Planning a Robot's Search for Multiple Residents in a Retirement Home Environment

Markus Schwenk[1,2] and **Tiago Vaquero**[1] and **Goldie Nejat**[1] and **Kai O. Arras**[2]
[1]Autonomous Systems and Biomechatronics Laboratory
Department of Mechanical and Industrial Engineering, University of Toronto, Canada
[2]Social Robotics Laboratory
Department of Computer Science, University of Freiburg, Germany
markus.schwenk@mail.utoronto.ca, {tvaquero, nejat}@mie.utoronto.ca, arras@informatik.uni-freiburg.de

Abstract

In this paper we address the planning problem of a robot searching for multiple residents in a retirement home in order to remind them of an upcoming multi-person recreational activity before a given deadline. We introduce a novel Multi-User Schedule Based (M-USB) Search approach which generates a high-level plan to maximize the number of residents that are found within the given time frame. From the schedules of the residents, the layout of the retirement home environment as well as direct observations by the robot, we obtain spatio-temporal likelihood functions for the individual residents. The main contribution of our work is the development of a novel approach to compute a reward to find a search plan for the robot using: 1) the likelihood functions, 2) the availabilities of the residents, and 3) the order in which the residents should be found. Simulations were conducted on a floor of a real retirement home to compare our proposed M-USB Search approach to a Weighted Informed Walk and a Random Walk. Our results show that the proposed M-USB Search finds residents in a shorter amount of time by visiting fewer rooms when compared to the other approaches.

1 Introduction

The health and quality of life of older adults living in long-term care facilities can be improved by these individuals engaging in stimulating recreational activities such as playing games, playing musical instruments, doing crossword puzzles, or reading (Menec 2003). These types of activities can delay age-related health decline (Bath and Deeg 2005) and prevent social isolation (Findlay 2003), which could potentially decrease the risk of dementia in elder adults (Wilson et al. 2007). However, the lack of these activities in eldercare facilities (PriceWaterCoopers LLP 2001) exists due to a shortage of healthcare workers (Sharkey 2008), which could be aggravated in the near future due to the rapid growth of the elderly population (Centre for Health Workforce Studies 2006). Socially assistive robots have been shown to be a promising technology to assist the elderly and to support caregivers in eldercare facilities (Oida et al. 2011; McColl, Louie, and Nejat 2013).

This is an extended version of our paper in AAAI 2014

Our research focuses on the development of socially assistive robots that can autonomously organize and facilitate group-based recreational activities for the elderly. In this paper, we address the planning problem of a robot searching for multiple residents in a retirement home environment in order to invite and remind them of an upcoming multi-person recreational activity. The robot's objective is to maximize the number of residents it finds in a given time frame before the activity starts. During the search, the robot has to consider that the residents have their own schedules which contain appointments in different rooms of the environment, during which the residents are not always available for interaction with the robot. Based on these schedules, the robot also considers the order in which the residents have to be found to avoid searching for unavailable people. We introduce a novel Multi-User Schedule Based (M-USB) Search method which plans the robot's search for a set of non-static residents in a structured environment within a given time frame based on the residents' daily schedules.

Robotic search for people in structured environments has been investigated in the literature for different scenarios. For example, in (Elinas, Hoey, and Little 2003), the robot HOMER was designed to deliver messages one at a time to a particular person in a workspace environment. The robot stored a likelihood function for each person's location and performed a best-first search. The search consisted of visiting the nearest location to the robot based on the likelihood function for a particular person. If the person was not found, the robot iteratively visited other rooms until either the person was found or all regions had been visited. For these scenarios, a person was assumed to be at a static location in the environment. The search for multiple static targets in an indoor environment has been addressed in (Lau, Huang, and Dissanayake 2005). A dynamic programming approach was used to plan the search on a topological ordered graph, using a probability distribution which models the probability of meeting one of the targets in a given room at a given time. In (Tipaldi and Arras 2011), a robot's ability to blend itself into the workflows of people within human office environments was addressed. The authors developed a spatial Poisson process which was learned from observations of people to represent spatio-temporal patterns of human activities. A Markov Decision Process (MDP) Model was used to generate a robot's path through the environment to maximize the

probability of encountering a person. In (Lau, Huang, and Dissanayake 2006), the problem of a robot searching for a moving target within an indoor environment has been addressed. The Optimal Searcher Path (OSP) Problem, which models the search for static targets as sequentially searching in adjacent equal-sized parts of an environment, was extended to: 1) handle different room sizes, and 2) to search for a single moving target whose movements were modelled as a Markov Process. A probability distribution over the different regions was used to express the knowledge about the target's location. A branch and bound method was proposed in order to plan a sequence of regions the robot had to visit in order to maximize the robot's chances of finding the target in a given amount of time.

Uniquely our work addresses the robotic search problem of finding a specific set of multiple moving residents in a retirement home setting considering their individual daily schedules. Such schedule-based multi-user search for non-static people has not yet been addressed in the literature. To address this problem, we obtain spatio-temporal likelihood functions for every resident of the retirement home. We propose an approach to generate these resident likelihood functions as a composition of: 1) the residents' schedules, 2) direct observations by the robot in the environment, and 3) the layout of the environment. To do this, a weighting is applied to the above sources of information based on their time-dependent certainties of predicting a person's location. The main contribution of our work is the development of an MDP planner that uses a novel approach to compute the reward to determine the robot's search plan for finding multiple residents using: 1) the resident likelihood functions, 2) the availabilities of the individual residents, and 3) the order in which the residents should be found.

The paper is organized as follows. In section 2 we introduce and formalize the M-USB Search approach. We describe the resident likelihood functions and define an MDP Model and an algorithm which generates the robot's search plan. In section 3 we describe the simulated experiments as well as the experiment results and discussion. Concluding remarks are presented in section 4.

Environment. We model the environment as a set of regions $\mathbb{R} \in \mathbb{R}^E$ (e.g. rooms, corridors, common areas) in which the search takes place. For each person $p \in \mathbb{P}$, where $\mathbb{P}$ is the set of all residents who are living in the environment, we assign one region of the environment as solely that person's: his/her private room. For each region, a room-class $\mathbb{C} = class(\mathbb{R}) \in \mathbb{C}^E$ is assigned, e.g. "Common Room", "Bedroom", "Corridor", or "Dining Hall". The room-classes depend on the activities the residents engage in when they are in these regions. The room-class $\mathbb{C}_{private} \in \mathbb{C}^E$ in particular is the room-class which contains all private rooms. We define $regions(\mathbb{C})$ to contain all regions $\mathbb{R}$ with $\mathbb{C} = class(\mathbb{R})$.

2 M-USB Search

The Multi-User Schedule Based Search presented in this work is a planning procedure that provides a plan $\mathcal{P}^*$ for a robot to find as many people as possible from a given set of target people in a given order within a defined time frame. The retirement home environment is defined to consist of several different regions which represent the topology of the building (e.g., rooms and corridors). The plan $\mathcal{P}^*$ will include a sequence of actions which model the whole search process. The possible actions are: *drive* which lets the robot travel from one region to another; *rest* which lets the robot rest for a short period of time; and *search* in which the robot executes a low-level search procedure in a specific region (e.g., frontier exploration, random walk). We use backwards induction to compute $\mathcal{P}^*$ based on a Markov Decision Process which models the search using the aforementioned actions. To obtain a reward for this MDP, we setup a likelihood function for each person which models the probability that the person is in a specific region at a given time of the day. The likelihood functions of the individual residents are combined to generate a reward which respects the target residents' availability constraints (obtained from their schedules) and the order in which the residents should be found.

This paper will present the computation of the plan $\mathcal{P}^*$. This plan is to be executed by a mobile socially assistive robot that can navigate the environment as well as detect and recognize individual residents. Once a specific person is found, the robot greets the person and verbally provides a reminder to him/her. We assume a local-search routine already exists on the robot which allows the robot to search for people in a given region. Once a person has been detected,

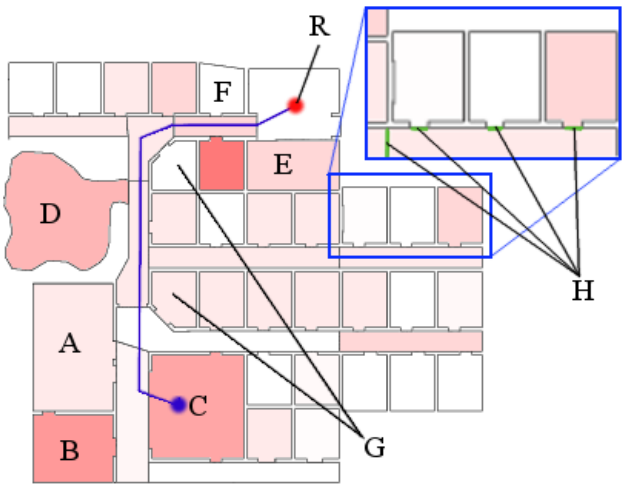


Figure 1: The map of the simulated retirement home with the rooms: dining hall (A), games room (B), TV-room (C), garden (D), nurse station (E), family visit room (F) and shower rooms (G). All other rooms are private rooms and corridors. The color of the regions indicates the current reward for each region (red: high, white: low) of an example scenario. The current generated plan has the robot (R) driving to the TV-room (blue trajectory) where it starts a local search. (H) shows the crossable edges (doors) in the scaled portion of the environment.

the robot has to compute a new plan (replanning).

Problem Setup

Each region can be represented as a polygon. The edges of this polygon can be marked as "crossable" if there is no physical border at an edge (e.g., if the edge represents a doorway). We define $neighbours(\mathbb{R})$ to be all regions which share a crossable edge with $\mathbb{R}$, i.e., a person or a robot can move from $\mathbb{R}$ to any region $\mathbb{R}_n \in neighbours(\mathbb{R})$ without entering a third region. Figure 1 shows an example environment.

Schedules and Availability of Residents. For each person $p \in \mathbb{P}$, we consider a schedule which defines all of his/her appointments on a given day. An appointment has a start time and an end time, and is assigned to a region $\mathbb{R}$ in which it takes place. We model the availability of each person $p \in \mathbb{P}$ as function $\beta_p(t)$ such that $\beta_p(t) = 1$ if p is available at time t and $\beta_p(t) = 0$ otherwise.

Motion Model. We assume a simple motion model for the residents. We model the probability $p_m(\mathbb{R}_1, \mathbb{R}_2, p, \Delta t)$ that person p has moved from region $\mathbb{R}_1$ to region $\mathbb{R}_2$ in the time frame Δt. This probability can be obtained from the person's speed v_p and the distance $d(\mathbb{R}_1, \mathbb{R}_2)$ between the two regions. We define the set $\mathbb{R}^r(\mathbb{R}, p, \Delta t)$ which contains all regions which the person could have entered:

$$\mathbb{R}^r(\mathbb{R}, p, \Delta t) = \{\mathbb{R}' \,|\, d(\mathbb{R}, \mathbb{R}') \leq v_p \cdot \Delta t\}\,. \tag{1}$$

The value of $p_m(\mathbb{R}_1, \mathbb{R}_2, p, \Delta t)$ can then be obtained as:

$$p_m(\mathbb{R}_1, \mathbb{R}_2, p, \Delta t) = \begin{cases} \mu, & \text{if } \mathbb{R}_2 \in \mathbb{R}^r(\mathbb{R}_1, p, \Delta t) \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

with $\mu = |\mathbb{R}^r(\mathbb{R}_1, p, \Delta t)|^{-1}$. This motion model considers all possible movements of the resident within the environment.

Search Query. When the robot receives a query q, it is to find a set of residents $p \in \mathbb{P}_q \subseteq \mathbb{P}$ within a given deadline t_{max}. Query q also specifies the order in which the residents should be found.

Setting up Resident Likelihood Functions

For each resident $p \in \mathbb{P}$, we can set up a likelihood function $\mathcal{L}(p, \mathbb{R}, t)$ which represents the probability that p is in region $\mathbb{R}$ at time t. This individual likelihood function is composed of four different weighted likelihood functions $\mathcal{L}_k(p, \mathbb{R}, t)$, with $k \in \{s, lkrl, l, env\}$. These likelihood functions are: $\mathcal{L}_s$ which is obtained by analyzing the resident's schedule; $\mathcal{L}_{lkrl}$ which is based on the last known location of the resident; $\mathcal{L}_l$ which describes the resident's behaviour that the robot has learned; and $\mathcal{L}_{env}$ which can be obtained from the structure of the environment. We will discuss these likelihood functions in the following sections. As convention, we define $0 \leq \mathcal{L}_k \leq 1$ and $\sum_{\mathbb{R}} \mathcal{L}_k(p, \mathbb{R}, t) = 1$ for each person $p \in \mathbb{P}$ and each likelihood function $\mathcal{L}_k$. A value $\mathcal{L}_k(p, \mathbb{R}, t) = 1$ means that the person is in $\mathbb{R}$ at time t while $\mathcal{L}_k(p, \mathbb{R}, t) = 0$ indicates that the person cannot be in the region at this time.

Schedule Analyzer. We model $\mathcal{L}_s(p, \mathbb{R}, t)$ using the schedule of resident p. Assuming that with a probability of $0 \leq p_a \leq 1$ the person participates in an appointment defined in his/her schedule, we set $\mathcal{L}_s(p, \mathbb{R}, t) = p_a$ for the time frame of the appointment for the region assigned to this particular appointment, and $\mathcal{L}_s(p, \mathbb{R}, t) = \frac{(1-p_a)}{|\mathbb{R}^E|-1}$ for all other regions. For any time t_k between two appointments where the last appointment ends at time t_{k-1} and the next appointment starts at t_{k+1}, we define $\alpha_{k-1} = \frac{t_{k+1}-t_k}{t_{k+1}-t_{k-1}}$ and $\alpha_{k+1} = \frac{t_k-t_{k-1}}{t_{k+1}-t_{k-1}}$. If there is no next appointment, we set $\alpha_{k+1} = 0$ and $\alpha_{k-1} = 1$. If there is no previous appointment, $\alpha_{k+1} = 1$ and $\alpha_{k-1} = 0$. We can then define the likelihood function to be:

$$\begin{aligned} &\mathcal{L}_s(p, \mathbb{R}, t_k) = \\ &\quad \alpha_{k-1} \cdot \sum_{\mathbb{R}' \in \mathbb{R}^{\mathbb{E}}} \mathcal{L}_s(p, \mathbb{R}', t_{k-1}) \cdot p_m(\mathbb{R}', \mathbb{R}, p, t_k - t_{k-1}) + \\ &\quad \alpha_{k+1} \cdot \sum_{\mathbb{R}' \in \mathbb{R}^{\mathbb{E}}} \mathcal{L}_s(p, \mathbb{R}', t_{k+1}) \cdot p_m(\mathbb{R}, \mathbb{R}', p, t_{k+1} - t_k) \end{aligned} \tag{3}$$

Last Known Resident Location. To be able to take into account when the robot last detected a person earlier that day who it is currently searching for, we set up a database which stores the time t_d^p and region $\mathbb{R}_d^p$ of the last detection of person p. Using the motion model of the resident, this information can be used to generate $\mathcal{L}_{lkrl}(p, \mathbb{R}, t)$:

$$\mathcal{L}_{lkrl}(p, \mathbb{R}, t) = p_m(\mathbb{R}_d^p, \mathbb{R}, p, t - t_d^p). \tag{4}$$

If the person has not been previously detected, we assign a uniform distribution $\mathcal{L}_{lkrl}(p, \mathbb{R}, t) = \frac{1}{|\mathbb{R}^E|}$.

Learned Behaviour. A person's behaviour, which is not defined in the schedule (e.g., a person often takes a walk in the garden after lunch) but which has been learned by the robot based on its observations is also stored by the robot as $\mathcal{L}_l(p, \mathbb{R}, t_k)$, where t_k indicates a time step. If no data from the previous days exist, we assign a uniform distribution $\mathcal{L}_l(p, \mathbb{R}, t_k) = \frac{1}{|\mathbb{R}^E|}$. During the day the robot remembers whether it detected person p in time step t_k in region $\mathbb{R}$. It saves this information in $g(p, \mathbb{R}, t_k)$ which is either 1 when the person has been detected or 0 otherwise. We define:

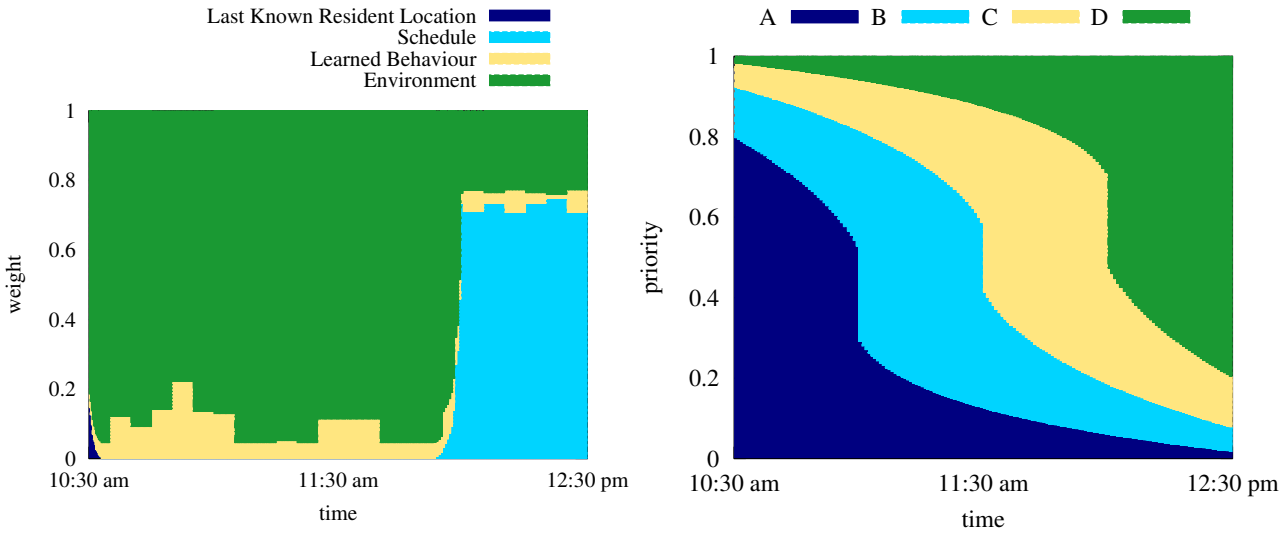
$$f(p, t_k) = \sum_{\mathbb{R} \in \mathbb{R}^E} g(p, \mathbb{R}, t_k) \tag{5}$$

to be the number of regions in which person p has been detected in time step t_k. At the end of the day the updated likelihood function evolves to:

$$\mathcal{L}'_l(p, \mathbb{R}, t_k) = \alpha \cdot \frac{g(p, \mathbb{R}, t_k)}{f(p, t_k)} + (1 - \alpha) \cdot \mathcal{L}_l(p, \mathbb{R}, t_k) \tag{6}$$

with $0 \leq \alpha \leq 1$ if $f(p, t_k) \geq 1$ and $\mathcal{L}'_l(p, \mathbb{R}, t_k) = \mathcal{L}_l(p, \mathbb{R}, t_k)$ otherwise.

Environment. The topology of the environment can be used to generate $\mathcal{L}_{env}(p, \mathbb{R}, t)$. We obtain $\mathbb{C}^p_{private}$ by removing the private room of p from $\mathbb{C}_{private}$ and define a



(a) Weights for one Person

(b) Priority Function.

Figure 2: (a) Weights for one person during the time frame of 10:30 am to 12:30 pm. At 12:00 pm the person has a one-hour appointment which gives the schedule a higher weight at this time. (b) Priority function for four residents who are all available in the considered time frame.

new room-class $\mathbb{C}^{p'}_{private}$ for the resident p, containing only his/her private room. We define $\mathbb{C}^p$ to be:

$$\mathbb{C}^p = \mathbb{C}^E \setminus \{\mathbb{C}_{private}\} \bigcup \left\{\mathbb{C}^p_{private}, \mathbb{C}^{p'}_{private}\right\}. \quad (7)$$

For each room-class $\mathbb{C} \in \mathbb{C}^p$, we define a probability $p_{\mathbb{C}}(p)$ to be the probability that person p is in one of the rooms in $regions(\mathbb{C})$. Assuming that a person will spend most of her/his spare time either in her/his private room or in the common rooms, we assign higher values for $\mathbb{C}^{p'}_{private}$ and the room-class containing the common rooms. We apply the constraint:

$$\sum_{\mathbb{C} \in \mathbb{C}^p} p_{\mathbb{C}}(p) = 1. \quad (8)$$

We define $\mathcal{L}_{env}(p, \mathbb{R}, t)$ to be:

$$\mathcal{L}_{env}(p, \mathbb{R}, t) = \frac{p_{class(\mathbb{R})}(p)}{|regions(class(\mathbb{R}))|} \quad (9)$$

with $class(\mathbb{R}) \in \mathbb{C}^p$. $\mathcal{L}_{env}(p, \mathbb{R}, t)$ is constant over the day.

Pre-computation. Since the schedules, the topology of the environment, and the learned behaviour remain the same once they are provided to the robot at the beginning of a day, $\mathcal{L}_s(p, \mathbb{R}, t)$, $\mathcal{L}_l(p, \mathbb{R}, t)$, and $\mathcal{L}_{env}(p, \mathbb{R}, t)$ can be computed before a search query is received. $\mathcal{L}_{lkrl}(p, \mathbb{R}, t)$ is computed dynamically when the query is received.

Combining the Likelihood Functions for one Person. The four likelihood functions $\mathcal{L}_k(p, \mathbb{R}, t)$ can be combined to generate $\mathcal{L}(p, \mathbb{R}, t)$. As the certainties with which the four likelihood functions can predict a resident's location differ (e.g., the Last Known Resident Location will have high uncertainty when the person has not been detected for several hours, and the Schedule Analyzer will have high certainty when the person has an appointment), a weighting function can be used. The certainty of one likelihood function $\mathcal{L}_k(p, \mathbb{R}, t)$ can be represented by its variance:

$$\mathrm{Var}(\mathcal{L}_k(p, \mathbb{R}, t)) = \frac{1}{|\mathbb{R}^E|} \cdot \sum_{\mathbb{R} \in \mathbb{R}^E} \left[\mathcal{L}_k(p, \mathbb{R}, t) - \frac{1}{|\mathbb{R}^E|}\right]^2. \quad (10)$$

For each likelihood function $\mathcal{L}_k$, we introduce the weight w_k at time t:

$$w_k(t) = \frac{\mathrm{Var}(\mathcal{L}_k(p, \mathbb{R}, t))}{\sum_k \mathrm{Var}(\mathcal{L}_k(p, \mathbb{R}, t))} \quad (11)$$

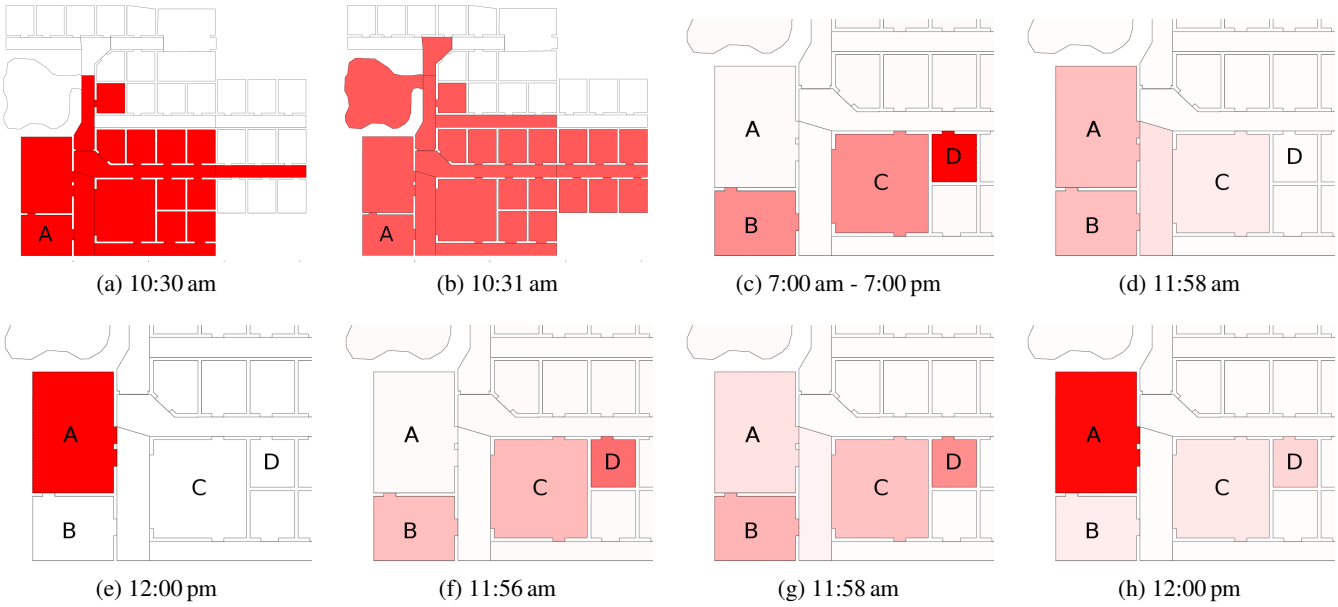
where $\sum_k w_k(t) = 1$. The final combined likelihood function is defined to be:

$$\mathcal{L}(p, \mathbb{R}, t) = \sum_k w_k(t) \cdot \mathcal{L}_k(p, \mathbb{R}, t). \quad (12)$$

Figure 2(a) shows an example of four weights used for one resident. Figure 3 shows examples for the different likelihood functions as well as the combined likelihood function for one resident.

Modelling the Transition System

The objective is to find a sequence of actions the robot should execute in order to find as many persons as possible in $\mathbb{P}_q$ within the given deadline t_{max}. We model the search as a Markov Decision Process. We discretize time using time steps of duration Δt which is the execution time for each individual action. The M-USB Search is modelled to consist of three possible action sequences that the robot can perform



(a) 10:30 am (b) 10:31 am (c) 7:00 am - 7:00 pm (d) 11:58 am

(e) 12:00 pm (f) 11:56 am (g) 11:58 am (h) 12:00 pm

Figure 3: Different likelihood functions for one resident (dark red: high likelihood, white: low likelihood). (a, b) $\mathcal{L}_{lkrl}(p, \mathbb{R}, t)$ three and four minutes after the person has been detected in the Games Room (A) at 10:27 am, (c) $\mathcal{L}_{env}(p, \mathbb{R}, t)$ for the resident living in D. B and C are common rooms. (d, e) $\mathcal{L}_s(p, \mathbb{R}, t)$ when the resident has an appointment in the Dining Hall (A) at 12 pm, and (f-h) the combined resident likelihood function $\mathcal{L}(p, \mathbb{R}, t)$ before and during the appointment.

in each region: 1) $rest$ in which the robot rests for one time step, 2) $search$ in which the robot performs a local search within the region, and 3) $drive$ in which the robot drives to one of the neighbouring regions. Since the time it takes to perform a local search within a region depends on the geometry of the region, we introduce $\mathbb{T}^s_k$ to represent the number of time steps Δt a search within region $\mathbb{R}_k$ takes. For region $\mathbb{R}_l \in neighbours(\mathbb{R}_k)$, we define $\mathbb{T}^d_{k,l}$ as being the number of time steps the transition between $\mathbb{R}_k$ and $\mathbb{R}_l$ takes.

The states $s \in \mathbb{S}$ of the MDP represent the states of the robot, which depend on the region the robot is in and the current action sequence it is performing. We define the following sets of states $\mathbb{S}^s_k$ and $\mathbb{S}^d_k$ to contain all states during the $search$ and $drive$ sequences for each region $\mathbb{R}_k \in \mathbb{R}^E$:

1. $\mathbb{S}^s_k = \{search^t_k\}$ with $0 \leq t < \mathbb{T}^s_k - 1$, and
2. $\mathbb{S}^d_k = \bigcup_{\mathbb{R}_l} \left\{drive^t_{k,l}\right\}$ with $0 \leq t < \mathbb{T}^d_{k,l} - 1$, $\mathbb{R}_l \in neighbours(\mathbb{R}_k)$.

In addition to the aforementioned states within the $search$ and $drive$ sequences, we define the state s'_k for each region $\mathbb{R}_k$ as the robot's state: 1) after a region has been entered by any drive sequence, 2) after the rest action has been executed, 3) after the full search sequence of $\mathbb{R}_k$ has been executed, or 4) when the robot is creating a new plan when being in $\mathbb{R}_k$. The set of all possible states for region $\mathbb{R}_k$ is defined to be:

$$\mathbb{S}_k = \{s'_k\} \cup \mathbb{S}^s_k \cup \mathbb{S}^d_k. \tag{13}$$

We define the following robot actions α for each region $\mathbb{R}_k \in \mathbb{R}^E$:

1. $\mathbb{A}^r_k = \{rest_k\}$,
2. $\mathbb{A}^s_k = \{search^t_k\}$ with $0 \leq t < \mathbb{T}^s_k$, and
3. $\mathbb{A}^d_k = \bigcup_{\mathbb{R}_l} \left\{drive^t_{k,l}\right\}$ with $0 \leq t < \mathbb{T}^d_{k,l}$, $\mathbb{R}_l \in neighbours(\mathbb{R}_k)$.

For each region $\mathbb{R}_k$, the set of all possible actions is:

$$\mathbb{A}_k = \mathbb{A}^r_k \cup \mathbb{A}^s_k \cup \mathbb{A}^d_k. \tag{14}$$

Transitions between the states describe how a robot state changes when it performs a particular action. In particular, the successor $succ(\alpha)$ of action α is defined as the state which follows α. The overall transition system is shown in Figure 4.

For each action a time-dependent reward $R(\alpha, t)$ is assigned. This reward is evaluated to compute the plan $\mathcal{P}^*$ and depends on the region in which the action is performed. Therefore, we define a region to each action: $\mathbb{R} = region(\alpha)$. For each action α in $\mathbb{A}^r_k$ and $\mathbb{A}^s_k$, we define $region(\alpha) = \mathbb{R}_k$. For each pair of neighbouring regions $\mathbb{R}_k$ and $\mathbb{R}_l$, we define $\mathbb{T}^{d,cross}_{k,l}$ to be the number of time steps after which the region $\mathbb{R}_l$ is entered during a drive sequence from $\mathbb{R}_k$ to $\mathbb{R}_l$. We then define $region(\alpha) = \mathbb{R}_k$ if $t < \mathbb{T}^{d,cross}_{k,l}$ and $region(\alpha) = \mathbb{R}_l$ if $t \geq \mathbb{T}^{d,cross}_{k,l}$ for each drive action α.

Resident Detection Probability of Actions. Assuming that during a search the probability that a person who is in the searched region is detected is different from the probability that a person is detected while the robot is

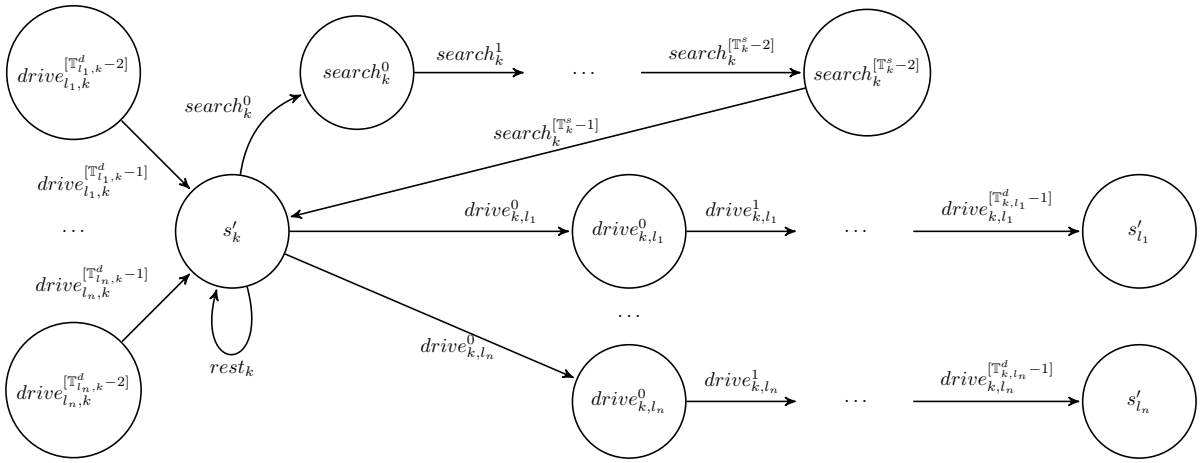


Figure 4: The transition system for an arbitrary region k with neighbours $l_1, \ldots, l_n$. Being in state s'_k, the robot can either *rest* one time step, start a full *search* sequence or a *drive* sequence to one of the neighbouring regions l_i with $i = 1, \ldots, n$ which leads to state s'_{l_i}. *Search* sequences and the *rest* action in region k lead to s'_k.

just driving between two regions or resting in one region, we define an attractivity $\delta(\alpha)$ with $0 \leq \delta(\alpha) \leq 1$ for each action α. This attractivity depends on the geometry of the region $\mathbb{R} = region(\alpha)$. We define the attractivity $\delta(\alpha_s) = \mathcal{A}(region(\alpha_s))^{-1}$ for each search action α_s with $\mathcal{A}(region(\alpha_s))$ being the area of $region(\alpha_s)$. We define $\delta(\alpha_r) = a \cdot \delta(\alpha_s)$ with $0 < a \leq 1$ for each rest action and $\delta(\alpha_d) = b \cdot \delta(\alpha_s)$ with $0 \leq b \leq 1$ for each drive action. If the robot detects residents with higher probability while driving then $b > a$ holds; $a > b$ holds otherwise.

Modelling the Order and Availability of Residents

The order in which the residents should be found is given and has been obtained from the persons' schedules to avoid searching for unavailable residents. The robot should try to keep this order if possible. However, if the robot can maximize the number of people found by changing the order, it can also do so. To search for the residents $p \in \mathbb{P}_q$ in the given order, we introduce a priority function $\pi_p(t)$ with $0 \leq \pi_p(t) \leq 1$ for each person $p \in \mathbb{P}_q$ and apply the following constraints:

$$\sum_{p \in \mathbb{P}_q} \pi_p(t) \cdot \beta_p(t) = 1 \; \forall t \tag{15}$$

and

$$\int_{t_0}^{t_{max}} \pi_p(t) \cdot \beta_p(t)\, dt = \frac{t_{max} - t_0}{|\mathbb{P}_q|} \;\; \forall p \tag{16}$$

which are used to ensure that the same search effort is applied to all residents during the search process. We model $\pi_p(t)$ to provide a high priority to the time interval assigned to the resident p based on the given order. However, to allow the robot to search for other residents $p' \in \mathbb{P}_q$ during this time interval, we allow $\pi_{p'}(t) \neq 0$ when $\beta_{p'}(t) = 1$. Figure 2(b) shows such a priority function for four people A, B, C, and D to be searched in this order.

Finding $\mathcal{P}^*$

In order to find the set of residents within the given deadline, we define a reward for each action of the MDP model of the search. The reward is based on the resident likelihood functions and the availabilities of the residents in $\mathbb{P}_q$, the aforementioned priority functions and the attractivities:

$$R(\alpha, t) = \delta(\alpha) \cdot \sum_{p \in \mathbb{P}_q} \pi_p(t) \cdot \beta_p(t) \cdot \mathcal{L}(p, region(\alpha), t). \tag{17}$$

Since a deadline t_{max} is given, the search evolves to be a finite horizon MDP which can be solved using backwards induction (Tipaldi and Arras 2011). In particular, the utility $U_t(s)$ is evaluated for each possible state s at time t using the Bellman equation:

$$U_t(s) = \max_{\alpha}[R(\alpha, t) + \gamma \cdot U_{t+1}(succ(\alpha))] \tag{18}$$

where α is any action that can be taken from s and γ is a factor with $0 \leq \gamma \leq 1$ which provides a weighting for the relationship between the importance of rewards which are earned in the near and in the far future.

The policy $\Pi_t(s)$ defines the action α the robot should take when in state s at time t in order to maximize the reward:

$$\Pi_t(s) = \arg\max_{\alpha}[R(\alpha, t) + \gamma \cdot U_{t+1}(succ(\alpha))]. \tag{19}$$

We also define the aforementioned finite horizon $H = t_{max} \cdot \Delta t^{-1}$, which is the number of time steps Δt from the start of the planning process to t_{max}. The initial state s_0 is the state the robot is in when the plan is determined. Since the planning procedure will be called at the beginning of the search and whenever a person has been found, we can define $s_0 = s'_k$ with $\mathbb{R}_k$ being the region the robot is in.

Given the computed policy $\Pi_{t \ldots H-1}$, we can generate the plan $\mathcal{P}^*_{t,s} = \{\mathcal{P}^*(t), \mathcal{P}^*(t+1), \ldots, \mathcal{P}^*(H-1)\}$. This plan is a sequence of actions as shown in Algorithm 1.

Algorithm 1 $\mathcal{P}^*_{t,s}(\Pi_{t\dots H-1})$

$\mathcal{P}^*(t) = \Pi_t(s)$;
for $k \leftarrow t+1$ **to** $H-1$ **do**
  $\mathcal{P}^*(k) = \Pi_k(succ(\mathcal{P}^*(k-1)))$;
end for
return $\mathcal{P}^*$;

From the backwards induction approach, we know the plan $\mathcal{P}^*_{t+1,\alpha} = \mathcal{P}^*_{t+1,succ(\alpha)}(\Pi_{t+1\dots H-1})$ when choosing an action $\Pi_t(s) = \alpha$ at time step t. Since we want to decrease the reward of a *rest* or *search* action when the region has already been searched in this plan in order to avoid endless search loops, we introduce a factor $h(\mathcal{P}^*_{t+1,\alpha}, \alpha)$ which reduces the reward when $region(\alpha)$ has been searched in $\mathcal{P}^*_{t+1,\alpha}$. We define the resulting reward as:

$$R'(\alpha, t) = h(\mathcal{P}^*_{t+1,\alpha}, \alpha) \cdot R(\alpha, t). \tag{20}$$

The value $0 \leq h(\mathcal{P}^*_{t+1,\alpha}, \alpha) \leq 1$ depends on when and how often $region(\alpha)$ has been searched in this plan. This greedy approach is shown in Algorithm 2 which can be used to compute the entire plan $\mathcal{P}^*$.

Algorithm 2 M-USB Planning

Input: $R(\alpha, t)$, t_{max}, $\mathbb{S}$, $s_0 \in \mathbb{S}$, γ;
Output: Reward maximizing plan $\mathcal{P}^*$;
$H \leftarrow t_{max}/\Delta t$;
$U_H(s) \leftarrow 0\ \forall s \in \mathbb{S}$;
for $t \leftarrow H-1$ **to** 0 **do**
  $U_t(s) = \max_{\alpha}[R'(\alpha, t) + \gamma \cdot U_{t+1}(succ(\alpha))]$;
  $\Pi_t(s) = \arg\max_{\alpha}[R'(\alpha, t) + \gamma \cdot U_{t+1}(succ(\alpha))]$;
end for
return $\mathcal{P}^*_{0,s_0}(\Pi_{0\dots H-1})$;

3 Simulated Experiments

Simulation Setup

To test the performance of the M-USB Search, we use a simulator we have developed to simulate a robot in a realistic retirement home environment. The simulation was executed on a Ubuntu machine with an AMD A10-5700 Processor and 12GB RAM.

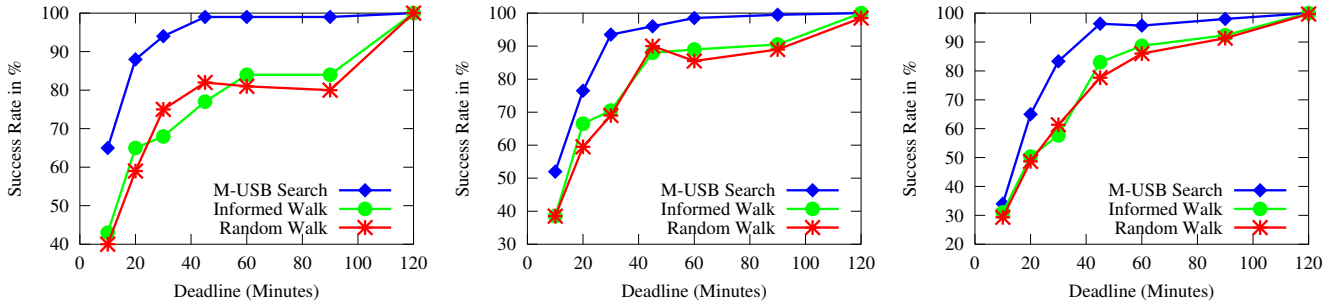
Simulation Environment. We created a map of a floor in a retirement home with 25 residents. The map consists of the residents' private rooms, two common rooms (TV-Room and Games Room), one Dining Hall, two Shower rooms, one Nurse Station, one Room for Family visits, and an outdoor Garden. All residents have their own unique schedules for the day. These schedules contain three meal times, breakfast (8 am-9 am), lunch (12 pm-1 pm), and dinner (6 pm-7 pm) during which the residents are available for the robot to interact with them. In addition, each schedule includes one 1-hour activity during which the residents are also available for interaction (e.g., walk and reading) and 2 to 4 appointments during which they must not be disturbed (e.g., doctor's visit). In his/her spare time, each resident visits random rooms at random times. A probability of $p_{miss} = 0.1$ is given for the residents not participating in their scheduled activities and behaving as if they have spare time. The simulated residents move with a speed of $v_p = 0.15\,\mathrm{m/s}$. The map used for these experiments is shown in Figure 1.

Performance Comparison. We compare the performance of our M-USB Search to both a *Weighted Informed Walk* and a *Random Walk* approach for the problem of a robot finding a group of residents within a deadline in the retirement home setting in order to remind them of an upcoming group-based recreational activity. The robot uses a speed of $v = 0.6\,\mathrm{m/s}$ and can detect people within a sensing range of $r = 1.8\,\mathrm{m}$ with respect to itself. If one of the searched residents is found, the robot stops for 1 minute in order to interact with the found person. The investigated search algorithms are:

1. **Random Walk.** The robot chooses a random room in the map, drives to this room and starts a local search in the room. This is repeated until all target residents are found or until the deadline is reached.
2. **Weighted Informed Walk.** Similar to the Random Walk, the Weighted Informed Walk algorithm picks a random room, drives there and starts a search in this room. However, a higher weighting is given to a resident's private room and common rooms. Namely, a weighting technique is applied to identify the importance of the regions accordingly to their room-classes. The weights for the different room-classes are: 0.5 for the room-class containing the private rooms of all searched residents; 0.25 for the room-class containing the common rooms; and 0.25 for a third room-class containing all other rooms. We assign an individual weight to each room in the environment. Namely, this individual weight is defined to be the corresponding weight for the room-class the room of interest is in divided by the number of rooms in this room-class. The robot applies a universal stochastic sampling technique based on the individual weights of the rooms to choose a room to search. The algorithm also considers the last 4 regions it has searched and does not search them again before 4 other regions have also been searched.
3. **M-USB-Search.** The proposed M-USB Search is used with a time discretization of $\Delta t = 10\,\mathrm{s}$. The schedule analyzer uses $\Delta t = 30\,\mathrm{s}$ and the database in which the robot saves the learned behaviour operates with $\Delta t = 300\,\mathrm{s}$. The attractivities for the actions are $\delta(\alpha_d) = 0.9 \cdot \delta(\alpha_s)$

Table 1: The $p_{\mathbb{C}}$ values used in the experiments.

Room Class	$p_{\mathbb{C}}$
Common Rooms	0.35
Corridors	0.05
Resident's private room ($\mathbb{C}^{p'}_{private}$)	0.4
Rooms in other room-classes	0.2



(a) N=5 (b) N=10 (c) N=15

Figure 5: Comparison Results: success rates for the different N.

Table 2: Comparison Results: mean search time per person, and number of visited regions within the environment for the different N.

Approach	Mean Search Time (Min.)			Visited Regions (mean)		
	N = 5	N = 10	N=15	N = 5	N = 10	N=15
M-USB Search	10.3	12.1	16.5	48	55	68
Weighted Informed Walk	15.5	18.1	23.2	176	194	202
Random Walk	14.6	15.9	24.0	119	137	214

and $\delta(\alpha_r) = 0.7 \cdot \delta(\alpha_s)$. For Eqs. (18) and (19), $\gamma = 0.99$ is used. For the learned behaviour we use $\alpha = 0.1$ in Eq. (6). In Eq. (9) the probabilities in Table 1 are applied to determine $\mathcal{L}_{env}(p, \mathbb{R}, t)$. To avoid endless search loops the robot needs to search 4 other rooms before searching the same room again. In particular, the value of $h(\mathcal{P}^*_{t+1,\alpha})$ for action α is set to zero when the room $region(\alpha)$ is contained in the next 4 searched regions in $\mathcal{P}^*_{t+1,\alpha}$.

Local Search in a Region. As our focus in this paper is on the high-level search to regions, for this comparison all aforementioned search approaches use the same random walk local search approach when they are searching within the region. The search time in the individual rooms is set to one second per squared meter.

The Search Queries. Each search approach was tested with different search queries q, which consisted of a robot finding $N = |\mathbb{P}_q|$ residents within different deadlines d. We used $N = 5, 10, 15$ and $d = 10, 20, 30, 45, 60, 90, 120$ minutes. For all combinations of N and d, we conducted 20 experiments. The robot started in the Games Room at 1:30 pm and searched for residents in $\mathbb{P}_q$ in order to invite them to a Bingo game that started at 4:00 pm. The start times were chosen such that the robot searched for residents in a time frame encompassing cases where residents had appointments, activities and spare time. The time t_0 indicates the time when the query was received.

Search Performance and Runtime

The performance metrics for the comparison are the success rate, the mean search time per person and the number of visited regions during the search procedure. We measure the pre-computation time needed at software start-up to create the MDP model and load the learned behaviour, and set up the three likelihood functions $\mathcal{L}_s$, $\mathcal{L}_l$, and $\mathcal{L}_{env}$. We also measure the computation times for the single plans (including the computation of $\mathcal{L}_{lkrl}$, the rewards, and the policies) which are computed when a query is received and whenever a new plan is generated due to replanning when a person has been found. The metrics are measured for: 1) different values of K, which represents the number of persons for which the plan is generated, and 2) for the different plan execution times, namely the time the robot plans into the future.

Results and Discussion

The comparison results are presented in Figure 5 and Table 2. Figure 5 shows that the proposed M-USB search finds more persons within a given deadline when compared to the other two approaches. All search algorithms have higher success rates for larger d since more residents can be found when more time is allocated to the search. It is interesting to note that the Weighted Informed Walk had comparable success rates to the Random Walk. We suspect that this is due to the time of the day in which the search took place. During portions of the search, some residents had activities in the garden. For the Weighted Informed Walk approach, the garden (which was not considered to be a common room) had a lower weight compared to the common rooms (TV-Room and Games Room) and private rooms. Therefore, these residents were not found in the majority of the searches with $d = 30, 45, 60, 90$ when using the Weighted Informed Walk approach due to the low weight given to the garden, which prevented the robot to visit this region often. However, these residents were found using the M-USB Search approach be-

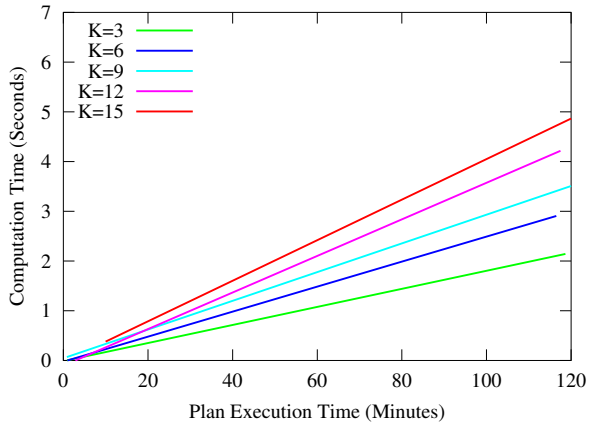


Figure 6: The M-USB Search computation time needed to compute a plan for K residents and a plan execution time.

cause the Schedule Analyzer increased the reward for the garden when one or more searched residents had an activity in this region.

The measured mean search times per person in Table 2 show that our M-USB Search is the fastest search approach. Furthermore, our M-USB Search approach finds a person by visiting the least amount of regions in the environment as also shown in Table 2. The overall results show that the use of the persons' schedules, learned behaviours, the topology of the environment, and the attempt to keep the search order in the proposed approach lowers the mean search times and the number of visited regions, and improves success rate.

The measured mean pre-computation time during system start-up for our approach was 40.51 s. The computation time needed when a query was received can be seen in Figure 6. It is linear for the number of residents (K) for whom the plan has to be generated since $\mathcal{L}_{lkrl}$ and the reward have to be computed K times. The computation time also increases with the plan execution time since the reward and the policy have to be computed for every time step during backwards induction. In general, the computation times are very short (e.g., we measure a mean of 4.8 s for $K = 15$ and a plan execution time of 120 minutes).

4 Conclusion

In this paper we address the problem where a robot needs to search for multiple non-static residents within a retirement home environment. We have developed the M-USB Search planning procedure which generates a high-level-plan to maximize the number of residents that are found within a given time frame. We obtain spatio-temporal likelihood functions for the individual residents using the schedules of the residents, the layout of the retirement home environment as well as direct observations by the robot. The M-USB search method uses a novel approach to compute the reward to determine the robot's search plan for finding multiple persons. We have compared our M-USB search method to a Weighted Informed Walk search and a Random Walk search for the proposed problem. Our results showed that the M-USB Search can find the residents in a shorter amount of time by visiting a fewer number of rooms.

Acknowledgments

This research has been funded by a Natural Sciences and Engineering Research Council of Canada (NSERC) Collaborative Research and Development Grant and by Dr Robot Inc. The first author has been funded by the German Academic Exchange Service (DAAD) PROMOS program.

References

Bath, P. A., and Deeg, D. 2005. Social engagement and health outcomes among older people: introduction to a special section. *European Journal of Ageing* 2(1):24–30.

Centre for Health Workforce Studies. 2006. The Impact of the Aging Population on the Health Workforce in the United States: Summary of Key Findings.

Elinas, P.; Hoey, J.; and Little, J. J. 2003. HOMER: Human Oriented MEssenger Robot. *AAAI Spring Symposium on Human Interaction with Autonomous Systems in Complex Environments, Stanford CA*.

Findlay, R. A. 2003. Interventions to reduce social isolation amongst older people: where is the evidence? *Ageing and Society* 23(5):647–658.

Lau, H.; Huang, S.; and Dissanayake, G. 2005. Optimal search for multiple targets in a built environment. In *IROS*, 3740–3745. IEEE.

Lau, H.; Huang, S.; and Dissanayake, G. 2006. Probabilistic Search for a Moving Target in an Indoor Environment. In *IROS*, 3393–3398. IEEE.

McColl, D.; Louie, W.-Y. G.; and Nejat, G. 2013. Brian 2.1: A Socially Assistive Robot for the Elderly and Cognitively Impaired. *IEEE Robot. Automat. Mag.* 20(1):7483.

Menec, V. H. 2003. The relation between everyday activities and successful aging: A 6-year longitudinal study. *The Journals of Gerontology Series B: Psychological Sciences and Social Sciences* 58(2):S74–S82.

Oida, Y.; Kanoh, M.; Inagaki, M.; Konagaya, Y.; and Kimura, K. 2011. Development of a robot-assisted activity program for elderly people incorporating reading aloud and arithmetic calculation. In *Asian Perspectives and Evidence on Health Promotion and Education*. Springer. 67–77.

PriceWaterCoopers LLP. 2001. Report of a Study to Review Levels of Service and Responses to Need in a Sample of Ontario Long Term Care Facilities and Selected Comparators.

Sharkey, S. 2008. People caring for people impacting the quality of life and care of residents of long-term care homes: a report of the independent review of staffing and care standards for long-term care homes in ontario.

Tipaldi, G. D., and Arras, K. O. 2011. I want my coffee hot! Learning to find people under spatio-temporal constraints. In *ICRA*, 1217–1222. IEEE.

Wilson, R. S.; Krueger, K. R.; Arnold, S. E.; Schneider, J. A.; Kelly, J. F.; Barnes, L. L.; Tang, Y.; and Bennett, D. A. 2007. Loneliness and Risk of Alzheimer Disease. *Arch Gen Psychiatry* 64(2):234–240.