

Kyle E. C. Booth, Goldie Nejat, and J. Christopher Beck

Department of Mechanical & Industrial Engineering
University of Toronto, Toronto, Ontario M5S 3G8, Canada
{kbooth, nejat, jcb}@mie.utoronto.ca

Abstract. We study the application of constraint programming (CP) to the planning and scheduling of multiple social robots interacting with residents in a retirement home. The robots autonomously organize and facilitate group and individual activities among residents. The application is a multi-robot task allocation and scheduling problem in which task plans must be determined that integrate with resident schedules. The problem involves reasoning about disjoint time windows, inter-schedule task dependencies, user and robot travel times, as well as robot energy levels. We propose mixed-integer programming (MIP) and CP approaches for this problem and investigate methods for improving our initial CP approach using symmetry breaking, variable ordering heuristics, and large neighbourhood search. We introduce a relaxed CP model for determining provable bounds on solution quality. Experiments indicate substantial superiority of the initial CP approach over MIP, and subsequent significant improvements in the CP approach through our manipulations. This work is one of the few, of which we are aware, that applies CP to multi-robot task allocation and scheduling problems. Our results demonstrate the promise of CP scheduling technology as a general optimization infrastructure for such problems.

1 Introduction

The progressive aging of populations, as observed primarily within developed countries, has important implications in a number of societal areas, including health and social care services for the elderly [1]. Such demographic trends have resulted in a dramatic increase in the number of seniors residing in retirement and nursing homes [2]. This increase in demand for care services, combined with a reduction in the working age population, will inevitably result in greater pressures on the quality of elderly care infrastructure, risking deterioration in the provision of medical services, daily assistance, social interaction, and overall quality of life for residents. Due to these demographic and industry dynamics, the investigation of the role of autonomous robotics within healthcare has been discussed for a number of decades, though primarily with respect to robots assisting physical rehabilitation. The design and deployment of socially assistive

robots for retirement home applications and elderly care is a more recent development [3]. Such social robots alleviate workforce pressures associated with the daily operation of retirement homes and work to give assistance through the autonomous facilitation of cognitively and socially stimulating leisure activities.

In this paper, we contribute a novel application of constraint programming (CP) to the automated planning and scheduling of a team of social robots in a retirement home. Our larger project involves the robots providing social and cognitive stimulation through the facilitation of bingo games involving multiple residents and telepresence sessions between residents and their family members. Here, we propose CP as part of a task planning system that must autonomously allocate, schedule, and facilitate these single and multi-resident leisure activities throughout the course of the day, while adhering to daily resident calendars defining their availability. The problem involves reasoning about which tasks should be implemented (i.e. *planning*), as well as which robot should facilitate each task and at what time (i.e. *scheduling*).

In the field of robotics, multi-robot task allocation (MRTA) aims to solve robot coordination problems pertaining to task decomposition from high-level goals, task distribution, and task scheduling. We extend the previously proposed single robot version of the retirement home problem [4, 5], to an MRTA problem. We investigate mixed-integer programming (MIP) and CP as allocation and scheduling strategies. These approaches model disjoint time windows, robot and user travel times within the retirement home, inter-schedule task dependencies, and robot energy consumption/replenishment. We investigate enhancements of our initial CP approach through grouped variable ordering heuristics and large neighbourhood search (LNS), and present a relaxed CP formulation used for determining provable bounds on solution quality. Numerical results indicate substantial superiority of the CP formulation over MIP, and we show significant improvements of the CP approach through our manipulations of the search. This experimentation illustrates CP scheduling technology as a promising general optimization framework for MRTA problems.

2 Related Work

CP has been applied to a wide range of combinatorial optimization problems, excelling most notably in scheduling applications [6], where it has established itself as a strong competitor to mathematical programming-based approaches, often out-performing state-of-the-art MIP solvers [7]. The flexible nature of CP, combined with its proficiency at representing and solving particular combinatorial substructure (e.g. problems with task sequencing) has led to its integration with other methods, producing stronger hybrid approaches. Examples of this integration include logic-based Benders decomposition (LBBD) [8] and constraint-integer programming (CIP) [9]. CP has also been used in combination with Local Search (LS) in the Large Neighbourhood Search (LNS) [10] framework. Indeed, commercial CP software has benefited tremendously from this integration as

seen within the incorporation of self-adapting LNS in state-of-the-art constraint solvers [11].

Initial approaches to MRTA problems used dispatch-style methods where a single task was allocated and executed before the next allocation was made [12, 13]. More recent approaches utilize decentralized methods such as market-based strategies [14], auction-based approaches [15], and distributed local task-swapping [16]. In the past decade, efforts have been made to use linear and integer programming techniques [5, 17], largely due to attractive bounds on solution quality. CP has been proposed as a suitable candidate approach for these problems [18, 19], however, the application of CP to multi-robot task planning and scheduling is, to the best of our knowledge, limited in the literature. The MACBETH [20] architecture makes use of a combination of hierarchical task networks and CP, where a human user specifies missions to a team of autonomous agents via a playbook graphic user interface. Another proposed method uses distributed constraint satisfaction problems (disCSP) to solve multi-robot exploration problems [21].

Socially assistive robots for elderly care have seen growing attention within the literature [22]. For the retirement home application studied in this paper, existing related work has presented temporal planning, MIP, and CP approaches for solving the single-robot task planning variant of the problem [4,5], where CP was demonstrated to outperform the other techniques. Multiple robot scenarios have also been recently studied [23], where a planning and scheduling architecture was introduced using off-the-shelf temporal planners for a specialization of the problem studied in this paper.

3 Problem Definition

Given a set of robots, R, a set of possibly optional tasks, T, and a problem-specific cost function, our MRTA problem involves determining a mapping of tasks to robots, $f : T \rightarrow R$, as well as an assignment of start times to tasks, such that the objective is optimized. In this section we discuss specific problem parameters and objectives associated with our retirement home application.

3.1 Parameters

We consider a set of users (retirement home residents), $U := \{u_1, u_2, ..., u_n\}$, where each user, $u_i \in U$, has a unique daily calendar, $\Sigma_i := \{\sigma_{i1}, \sigma_{i2}, ..., \sigma_{i5}\}$, identifying five busy periods where the user is not available for interaction. For modeling purposes, we treat each busy period as a required task defined on a closed interval with a fixed start and end time. These intervals include one hour breaks for breakfast (8:00-9:00 AM), lunch (12:00-1:00 PM), and dinner (5:00-6:00 PM), as well as two additional breaks with duration ranging from 30 minutes to one hour. An estimate of user movement speed, ν_u in metres/minute, used to approximate travel times between locations.

We also consider a set of robots, $R := \{r_1, r_2, ..., r_m\}$, that facilitate human-robot interaction (HRI) tasks within the retirement home. Each robot, $r_k \in R$, starts and ends in the *robot depot*, a location containing a recharging station. Robot movement speed, ν_r in metres/minute, is known, as well as lower and upper limits on battery level, β_{min} and β_{max}, respectively. Robot energy consumption rates are defined for robot movement, ξ_Δ, and consumption (or replenishment in the case of recharge tasks) for task j as ξ_j.

The retirement home contains a set of locations, $L := \{\ell_r, \ell_g, \ell_\sigma\} \cup \{\ell_1, \ell_2, ... , \ell_n\}$, representing the robot depot, games room, meal/break room, and a personal room for each of the users. Distances between any pair of locations ℓ_a and ℓ_b are known, and defined as $\delta_{(a,b)}$, in meters. Travel time matrices are generated for users, $\Delta^u := \{\frac{\delta_{(a,b)}}{\nu_u} : (a, b) \in L \times L\}$, and for robots, $\Delta^r := \{\frac{\delta_{(a,b)}}{\nu_r} : (a, b) \in L \times L\}$, where travel times are estimated in minutes.

The problem considers individual and group HRI tasks, each requiring a single robot facilitator. These task types are: *telepresence* tasks (individual), *bingo games* (group), and *bingo game reminders* (individual). The set of telepresence tasks is defined as $P := \{p_1, p_2, ..., p_n\}$; these tasks take place in the personal room of the user and have a duration of 30 minutes. The set of bingo game tasks, $G := \{g_1, g_2, ..., g_{UB_1}\}$, is defined based on a calculated upper bound, UB_1, as the number of games that can be facilitated is unknown a priori. These tasks take place in the games room and have a duration of 60 minutes. The necessity for this upper bound illustrates an important limitation when using scheduling methods for problems with underlying planning characteristics: a predefined number of tasks is required. We define the set of available reminder tasks as $\mathcal{M} := \bigcup_{i=1}^{n} M_i$ where the subset of reminder tasks for each user, $u_i \in U$, is defined as $M_i := \{m_{i1}, m_{i2}, ..., m_{iUB_1}\}$. Each reminder takes place in the personal room of the participating user and has a duration of two minutes. The duration of telepresence, bingo game, and reminder tasks are represented as d_j for task j. We also define a set of available *recharge* tasks, $\mathcal{C} := \bigcup_{k=1}^{m} C_k$, where the subset of these tasks for each robot, $r_k \in R$, is defined as $C_k := \{c_{k1}, c_{k2}, ...c_{kUB_2}\}$. The number of these tasks available for each robot is defined based on the calculated upper bound, UB_2, as the number of recharge tasks a robot may require is also unknown a priori. The duration (in minutes) of each recharge task varies within the closed interval $[0, \frac{\beta_{max}-\beta_{min}}{\xi_j}]$ for $j \in C_k,\ \forall r_k \in R$.

We define the set of all tasks potentially involving each user, $u_i \in U$, as $T_i^u := \{\Sigma_i \cup p_i \cup G \cup M_i\}$, and mandatory start and end dummy tasks with zero duration for users as $\dot{u}_i$ and $\ddot{u}_i$, respectively. These tasks facilitate user task sequencing and have zero transition time to all other task locations. We define the set of all tasks potentially involving each robot, $r_k \in R$, as $T_k^r := \{P \cup G \cup \mathcal{M} \cup C_k\}$, with mandatory start and end dummy tasks with zero duration as $\dot{r}$ and $\ddot{r}$, respectively. These tasks are located at the robot depot, and have associated spatial transition times to other tasks, ultimately ensuring that robot schedules start and end at the robot depot.

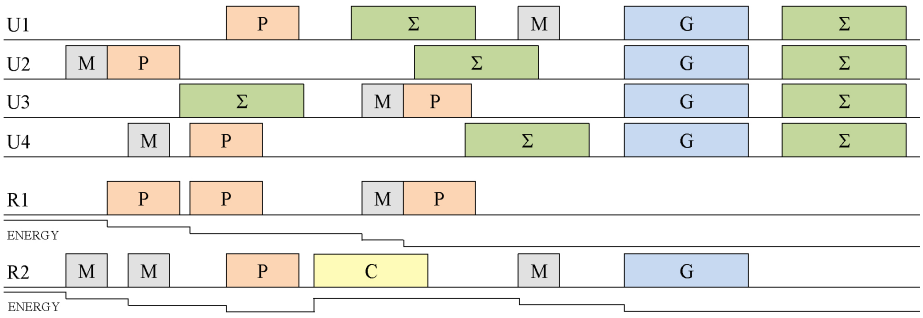

Fig. 1. Time-extended MRTA for a retirement home: Feasible task allocations and schedules. Instance size: $|U| = 4, |R| = 2$.

3.2 Objective

Given a single day (7:00 AM-7:00 PM) planning horizon, H, a time-extended allocation of tasks to robots must be determined that integrate with user schedules. The battery level of each robot, $r_k \in R$, is known throughout the day, and must stay within the specified closed interval, $[\beta_{min}, \beta_{max}]$. Each user must participate in exactly one telepresence activity but user participation in bingo games is optional. If a user participates in a bingo game activity, the associated reminder task must be done before the game. The optimization objective of the problem is to maximize bingo game participation over all users. Solutions with equivalent bingo game participation are prioritized by favoring schedules with fewer robot recharge tasks.

A feasible solution to a small problem instance is illustrated in Figure 1. Telepresence (orange), bingo game (blue), reminder (grey), and recharge (yellow) tasks are all represented, as well as user busy periods (green). We note that unless tasks occur in the same location (e.g. reminder and telepresence in a user personal room), the tasks are never scheduled immediately next to each other. This is due to the modeling of travel times for both robot and users.

Though in this particular problem definition we are generating a single time-extended plan, complete solution methods for this problem will need to incorporate replanning due to likely discrepancies during schedule execution (e.g. a missed telepresence). As such, we look to generate high-quality allocations within reasonably short time-frames ($\leq$ 5 minutes).

4 Task Allocation and Scheduling Models

In this section, we present task allocation and scheduling formulations using CP and MIP. We formally define both models and discuss key modeling considerations made.

4.1 Constraint Programming Model

We present a CP model in Figure 2. For this model, we utilize cumulative variables and *optional interval variables*, which are decision variables whose possible values are a convex interval: $\{\perp\} \cup \{[s, e) | s, e \in \mathbb{Z}, s \leq e\}$, where s and e are the start and end values of the interval and $\perp$ is a special value indicating the variable is not present in the solution [24]. The variable $\texttt{Pres}(var)$ is 1 if interval variable var is present in the solution, and 0 otherwise. Model constraints are only enforced on such interval variables that are present in the solution. $\texttt{Start}(var)$, $\texttt{End}(var)$, and $\texttt{Length}(var)$ return the integer start time, end time, and length, respectively, of the interval variable var.

We define the decision variables used in the CP formulation as follows:

$x_{ij} :=$ (*interval*) present if user u_i attends task j and absent otherwise,
$y_{kj} :=$ (*interval*) present if robot r_k facilitates task j and absent otherwise,
$E_k :=$ (*cumulative*) energy level of robot r_k throughout the schedule.

$$max \quad \sum_{u_i \in U} \sum_{j \in G} \texttt{Pres}(x_{ij}) - 0.01 \cdot \sum_{r_k \in R} \sum_{j \in C_k} \texttt{Pres}(y_{kj}) \tag{1}$$

$$s.t. \quad \texttt{NoOverlap}([x_{i\dot{u}}, x_{i1}, x_{i2}, ..., x_{i|T_i^u|}, x_{i\ddot{u}}], \Delta^u), \qquad \forall u_i \in U \tag{2}$$

$$\texttt{NoOverlap}([y_{k\dot{r}}, y_{k1}, y_{k2}, ..., y_{k|T_k^r|}, y_{k\ddot{r}}], \Delta^r), \qquad \forall r_k \in R \tag{3}$$

$$\texttt{Pres}(x_{ip_i}) = \sum\nolimits_{r_k \in R} \texttt{Pres}(y_{kp_i}) = 1, \qquad \forall u_i \in U \tag{4}$$

$$\texttt{Pres}(x_{ij}) \leq \sum\nolimits_{r_k \in R} \texttt{Pres}(y_{kj}) \leq 1, \qquad \forall u_i \in U; j \in G \tag{5}$$

$$\texttt{End}(x_{im_{ij}}) \leq \texttt{Start}(x_{ij}), \qquad \forall u_i \in U; j \in G \tag{6}$$

$$\sum\nolimits_{r_k \in R} \texttt{Pres}(y_{km_{ij}}) = \texttt{Pres}(x_{ij}), \qquad \forall u_i \in U; j \in G \tag{7}$$

$$\texttt{StartAtStart}(x_{ij}, y_{kj}, 0), \qquad \forall u_i \in U; r_k \in R; j \in T_i^u \cap T_k^r \tag{8}$$

$$\theta_{kj} = \texttt{Length}(y_{kj}) \cdot \xi_j + \Delta^r_{(pre_j, j)} \cdot \xi_\Delta, \qquad r_k \in R; j \in T_k^r \cup \{\ddot{r}\} \tag{9}$$

$$E_k = \sum\nolimits_{j \in T_k^r \cup \{\ddot{r}\}} \texttt{StepAtStart}(j, -\theta_{kj}), \qquad \forall r_k \in R \tag{10}$$

$$\beta_{min} \leq E_k \leq \beta_{max}, \qquad \forall r_k \in R \tag{11}$$

$$\texttt{Pres}(x_{ij}) = 1, \texttt{Start}(x_{ij}) = \sigma_{ij}, \qquad \forall u_i \in U; j \in \Sigma_i \tag{12}$$

$$\texttt{Pres}(x_{ij}) \in \{0, 1\}, \texttt{Start}(x_{ij}) \in [0, H], \qquad \forall u_i \in U; j \in T_i^u \setminus \Sigma_i \tag{13}$$

$$\texttt{Pres}(y_{jk}) \in \{0, 1\}, \texttt{Start}(y_{jk}) \in [0, H] \qquad \forall r_k \in R; j \in T_k^r \tag{14}$$

Fig. 2. Constraint programming model

Objective (1) maximizes the total number of bingo games played across all users, prioritizing solutions with fewer recharge tasks in the event of equivalent group activity participation. Constraints (2) and (3) encapsulate the sequencing requirement for all potential tasks associated with each user and each robot, including required dummy tasks. We utilize the `NoOverlap` global constraint

which performs efficient domain filtering on the interval variable start times by reasoning about task time windows, processing times, and the relationship that no pair of tasks can overlap in time, with consideration for transition times [6]. In our model, the `NoOverlap` constraints enforced on users differ from those enforced on robots due to the different tasks within the sets T_i^u and T_k^r. Since both users and robots move within the retirement home throughout the day, these constraints ensure that their final task plans are properly sequenced and allow for transition times through the inclusion of the Δ^u and Δ^r transition time matrices.

Constraint (4) ensures that, for each user, exactly one robot facilitates the required telepresence task. Constraint (5) links the user participation bingo game variables to the robot facilitation variables; the decision for user $u_i \in U$ to attend bingo game $j \in G$ is bounded by the presence of a robot $r_k \in R$ facilitating that game. This constraint also restricts each bingo game to be facilitated by at most one robot. Constraint (6) enforces the precedence relationship between user bingo game participation and reminder tasks. Constraint (7) ensures that if a user is attending a bingo game, he/she must receive exactly one reminder for that particular bingo game, and zero otherwise. Constraint (12) identifies user calendar tasks as mandatory.

Constraint (8) ensures that the start times of intersecting tasks between user and robot schedules are synchronized. We use the `StartAtStart` constraint which ensures that, whenever both variables are present, the distance between their start times, $\texttt{Start}(x_{ij}) - \texttt{Start}(y_{kj}) = 0$. Constraints (9)-(11) represent the energy-related constraints for each robot. Collectively, these constraints ensure that the cumulative function energy level variable, e_k, stays within specified bounds while using the `StepAtStart` global constraint to model the various energy consumptions of the tasks. The term θ_{kj} in Constraint (9) represents the energy consumption of facilitating task j combined with the energy consumed in travelling to the location of the task from the previous location in the sequence. Constraints (13) and (14) identify the optionality of the interval variables, as well as start time domains.

4.2 Mixed-Integer Programming Model

For comparison purposes, we also present a MIP model for the problem as defined in Figure 3. The model is based on the formulation for the electric vehicle-routing problem with time windows (E-VRPTW) [25], treating each of the sets T_i^u and T_k^r as completely-connected graphs with edge-weights representing travel times between locations. We make extensive use of Miller-Tucker-Zemlin (MTZ) [26] sequencing constraints to model task start times and robot energy levels throughout the planning horizon. We extend the E-VRPTW by including task synchronization constraints, a concept that has been applied within the context of vehicle routing previously [27], as well as problem-specific inter-schedule task dependencies (i.e. reminders delivered before bingo games).

We define the decision variables used in the MIP formulation as follows:

x_{ij} := (*binary*) 1 if user u_i attends task j and 0 otherwise,
y_{kj} := (*binary*) 1 if robot r_k facilitates task j and 0 otherwise,
α_{ijl} := (*binary*) 1 if task j directly precedes task l for user u_i and 0 otherwise,
γ_{kjl} := (*binary*) 1 if task j directly precedes task l for robot r_k and 0 otherwise,
ϕ_{ij} := (*integer*) start time of task j in the schedule of user u_i,
ψ_{kj} := (*integer*) start time of task j in the schedule of robot r_k,
D_{kj} := (*integer*) length of task j in the schedule of robot r_k,
ϵ_{kj} := (*integer*) energy level of robot r_k after completing task j.

$$max \quad \sum_{u_i \in U} \sum_{j \in G} x_{ij} - 0.01 \cdot \sum_{r_k \in R} \sum_{j \in C_k} y_{kj} \tag{15}$$

$$s.t. \quad \sum\nolimits_{l \in T_i^u \cup \{\ddot{u}\}, l \neq j} \alpha_{ijl} = x_{ij}, \qquad \forall u_i \in U; j \in T_i^u \cup \{\dot{u}\} \tag{16}$$

$$\sum\nolimits_{j \in T_i^u \cup \{\dot{u}\}, j \neq l} \alpha_{ijl} = x_{il}, \qquad \forall u_i \in U; l \in T_i^u \cup \{\ddot{u}\} \tag{17}$$

$$\sum\nolimits_{l \in T_k^r \cup \{\ddot{r}\}, l \neq j} \gamma_{kjl} = y_{kj}, \qquad \forall r_k \in R; j \in T_k^r \cup \{\dot{r}\} \tag{18}$$

$$\sum\nolimits_{j \in T_k^r \cup \{\dot{r}\}, j \neq l} \gamma_{kjl} = y_{kl}, \qquad \forall r_k \in R; l \in T_k^r \cup \{\ddot{r}\} \tag{19}$$

$$\phi_{ij} + d_j + \Delta^u_{(j,l)} \leq \phi_{il} + H \cdot (1 - \alpha_{ijl}), \qquad \forall u_i \in U; j, l \in T_i^u, j \neq l, \tag{20}$$

$$\psi_{kj} + D_{kj} + \Delta^r_{(j,l)} \leq \psi_{kl} + H \cdot (1 - \gamma_{kjl}), \qquad \forall r_k \in R; j, l \in T_k^r, j \neq l, \tag{21}$$

$$\epsilon_{kl} + D_{kj} \cdot \xi_j + \Delta^r_{(j,l)} \cdot \xi_\Delta \leq \epsilon_{kj} + \beta_{max} \cdot (1 - \gamma_{kjl}), \qquad \forall r_k \in R; j, l \in T_k^r, j \neq l \tag{22}$$

$$\phi_{ij} = \psi_{kj}, \qquad \forall u_i \in U; r_k \in R; j \in T_i^u \cap T_k^r, \tag{23}$$

$$x_{ip_i} = \sum\nolimits_{r_k \in R} y_{kp_i} = 1, \qquad \forall u_i \in U \tag{24}$$

$$x_{ij} \leq \sum\nolimits_{r_k \in R} y_{kj} \leq 1, \qquad \forall u_i \in U; j \in G \tag{25}$$

$$\sum\nolimits_{r_k \in R} y_{km_{ij}} = x_{ij}, \qquad \forall u_i \in U; j \in G \tag{26}$$

$$D_{kj} = d_j, \qquad \forall r_k \in R; j \in T_k^r \setminus C_k \tag{27}$$

$$0 \leq D_{kj} \leq \tfrac{\beta_{max} - \beta_{min}}{\xi_j}, \qquad \forall r_k \in R; j \in C_k \tag{28}$$

$$\beta_{min} \leq \epsilon_{kj} \leq \beta_{max}, \qquad \forall r_k \in R; j \in T_k^r \tag{29}$$

$$x_{ij} = 1, \phi_{ij} = \sigma_{ij}, \qquad \forall u_i \in U; j \in \Sigma_i \tag{30}$$

$$x_{ij}, \alpha_{ijl} \in \{0, 1\},\ \phi_{ij} \in [0, H], \qquad \forall u_i \in U; j, l \in T_i^u \setminus \Sigma_i \tag{31}$$

$$y_{kj}, \gamma_{kjl} \in \{0, 1\},\ \psi_{kj} \in [0, H] \qquad \forall r_k \in R; j \in T_k^r \tag{32}$$

Fig. 3. Mixed-integer programming model

Objective (15) is functionally equivalent to the objective of the CP model. Constraints (16) and (17) represent the node degree constraints for user tasks, and Constraints (18) and (19) the node degree constraints for robot tasks. Constraints (20) and (21) are MTZ sequencing constraints used to determine valid start times of user and robot tasks while adhering to task duration and transi-

tion times. Constraint (22) uses MTZ sequencing to model robot energy level, where energy is consumed or replenished depending on the task sequencing.

Constraint (23) synchronizes the start times of intersecting tasks between user and robot schedules, necessary for the integration of these task plans. Constraint (24) ensures that each user participates in exactly one telepresence task, and that each of these tasks is facilitated by exactly one robot. Constraint (25) constrains bingo game facilitation to at most one robot, and ensures user participation is bounded by this value. Constraint (26) ensures that if a user participates in a bingo game, he/she receives exactly one reminder facilitated by a single robot.

Constraint (27) defines the length of robot tasks not including recharge tasks to be constant and Constraint (28) identifies the bounds on variable-length recharge tasks. Constraint (29) defines the acceptable bounds on robot battery level and the remainder of the model, Eqns. (30)-(32), dictates the domains of the decision variables as fixed, binary, or positive integer.

4.3 Modeling Considerations

The schedules produced for the users and robots must be temporally synchronized, must accurately model robot and user travel times, and must ensure adherence to the energy capacity of the robots. This section identifies key considerations made when modeling these complex relationships.

Schedule Synchronization Task synchronization between user and robot schedules is a primary concern in this problem. While proposed methods for the single-robot retirement home application [5, 28] have accounted for transition times between robot tasks, they have assumed users travel between locations instantly. As a result, if a candidate start time for a robot-facilitated task did not conflict with the availability calendar of that user, the start time was considered valid (assuming task duration and end time did not pose conflict).

When user movement within the environment is relaxed, straightforward modeling within CP would utilize the `ForbidExtent`(var, f) global constraint, which prevents an interval variable var from overlapping a time point t where $f(t)$, an integer step function, is equal to 0 [24]. This constraint represents a natural way to model relationships involving disjoint resource time windows or calendars, supplementing user-sequencing Constraint (2) within our formulation. This method of modeling is inaccurate as it does not represent the travel times of users to and from the locations of subsequent tasks nor break periods. To remedy this, we include `NoOverlap` global constraints for both users and robots.

Properly accounting for both user and robot movement within the environment brings further modeling challenges pertaining to schedule synchronization. Since user availability is now dependent on spatial transition times in addition to their calendars, the temporal synchronization of a task involving both a user and robot on their respective schedules is necessary. This requirement is achieved within the CP and MIP models using Constraints (8) and (23), respectively. The

modeling presented is then further strengthened by noting that tasks involving a one-to-one mapping of robots to users (e.g. telepresence and reminder tasks) can be simultaneously represented on both user and robot schedules. Tasks involving a one-to-many mapping of robots to users (e.g. bingo game activities) are linked with the aforementioned synchronization constraints.

Symmetry Breaking The problem has a number of inherent symmetries due to the homogenous nature of resources and tasks. We investigate a number of symmetry breaking options in efforts to reduce the search. These constraints are formulated in CP, though similar MIP constraints can be expressed with binary variables. For a given robot, each of the recharging tasks available to it are identical. Our models, as formulated, treat a single recharge solution using recharge task, c_{kj}, as functionally different than a solution using recharge task, c_{kl}, even if the start times and durations of each are the same. These symmetries can be broken using the following CP constraint to order the use of recharge tasks:

$$\texttt{Pres}(y_{kj+1}) \leq \texttt{Pres}(y_{kj}), \forall r_k \in R, j \in C_k \setminus \{c_{kUB_2}\} \tag{33}$$

We can also consider breaking symmetries pertaining to bingo game tasks, as these tasks are also homogeneous. We break these symmetries by enforcing lexicographic ordering within robot facilitation decisions and user participation. We enforce an ordering on bingo game tasks with the following constraints:

$$\textstyle\sum_{r_k \in R} \texttt{Pres}(y_{kj+1}) \leq \sum_{r_k \in R} \texttt{Pres}(y_{kj}), \forall j \in G \setminus \{g_{UB_1}\} \tag{34}$$

$$\textstyle\sum_{u_i \in U} \texttt{Pres}(x_{ij+1}) \leq \sum_{u_i \in U} \texttt{Pres}(x_{ij}), \forall j \in G \setminus \{g_{UB_1}\} \tag{35}$$

Since bingo game tasks are linked among users and robots, symmetry breaking can only be expressed over the sum of such variables. As previously noted [29], symmetry breaking can be counterproductive and delay the discovery of feasible solutions. We investigate these constraints experimentally in Section 6.

5 CP Search Manipulations

As presented in Section 6, the initial MIP model exhibits very poor performance when compared to the initial CP approach. As such, we pursue CP as the more suitable technology for the given application. In this section we discuss methods for increasing the performance of the initial CP formulation, using grouped variable orderings heuristics and large neighbourhood search.

5.1 Grouped Variable Ordering Heuristics

One of the key focuses of this work is to determine if CP can be used to generate time-extended allocations within realistic timeframes ($\leq$ 5 minutes). In order to increase the performance of the CP formulation, we conduct a detailed investigation of *grouped variable ordering heuristics*, specific instantiation orderings of

groups of variables, to uncover elements of problem structure and help reduce the search space.

We define groups of variables and then instantiate each group according to a specified order within the search. Variable groups that appear earlier in the ordering have all of their elements instantiated before subsequent variables are considered. For the purposes of our investigation, we consider groups of variables associated with robot bingo game facilitation, bingo user participation, user telepresence participation, reminder participation, and robot recharge tasks. We implement instantiation orderings over variable groups defined as: $\mathcal{V} := \{\{y_{kg_1}, ..., y_{kg_{UB_1}}\}, \{x_{ig_1}, ..., x_{ig_{UB_1}}\}, \{x_{ip_1}, ..., x_{ip_n}\}, \{x_{im_{ig_1}}, ..., x_{im_{ig_{UB_1}}}\}, \{y_{kc_{k1}}, ..., y_{kc_{kUB_2}}\}\}$ for all robots $r_k \in R$ and users $u_i \in U$. Within these variable groups we investigate orderings on all possible subsets of $\mathcal{V}$ of size one and two (single and double stage). Problem variables not included in the selected subset will be instantiated after those selected. By inspecting Figure 1, it is clear that instantiating the set of $\{y_{kg_1}, ..., y_{kg_{UB_1}}\}$ variables for all $r_k \in R$, as detailed in Constraint (5), will have high impact on other variables and thus may be promising candidates for early instantiation decisions.

For the first set of experiments, we consider subsets of $\mathcal{V}$ of size one resulting in five orderings. The remainder of the variables are then instantiated using the default solver strategy. The second set of experiments uses a double-stage search phase with subsets of size two. We explore all two-stage permutations of decision variable groups in $\mathcal{V}$, yielding 20 unique group orderings. With this two-stage assessment we hope to uncover findings pertaining to problem structure that may not be apparent upon initial inspection.

5.2 Large Neighbourhood Search

Large neighbourhood search (LNS) [10] is a method that combines local search (LS) with constraint programming (CP). It has proven to be effective for solving large, complex optimization problems [30]. We implement LNS in order to further improve solution quality on larger instances of our problem, using a variation of the *time window neighbourhood* selection heuristic [30]. Other selection heuristics that exploit problem-specific structure did not perform as well.

As initial solutions to the global problem are often of low quality, we allot one minute of run-time to a CP search using our best grouped variable ordering heuristic to find an incumbent solution. This initial search helps ensure that the LNS procedure begins with a high-quality solution, ultimately improving the performance of the method. Next, with variable set N, we unassign all variables that are: i) present, and ii) have start times within the current time window (initially this window is defined on the closed interval [7:00AM, 10:00AM]). We fix the remainder of the solution variable start times; the unassigned set is S, and the fixed set $r := N \setminus S$. We solve the resultant problem to try to quickly find improving solutions, if they exist. The time limit used here is set to 20 seconds, and a backtrack limit is enforced to prevent fruitless exploration. If the solution is improved, we reset the time window to its initial interval, replace the incumbent

solution and repeat. In the event the solution does not improve, we shift our time window to the right (i.e. later in time) by one hour and repeat the process. If all time windows, of the current size, are explored without improvement (with final window [4:00PM, 7:00PM]), we reset the time window to its initial interval, increase its size by one hour, and repeat. This effectively defines a neighbourhood selection heuristic that increases in size over time.

6 Experimental Results and Analysis

In this section we present a systematic experimental analysis of our initial models as well as the search manipulation results for our CP approach. All experiments are implemented in C++ on a hexacore machine with a Xeon processor and 16GB of RAM running Mac OS X Yosemite. We use CP Optimizer (for CP) and CPLEX (for MIP) from the IBM ILOG CPLEX Optimization Studio version 12.6.2 single-threaded for all simulations with default search and inference settings unless otherwise noted.

Problem Instances We consider five instances sizes defined based on the number of robots, $|R|$, and users, $|U|$. These sizes are: 2×5, 2×10, 3×15, 3×20, 4×25. These sizes are selected to reflect real-world retirement home problems. For each problem size we produce 10 unique instances, resulting in an instance set of 50 problems. Transition matrices, Δ, for each instance are based on randomly generated travel distances between locations within the facility, satisfying the triangle inequality ($\delta_{(a,b)} + \delta_{(b,c)} \geq \delta_{(a,c)}$). Two mandatory break periods (in addition to mealtimes) are randomly inserted into user calendars, each with a duration ranging from 30 minutes to one hour and a randomly assigned start time. We use $|G| = UB_1 = 5$ available bingo game tasks and $|C_k| = UB_2 = 3, \forall r_k \in R$, available recharge tasks. We use a 5 minute run-time limit for all experiments, unless otherwise noted.

Initial CP and MIP Performance We ran experiments on the problem scenarios using the initial CP and MIP formulations presented in Section 4 with default solver settings. CP is able to find feasible solutions for 9/50 problem instances, as seen in Table 1, while the MIP formulation is not able to find any feasible solutions. The numerous MTZ sequencing constraints have a poor linear relaxation strength, largely due to the inclusion of large integer values (i.e., H and β_{max}) for disjunctive reasoning. We believe the poor MIP performance is due to the extensive reliance on these constraints; future work will involve looking at using generalized subtour elimination [31] to improve algorithm performance. These experiments strongly suggest that CP is more suitable, of the two methods, for solving this problem. This finding is in agreement with the conclusions of previous research on a similar single-robot variant of the problem [5].

Bounds on Solution Quality In order to measure the performance of our methods, we make efforts to determine provable and non-trivial upper bounds on solution quality. Valid bounds can be determined using the best dual bound from MIP, however, for these problems MIP is unable to produce non-trivial bounds within run-times of one hour, even when the problem is relaxed, multi-threading is permitted, and the search emphasis is set to focus on dual bound strengthening.

Instead, we use a relaxation of our initial CP formulation, noting that the relaxation is only a true bound if the solution is proven optimal. The relaxed formulation is as follows:

$$\left\{ max \sum_{u_i \in U} \sum_{j \in G} \texttt{Pres}(x_{ij}) \ : \ Constraints\ (2){-}(8), (12){-}(14), (34){-}(35) \right\} \quad (36)$$

This formulation relaxes the energy component of the problem in both the objective and constraints, while the remainder of the constraints are enforced. The set of tasks available to a robot becomes $\bar{T}_k^r := T_k^r \setminus C_k$. Solver inference is adjusted to the highest level (*extended*) in order to increase the amount of propagation performed at each search node, and symmetry breaking constraints are included. With these considerations made, the above model is only able to solve the first ten instances (instance sizes 2×5 and 2×10) to proven optimality within a run-time limit of one hour. The remainder of the instances yield unproven upper bound estimations.

Numerical Results We present the performance of the various CP-based approaches we have investigated in Table 1. $CP_{default}$ is the original formulation with default solver settings, $CP_{default+SB}$ adds the symmetry breaking constraints, CP_{SP_1} represents the best performing single-stage variable instantiation strategy, and $CP_{SP_{1\rightarrow 2}}$ the best performing double-stage instantiation strategy. To implement grouped variable ordering heuristics within IBM ILOG CP Optimizer, we use *search phases*. We conduct 250 experiments (50×5) and 1,000 experiments (50×20), respectively, for the single and double-stage orderings, in efforts to deduce the best variable instantiation strategy. $CP + LNS$ represents the performance of our CP-based time-window LNS approach.

We use mean relative error (MRE) to measure performance, calculated as follows:

$$MRE_{(\Omega,0.1)} = \sum_{f \in F} \frac{c^*(f) - c(\Omega, 0.1)}{|F| \times c^*(f)} \times 100 \quad (37)$$

where the MRE for a particular method Ω at 0.1 seconds, for example, is calculated as above. The solution $c^*(f)$ represents the upper bound (or upper bound approximation) obtained by solving the relaxed CP formulation as presented in Formulation (36) for one hour. Problem instances $f \in F$ represent problems for which feasible solutions are found in the 5 minute run-time limit.

Table 1. CP Approach Results: Mean relative error (%) over time. '†' indicates approximate bound, $c^*(f)$, used for calculation. A value of '-' indicates the approach failed to find a feasible solution for all ten instances at that run-time. '# Inf.' represents the number of instances for which no feasible plan was found after 300 seconds of run-time.

		Run-time (s)						
$\|R\| \times \|U\|$	Approach	0.1	1	5	10	100	300	# Inf.
2×5	$CP_{default}$	91.2	73.1	36.5	35.4	22.3	20.1	2
	$CP_{default+SB}$	97.2	91.5	65.6	55.9	29.9	27.0	2
	CP_{SP_1}	**74.6**	50.7	18.4	15.1	**0.1**	**0.1**	**0**
	$CP_{SP_{1\rightarrow 2}}$	89.7	**48.0**	**17.1**	**15.0**	1.3	**0.1**	**0**
	$CP + LNS$	**74.6**	50.7	18.4	15.1	**0.1**	**0.1**	**0**
2×10	$CP_{default}$	-	-	-	-	93.5	91.5	9
	$CP_{default+SB}$	-	-	-	-	-	-	10
	CP_{SP_1}	-	97.8	51.8	45.8	**20.6**	12.8	**0**
	$CP_{SP_{1\rightarrow 2}}$	-	**89.9**	54.0	49.0	22.6	19.8	**0**
	$CP + LNS$	-	97.8	**51.5**	**45.1**	22.0	**2.8**	**0**
$3 \times 15^{\dagger}$	$CP_{default}$	-	-	-	-	-	-	10
	$CP_{default+SB}$	-	-	-	-	-	-	10
	CP_{SP_1}	-	99.6	83.2	53.7	32.9	21.7	**0**
	$CP_{SP_{1\rightarrow 2}}$	-	**99.5**	**65.1**	**44.8**	29.4	25.9	**0**
	$CP + LNS$	-	**99.5**	82.5	53.3	**29.0**	**13.7**	**0**
$3 \times 20^{\dagger}$	$CP_{default}$	-	-	-	-	-	-	10
	$CP_{default+SB}$	-	-	-	-	-	-	10
	CP_{SP_1}	-	-	97.3	87.4	48.9	41.5	**0**
	$CP_{SP_{1\rightarrow 2}}$	-	-	**91.6**	**76.3**	**41.2**	**34.5**	**0**
	$CP + LNS$	-	-	97.3	87.2	43.4	35.2	**0**
$4 \times 25^{\dagger}$	$CP_{default}$	-	-	-	-	-	-	10
	$CP_{default+SB}$	-	-	-	-	-	-	10
	CP_{SP_1}	-	-	**99.0**	**91.4**	**45.1**	36.1	**0**
	$CP_{SP_{1\rightarrow 2}}$	-	-	96.2	86.0	47.5	39.8	**0**
	$CP + LNS$	-	-	99.3	92.1	**45.1**	**35.6**	**0**

The default CP solver settings struggle to find any solutions for instances beyond 2×5 in size. Furthermore, it would seem that the symmetry breaking constraints reduce performance. The initial CP formulation without symmetry breaking is able to find feasible solutions to one 2×10 instance, whereas the symmetry breaking model could not find feasibility for this problem size.

The single stage search phase that performed the strongest involved instantiating the robot bingo game facilitation variables, $y_{kj}\ \forall r_k \in R;\ i \in G$, first as was previously postulated. Somewhat surprisingly, however, is the substantial impact on solver performance compared to the default settings, improving MRE by $\geq 20\%$ for smaller instances and even more for larger problems. Such an improvement in MRE translates to a proportional increase in social and cognitive activity participation throughout the course of the day. The best double-phase instantiation involved fixing user bingo game participation x_{ij} variables first, and then robot bingo game facilitation (as in single stage) variables second. Referring to the table, double-stage instantiation offers benefit in a number of areas, particularly for the instances involving one and three robots in time limits ranging from 5 to 100 seconds. We note that both instantiation methods find solutions with just 0.1% MRE for the first instance size. It appears that it be-

comes difficult to exploit problem structure past the assignment of bingo game tasks for this MRTA problem. Instantiating other groups of variables first (e.g., recharge tasks or reminders) resulted in performance similar to the CP default settings, although still somewhat stronger.

Due to the first minute spent finding an incumbent, CP with large neighbourhood search (LNS) has better performance in the later run-time limits. The performance of the method is notably strong at the 300 second run-time limit for instances 2×10 and 3×15, where it outperforms the search phases by a significant $\geq 12\%$ MRE. The LNS method uses an instantiation strategy very similar to CP_{SP_1} for the first minute, and as such the values are very similar for those run-times. LNS successfully finds the best solution for the largest problem by the run-time limit, outperforming both search phase methods.

7 Conclusions and Future Work

We applied constraint programming (CP) and mixed-integer programming (MIP) to the planning and scheduling of multiple social robots within a retirement home. This problem required task allocation and scheduling, aiming to maximize bingo game participation, while minimizing an energy consumption component. The proposed approaches reason about disjoint time windows, cross-schedule precedence relationships, spatial transition times of both users and robots within the environment, and robot energy consumption/replenishment.

Initial numerical experiments using default solver settings indicate that CP significantly outperforms MIP for the studied problem. We present methods for further enhancing CP performance through search manipulations, as well as a method for generating provable bounds for this problem by solving a relaxed CP formulation. Specifically, we investigated single and double-stage grouped variable ordering heuristics, concluding that instantiating the variables related to bingo game facilitation first has high positive impact on solution quality, significantly outperforming the default settings of the CP solver. We also implement a large neighbourhood search (LNS) using a time window variable selection heuristic. This method significantly outperforms the other approaches for mid-sized instances, and yields the strongest performance on the largest instances within the run-time limit. Due to these promising results, we plan to investigate alternative LNS procedures in future work.

Overall, results indicate that CP is a promising technology for our retirement home application. The next step is to move from simulation-based experimentation to deployment on real robots. We are integrating the CP solver into our robot architecture for field trials. In parallel, we are continuing research on the use of constraint programming in rescheduling and replanning as this will be a key functionality of the deployed system.

Acknowledgment The authors would like to thank the Natural Sciences & Engineering Research Council of Canada (NSERC), Dr. Robot Inc., and the Canada Research Chairs (CRC) Program.

References

1. Alessandro E De Luca, S Bonacci, and G Giraldi. Aging populations: the health and quality of life of the elderly. *La Clinica Terapeutica*, 162(1):e13–8, 2010.
2. Colombo Francesca, Llena-Nozal Ana, Mercier Jérôme, and Tjadens Frits. *OECD Health Policy Studies Help Wanted? Providing and Paying for Long-Term Care: Providing and Paying for Long-Term Care*, volume 2011. OECD Publishing, 2011.
3. Roger Bemelmans, Gert Jan Gelderblom, Pieter Jonker, and Luc De Witte. Socially assistive robots in elderly care: A systematic review into effects and effectiveness. *Journal of the American Medical Directors Association*, 13(2):114–120, 2012.
4. Wing-Yue Geoffrey Louie, Tiago Vaquero, Goldie Nejat, and J Christopher Beck. An autonomous assistive robot for planning, scheduling and facilitating multi-user activities. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5292–5298. IEEE, 2014.
5. Kyle EC Booth, Tony T Tran, Goldie Nejat, and J Christopher Beck. Mixed-integer and constraint programming techniques for mobile robot task planning. *Robotics and Automation Letters*, 1(1):500–507, 2016.
6. Philippe Baptiste, Claude Le Pape, and Wim Nuijten. *Constraint-based scheduling: applying constraint programming to scheduling problems*, volume 39. Springer Science & Business Media, 2012.
7. Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006.
8. John N Hooker and Greger Ottosson. Logic-based benders decomposition. *Mathematical Programming*, 96(1):33–60, 2003.
9. Tobias Achterberg, Timo Berthold, Thorsten Koch, and Kati Wolter. Constraint integer programming: A new approach to integrate cp and mip. In *Integration of AI and OR techniques in constraint programming for combinatorial optimization problems*, pages 6–20. Springer, 2008.
10. Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming (CP 1998)*, pages 417–431. Springer, 1998.
11. Philippe Laborie and Daniel Godard. Self-adapting large neighborhood search: Application to single-mode scheduling problems. *Proceedings MISTA-07, Paris*, pages 276–284, 2007.
12. Lynne E Parker. L-alliance: Task-oriented multi-robot learning in behavior-based systems. *Advanced Robotics*, 11(4):305–322, 1996.
13. Sylvia C Botelho and Rachid Alami. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1234–1239. IEEE, 1999.
14. M Bernardine Dias and Anthony Stentz. Traderbots: A market-based approach for resource, role, and task allocation in multirobot coordination. 2003.
15. Brian P Gerkey and Maja J Matari. Sold!: Auction methods for multirobot coordination. *Robotics and Automation, IEEE Transactions on*, 18(5):758–768, 2002.
16. Lantao Liu, Nathan Michael, and Dylan Shell. Fully decentralized task swaps with optimized local searching. In *Proceedings of Robotics: Science and Systems*, 2014.
17. G Ayorkor Korsah, Balajee Kannan, Brett Browning, Anthony Stentz, and M Bernardine 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*, pages 115–122. IEEE, 2012.

18. Pascal Van Hentenryck and Vijay Saraswat. Strategic directions in constraint programming. *ACM Computing Surveys (CSUR)*, 28(4):701–726, 1996.
19. Alexander Nareyek, Eugene C Freuder, Robert Fourer, Enrico Giunchiglia, Robert P Goldman, Henry Kautz, Jussi Rintanen, and Austin Tate. Constraints and AI planning. *Intelligent Systems, IEEE*, 20(2):62–72, 2005.
20. Robert P Goldman, Karen Zita Haigh, David J Musliner, and Michael JS Pelican. Macbeth: a multi-agent constraint-based planner [autonomous agent tactical planner]. In *Digital Avionics Systems Conference, 2002. Proceedings. The 21st*, volume 2, pages 7E3–1. IEEE, 2002.
21. Arnaud Doniec, Noury Bouraqadi, Michael Defoort, Van Tuan Le, and Serge Stinckwich. Distributed constraint reasoning applied to multi-robot exploration. In *Tools with Artificial Intelligence, 2009. ICTAI'09. 21st International Conference on*, pages 159–166. IEEE, 2009.
22. Joost Broekens, Marcel Heerink, and Henk Rosendal. Assistive social robots in elderly care: a review. *Gerontechnology*, 8(2):94–103, 2009.
23. Tiago Vaquero, Sharaf Christopher Mohamed, Goldie Nejat, and J Christopher Beck. The implementation of a planning and scheduling architecture for multiple robots assisting multiple users in a retirement home setting. In *Artificial Intelligence Applied to Assistive Technologies and Smart Environments (AAAI 2015)*, 2015.
24. Philippe 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*, pages 148–162. Springer, 2009.
25. Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520, 2014.
26. Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.
27. Michael Drexl. Synchronization in vehicle routing-a survey of vrps with multiple synchronization constraints. *Transportation Science*, 46(3):297–316, 2012.
28. Wing-Yue Geoffrey Louie, Jie Li, Tiago Vaquero, and Goldie Nejat. A focus group study on the design considerations and impressions of a socially assistive robot for long-term care. In *Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on*, pages 237–242. IEEE, 2014.
29. Torsten Fahle, Stefan Schamberger, and Meinolf Sellmann. Symmetry breaking. In *Principles and Practice of Constraint Programming (CP 2001)*, pages 93–107. Springer, 2001.
30. Tom Carchrae and J Christopher Beck. Principles for the design of large neighborhood search. *Journal of Mathematical Modelling and Algorithms*, 8(3):245–270, 2009.
31. Kyle EC Booth, Tony T Tran, and J Christopher Beck. Logic-based decomposition methods for the travelling purchaser problem. In *Proceedings of the Thirteenth International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming (CPAIOR 2016)*, pages 55–64. Springer, 2016.