IEEE Transactions on Cybernetics. This is the author's version of an article that has been published. Changes were made to this version by the publisher prior to publication. (https://doi.org/10.1109/TCYB.2022.3166481)

SUPPLEMENTARY INFORMATION FOR A MULTIROBOT PERSON SEARCH SYSTEM FOR FINDING MULTIPLE DYNAMIC USERS IN HUMAN-CENTERED ENVIRONMENTS

Sharaf C. Mohamed, Angus Fung, and Goldie Nejat, *Member, IEEE*

A. EXAMPLE SEARCH SCENARIOS

Figure A represents larger scenario images of our example applications, Fig.1, for which our 2-MRPSS method can be applied. Namely, a floor in: a long-term care home (Fig. A1 (a)); a hospital (Fig. A1 (b)); and an office building (Fig. A1 (c)).

(a) (b) (c)

Fig. A1. Examples of typical scenarios of a floor in a: (a) long-term care home, (b) hospital, and (c) office building. Robots are orange, target people are cyan.

B. LIST OF SYMBOLS

1) Target Users

U'	set of all target users
Z	total number of target users
U'_z	z^{th} target user

2) Regions

R	set of all regions
I	total number of regions
R_i	i^{th} region
$R_{i'}$	i'^{th} region
$t_i^{i'}$	travel time between R_i and $R_{i'}$
β_i	number of cells in R_i

3) Dataset

D_z	dataset containing observations of the daily location patterns for U'_z
Y	number of observation days
ϵ_y	y^{th} observation day
$d_{z,i,y}$	segment of time U'_z occupied R_i during ϵ_y in the dataset D_z

4) Time Periods

T	set of all time periods
Ω	total number of time periods
T_ω	ω^{th} time period
T_j	j^{th} time period
T_k	k^{th} time period

t^{period}	duration of a time period
$\mathbb{P}$	power set of all subsets of time periods
Λ	number of subsets in $\mathbb{P}$
$\mathbb{P}_\lambda$	λ^{th} subset in $\mathbb{P}$
$T_{j,k}$	time window between T_j and T_k

5) Robots

$\mathbb{R}$	set of all robots
B	total number of robots
$\mathbb{R}_b$	b^{th} robot
$R_0^{(b)}$	initial region of $\mathbb{R}_b$
$R_{\omega,0}^{(b)}$	initial region of $\mathbb{R}_b$ during T_ω
$a_{i,\omega,t}$	robot action to search R_i during T_ω for a duration of t
t^{move}	average time for a robot to move between regions

6) Search Query

S	search query
t^{start}	start time of the search
t^{end}	end time of the search
t^{plan}	duration of time allocated to planning

7) Search Plan

TP	team plan
$a_{i,\omega}^*$	team action for searching R_i during T_ω, the asterisk indicates the action belongs to the team
t^{unit}	discrete time increment for search actions
$SP_\omega^{(b)}$	search plan for $\mathbb{R}_b$ during T_ω
$a_{\omega,h}^{(b)}$	h^{th} search action of $SP_\omega^{(b)}$
$R_{\omega,h}^{(b)}$	h^{th} search region of $SP_\omega^{(b)}$
$t_{\omega,h}^{(b)}$	h^{th} search duration of $SP_\omega^{(b)}$
$\mathbb{S}_{i,\omega}$	set of robot actions in TP searching R_i during T_ω

8) Unallocated Actions

UA	unallocated team search actions
UA_ω	unallocated team search actions during T_ω
$UA_{i,\omega}$	unallocated team search actions for R_i during T_ω

9) Min-flow graph

G_ω	minimum flow graph for T_ω
Q_ω	time elapsed in T_ω
$N_{Q_\omega}^{i,\omega}$	decision node for the sequential min-flow graph to select the search duration of R_i in T_ω given that Q_ω time elapsed while searching R_1 to R_{i-1}
$E_{t(a_{i,\omega}^*)}^{i,\omega}$	edge for the sequential min-flow graph corresponding to the decision to search R_i for a duration of $t(a_{i,\omega}^*)$ during T_ω

10) Clusters

FC_ω	set of fuzzy clusters in T_ω
F	number of fuzzy clusters in FC_ω
$FC_{\omega,f}$	f^{th} fuzzy cluster in FC_ω
$FC_{\omega,f'}$	f'^{th} fuzzy cluster in FC_ω
$CT_f^{f'}$	distance between clusters $FC_{\omega,f}$ and $FC_{\omega,f'}$
$\mathbb{R}_{\omega,f}$	robot assigned to $FC_{\omega,f}$
$\rho_{i,\omega,f}$	ownership of cluster $FC_{\omega.f}$ over action $a_{i,\omega}^*$
$\rho_{i,\omega}^+$	amount of ownership over $a_{i,\omega}^*$ transferred
$\rho_{i,\omega,f}^+$	ownership of cluster $FC_{\omega,f}$ over action $a_{i,\omega}^*$ after a transfer
$\overline{FC}_\omega$	highest cost fuzzy cluster in FC_ω
$\dot{FC}_\omega$	fuzzy cluster closest to $\overline{FC}_\omega$ in FC_ω
FS	far set
CS	close set
$\widetilde{a_f}$	action in $FC_{\omega,f}$ closest to CS
$\widetilde{FC_\omega}$	fuzzy clusters in the order they are added to CS
$\widetilde{FC_{\omega,f}}$	f^{th} fuzzy cluster in $\widetilde{FC_\omega}$
$\widetilde{\Psi_\omega}$	cost of ordered cluster set $\widetilde{FC_\omega}$
$\widetilde{\Psi_{\omega,f}}$	cost of cluster $\widetilde{FC_{\omega,f}}$

11) Cluster Plan

$CP_{\omega,f}$	cluster plan
M_f^{CP}	number of actions in $CP_{\omega,f}$
$a_{\omega,g}^{(f),CP}$	g^{th} action in $CP_{\omega,f}$
$R_g^{(f),CP}$	g^{th} region in $CP_{\omega,f}$

12) Local Search

t^{cell}	duration of time for a robot to search a cell
$\zeta_{i,\omega}^{(b)}$	number of cells for $\mathbb{R}_b$ to search in R_i during T_ω
$\gamma_{i,\omega}$	occurrence of a search in R_i during T_ω
θ_z^{UA}	occurrence of the team finding U_z' when performing actions in UA
$\theta_{z,i}^{UA}$	occurrence of the team finding U_z' in R_i when performing actions in UA
$\phi_{z,i,y}^{UA}$	occurrence of the team finding U_z' in R_i when performing actions in UA based on $d_{z,i,y}$
$\phi_{z,i,y,\omega}^{UA}$	occurrence of the team finding U_z' in R_i during T_ω performing actions in UA based on $d_{z,i,y}$
$\xi_{z,i,y,\omega}^{UA}$	the expected time during the search of R_i in T_ω in which U_z' is in R_i based on $d_{z,i,y}$

13) Functions

$P(x)$	probability of occurrence x
$P_z(x)$	probability of occurrence x for U'_z
$\mathbb{S}_\omega(x)$	set of robot-cluster pairs with cost below x
$t(x)$	duration of x
$W(x)$	reward acquired by x
$\Psi(x)$	cost of x
$\Psi^+(x)$	cost of x after transfer
$\|x\|$	cardinality of x

C. User Model

The user location model is extended from [21], in which it was presented for a single robot. Prior to searching the environment, the robot team acquires data on users for Y days. The dataset for each user, $D_z = \{(d_{z,1,1}, \dots, d_{z,1,Y}), \dots, (d_{z,I,1}, \dots, d_{z,I,Y})\}$, contains observations of the location patterns for U'_z. Each observation $d_{z,i,y}$ indicates the time segments that U'_z occupies R_i on observation day ϵ_y.

During a search, user data is used to predict their locations. A user has an equal probability of repeating any day, $P_z(\epsilon_y)$:

$$P_z(\epsilon_y) = \frac{1}{Y}, \forall y \in [1, Y]. \tag{C1}$$

Each observation that occurs during ϵ_y has a probability of occurring, $P(d_{z,i,y})$, equal to the probability of ϵ_y occurring:

$$P(d_{z,i,y}) = P_z(\epsilon_y), \forall z \in [1, Z], i \in [1, I]. \tag{C2}$$

Given an observation $d_{z,i',y'}$ occurs, then all observations from that day occur and all observations from other days do not:

$$P(d_{z,i,y} | d_{z,i',y'}) = \begin{cases} 0, \text{if } y' \neq y \\ 1, \text{if } y' = y \end{cases}, \forall i \in [1, I]. \tag{C3}$$

As each day is mutually exclusive, we obtain the user's unique location patterns demonstrated during the observation day.

In the user location model, we assume the probability of U'_z occupying R_i depends on if U'_z was previously in R_i, but is independent of other regions $R_{i'}$. By capturing the dependence within a region, the location model can determine when a user will revisit a region. Furthermore, the number of probabilities needed by the model is reduced from $\left(\frac{t^{period}}{t^{unit}}\right)^{I\Omega}$ to $\left(\frac{t^{period}}{t^{unit}}\right)^{\Omega}$; e.g., for $I = 30$ regions, $\Omega = 3$ time periods, $t^{period} = 900s$, and $t^{unit} = 15s$, the total number of probabilities to compute is reduced from 1.1×10^{160} to 2.2×10^5. This location model is used to generate rewards for robot search actions that can effectively reason about when to search a region and if needed to search the region multiple times during a search time frame.

Empirical analysis verified that the use of conditional probabilities within the user location model improved the search performance with respect to the mean success rate (the ratio of users found over the target users) compared to methods that assume conditional independence in their user model.

Experiments were also conducted to demonstrate the robustness of the user location model by introducing the following forms of uncertainty [21]: 1) misalignment of user activities with time periods, 2) observational errors during data collection, 3) deviation of user behaviors from their observed data, 4) varying number of observation days, and 5) introduction of detection errors during the search. It was found that when uncertainty led to errors in the users' location probability distributions, uncertainty forms (1)-(3) above, the search planner outperformed both MDP and coverage planners. In fact, misalignment of activities with time periods had little impact on the planner's mean success rate, and the planner had higher mean success rates when observational errors and behavior deviations were less than 75%.

D. Local Search

Our 2-MRPSS presented in the paper is independent of the local planner used, as long as the local planner can provide $P(\theta_{z,i}^{UA})$ as needed in Eqs. (8)-(11) of the paper. Herein, for searching a region R_i in time period T_ω we used a two-stage approach where we first determined an ordered set of cells for the team to visit in the region and then divided the cells amongst the robots. For the first stage, the team selected the ordered cells in the region a single robot would search given the team search time $t(a_{i,\omega}^*)$ using a grid-based coverage local search [21]. For the second stage, the selected ordered set of cells are divided amongst multiple robots such that each robot $\mathbb{R}_b$ searches $\zeta_{i,\omega}^{(b)}$ cells based on its assigned search duration for R_i during T_ω, $t(a_{i,\omega}^{(b)})$:

$$\zeta_{i,\omega}^{(b)} = \frac{t(a_{i,\omega}^{(b)})}{t^{cell}}. \tag{D1}$$

Specifically, $\mathbb{R}_b$ searches cells $(\zeta_{i,\omega}^{(b-1)} + 1)$ to $\zeta_{i,\omega}^{(b)}$ in the ordered set of cells for the team, where $\zeta_{i,\omega}^{(0)}$ is defined as 0.

To determine $P(\theta_{z,i}^{UA})$ for the local search, let $\phi_{z,i,y}^{UA}$ represent the occurrence that U'_z is found in R_i if the robots execute the actions in UA and U'_z is in R_i during the time indicated by observation $d_{z,i,y}$. Then, $\theta_{z,i}^{UA}$ must occur if for any $y \in [1, Y_i]$ the occurrence $\phi_{z,i,y}^{UA}$ and observation $d_{z,i,y}$ both occur:

$$P(\theta_{z,i}^{UA}) = P(\bigcup_{y=1}^{Y_i} d_{z,i,y}, \phi_{z,i,y}^{UA}). \tag{D2}$$

The right hand side of the equation can be simplified as observations $d_{z,i,y}$ and $d_{z,i,y'}$, $\forall y \neq y'$, are mutually exclusive:

$$P(\bigcup_{y=1}^{Y_i} d_{z,i,y}, \phi_{z,i,y}^{UA}) = \sum_{y=1}^{Y_i} P(d_{z,i,y}, \phi_{z,i,y}^{UA}). \tag{D3}$$

To determine $P(d_{z,i,y}, \phi_{z,i,y}^{UA})$, we note that $d_{z,i,y}$ is independent of $\phi_{z,i,y}^{UA}$. Namely, the probability of an observation being uniformly sampled from the data set is independent of the search actions in UA finding the user during that observation:

$$P(d_{z,i,y}, \phi_{z,i,y}^{UA},) = P(d_{z,i,y}) P(\phi_{z,i,y}^{UA}). \tag{D4}$$

$P(d_{z,i,y})$ is obtained using Eq. (C2). To determine $P(\phi_{z,i,y}^{UA})$,

we introduce $\phi^{UA}_{z,i,y,\omega}$ which represents the occurrence of $\phi^{UA}_{z,i,y}$ during T_ω. U'_z is found in R_i during the search if the user is found in R_i during any time period:

$$P\left(\phi^{TP}_{z,i,y}\right) = P\left(\cup^{\Omega}_{\omega=1}\phi^{UA}_{z,i,y,\omega}\right). \quad \text{(D5)}$$

Moreover, we can express the union as a series of intersections:

$$P\left(\cup^{\Omega}_{\omega=1}\phi^{UA}_{z,i,y,\omega}\right) = \sum_{\mathbb{p}_\lambda\in\mathbb{P}(T)} \mathrm{M}(\mathbb{p}_\lambda) P\left(\cap_{T_\omega\in\mathbb{p}_\lambda}\phi^{UA}_{z,i,y,\omega}\right), \quad \text{(D6a)}$$

and

$$\mathrm{M}(\mathbb{p}_\lambda) = \begin{cases} 1, & \text{if } |\mathbb{p}_\lambda| \text{ is odd} \\ -1, & \text{if } |\mathbb{p}_\lambda| \text{ is even} \end{cases}. \quad \text{(D6b)}$$

$\mathbb{p}_\lambda$ is a unique subset of $T = \{T_1, \dots, T_\omega\}$ and $\mathbb{P} = \{\mathbb{p}_1, \dots, \mathbb{p}_\Lambda\}$ is the power set of T. $|\mathbb{p}_\lambda|$ is the cardinality of $\mathbb{p}_\lambda$. Namely, to compute a union, all intersects of odd sized subsets must be added, and all intersect of even sized subsets must be subtracted. As $\phi^{UA}_{z,i,y,\omega}$ and $\phi^{UA}_{z,i,y,\omega'}$ are independent $\forall\omega \neq \omega'$, $P\left(\cap_{T_\omega\in\mathbb{p}}\phi^{UA}_{z,i,y,\omega}\right)$ can be computed as a product:

$$P\left(\cap_{T_\omega\in\mathbb{p}}\phi^{UA}_{z,i,y,\omega}\right) = \prod_{T_\omega\in\mathbb{p}} P\left(\phi^{UA}_{z,i,y,\omega}\right). \quad \text{(D7)}$$

As the user location data only indicates the region a user is in, and not the cell within the region, user locations across the cells have a uniform probability distribution. We introduce $\xi^{UA}_{z,i,y,\omega}$, as the expected time during the search of R_i in T_ω in which U'_z is in R_i based on $d_{z,i,y}$. The probability of finding the user in region R_i can then be expressed as the expected amount of time spent searching R_i while the user is present divided by the time to search the entire region, denoted β_i:

$$P\left(\phi^{UA}_{z,i,y,\omega}\right) = \frac{\xi^{UA}_{z,y,i,\omega}}{\beta_{\mathrm{i}}}. \quad \text{(D8)}$$

As UA does not specify when a search occurs during a time period, each search action can start at any time, τ, during the time period with equal probability. To determine $\xi^{UA}_{z,y,i,\omega}$, we integrate over all possible starting times τ for which the entire search action $a^*_{i,\omega}$ can be completed within T_ω:

$$\xi^{UA}_{z,v,i,\omega} = \int_{-\infty}^{\infty}\int_{0}^{t^{period}-t\left(a^*_{i,\omega}\right)} occ_{z,i,y,\omega}(t) \times rec_{t\left(a^*_{i,\omega}\right),\omega}(t-\tau)d\tau\mathrm{d}t, \quad \text{(D9a)}$$

$$occ_{z,i,y,\omega}(t) = \begin{cases} 1, \text{ if } d_{z,i,y} \text{ indicates } U'_z \text{ is in } R_i \text{ at } t \\ 0, \text{ otherwise} \end{cases}, \quad \text{(D9b)}$$

$$rec_{t\left(a^*_{i,\omega}\right),\omega}(t) = \begin{cases} 1, 0 \le t \le t\left(a^*_{i,\omega}\right) \\ 0, \text{ otherwise} \end{cases}. \quad \text{(D9c)}$$

$occ_{z,i,y,\omega}(t)$ is a binary function which represents the time during T_ω that U'_z is in R_i based on $d_{z,i,y}$, and $rec_{t\left(a^*_{i,\omega}\right),\omega}(t)$ is a rectangular function which represents the duration of the search action $a^*_{i,\omega}$.

E. Complexity Analysis

We provide the time and space complexity for the proposed team action selection and action allocation approach. In particular, for solving 1) the sequential min-flow graph, 2) fuzzy clustering, and 3) the overall 2-MRPSS.

1) Sequential Min-flow Graph

The time complexity of solving the sequential min-flow graph using the Bellman-Ford algorithm is:

$$O(|V||E|), \quad \text{(E1)}$$

where:

$$|V| = I\left(\frac{t^{period}}{t^{unit}}\right), \quad \text{(E2)}$$

$$|E| = |V|(\max\beta_i). \quad \text{(E3)}$$

The space complexity is:

$$O(|V| + |E|). \quad \text{(E4)}$$

2) Fuzzy Clustering

The time complexity of the fuzzy clustering approach is:

$$\mathcal{O}\left(\frac{t^{period}}{t^{unit}}B^2I^2 + log\, B^2 \cdot B^{2.5}\right). \quad \text{(E5)}$$

It represents the summation of the time complexity of generating initial clusters with K-means++ $\mathcal{O}(\mathrm{BI})$; the EM algorithm $\mathcal{O}\left(\frac{t^{period}}{t^{unit}}B^2I^2\right)$; and assigning robots to clusters $\mathcal{O}(\log \mathrm{B}^2 \cdot B^{2.5})$, where for the latter term, $log\, B^2$comes from binary search and $B^{2.5}$ comes from Hopcroft-Karp algorithm. The space complexity is:

$$\mathcal{O}(I^2 + IB + B^2) = \mathcal{O}(I^2), \quad \text{(E6)}$$

where $B \le I$, $I^2$is the region distance matrix, B^2 is the cluster distance matrix, and IB is the region partial ownerships.

3) 2-MRPSS

The time complexity of the combined two-stage approach is:

$$\mathcal{O}\left(\left(\frac{t^{period}}{t^{unit}}\right)^2 I^2 \max\beta_i + \frac{t^{period}}{t^{unit}}B^2I^2 + B^{2.5}\log B^2\right), \quad \text{(E7)}$$

and the space complexity is:

$$\mathcal{O}\left(I\left(\frac{t^{period}}{t^{unit}}\right)(1 + \max\beta_i) + I^2\right). \quad \text{(E8)}$$

F. Alternative Action Allocation Methods for Experiment #1

In the paper, we compare the mean maximum search time (MMST) of our clustering action allocation method with three alternatives methods: 1) naïve, 2) random, and 3) memetic.

Below, we detail the implementation of these alternative methods.

1) Alternative Approach #1: Naïve Allocator

We considered a three-stage naïve approach. The first step generates the order to perform all the team search actions using a single robot approximation. The second step follows the ordering to generate all the individual robot plans. The third step assigns specific robots to each plan. The naïve method is presented in Fig. F1. We implemented this approach as it is a direct extension of the single robot approach in [21].

The first step is to *generate an optimal single robot ordering* for the team to perform all the search actions in UA_ω. This is performed by setting the starting location to $R^{(1)}_{\omega,0}$ and solving the TSP_2 we defined in Section IV.E.1.b of the paper. Next, we *sequentially generate B plans*. Each plan is assigned actions until no additional action can be assigned without the plan duration exceeding $\frac{1}{B}$ of total time to complete the ordering. A plan may be assigned a portion of the search time for an action, in which case the following plan will be assigned the remainder of the search time for that action. To generate TP_ω, we optimally *assign robots* to the plans by solving the LBAP described in Section IV.E.1.d of the paper.

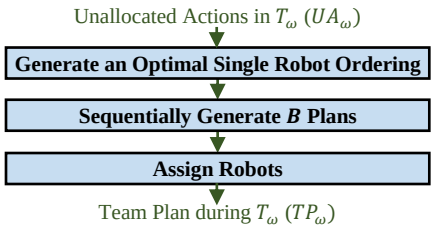

Fig. F1. Flow chart of the naïve allocator.

2) Alternative Approach #2: Random Allocator

The random approach generates a very large number of candidate team plans and selects the team plan with the minimum duration. We implemented this approach as the low computational complexity of randomly generating a plan enables a large percentage of the solution space to be sampled, and thus, a high-performance solution that minimizes the MMST to be selected. The random method is outlined in Fig. F2.

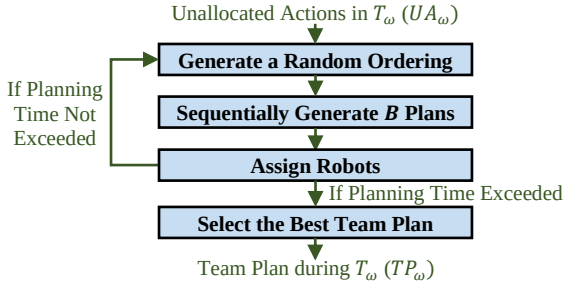

Fig. F2. Flow chart of random allocator.

The first step of the method is to *generate a random ordering* of UA_ω by selecting, with uniform probability, from all permutations of UA_ω. Based on this ordering, we then *sequentially generate B plans* using the same procedure as the naïve method. The LBAP was then solved to *assign robots* to these plans and generate a proposed team plan. This planning process was repeated for the entire planning duration using various permutations of UA_ω. From all the proposed team plans, we *select the best team plan* to be the one which minimizes Eq. (10) in the paper, and this plan is executed by the team.

3) Alternative Approach #3: Memetic Allocator

The memetic allocator used is adapted from the memetic algorithm based on sequential variable neighborhood descent (MASVND) presented in [38] as it is the state-of-the-art for solving the min-max mTSP in real-time. The memetic approach works by first generating a set of random team plans and then modifying or combining these plans to generate new team plans as shown in Fig. F3.

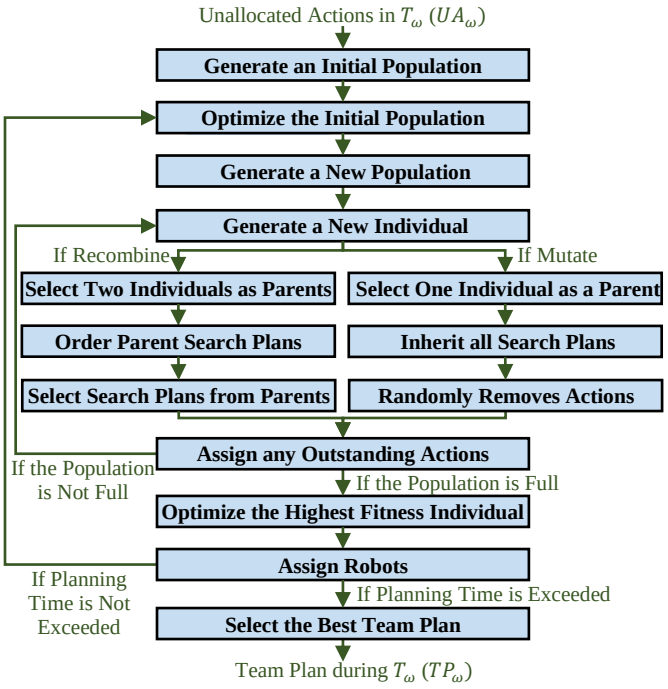

Fig. F3. Flow chart of memetic allocator.

The first step of the memetic approach is to *generate an initial population*. A population consists of 100 individuals and each individual represents a candidate team plan. Each initial individual was generated from UA_ω using the random team search action allocator discussed above, without the robot assignments. The second step is to *optimize the initial population* by solving TSP_1 from Section IV.E.1.b for all robot search plans. A measure of fitness is assigned to each individual based on the inverse of its total duration, where duration is determined based on Eq. (10). The next step is to *generate a new population* and iteratively *generate a new individual* using the recombine or mutate operator to fill the population. The recombine operation is selected with a 30% probability and the mutate operation is selected otherwise.

To perform the recombine operation, the first step is to *select two individuals as parents*. Each parent is determined by randomly selecting two individuals from the current population. The higher fitness individual is assigned as a parent with 80% probability and the other individual is assigned otherwise. After

selecting two parents, we *order parent search plans* from shortest to longest and *select search plans from parents*. With equal probability, the smallest cost plan from one of the parents is added to the new individual, and the other parent's smallest cost plan is discarded. This process is then repeated until the new individual has B plans, one for each robot. After each repetition, the actions in the selected plan are removed from all other plans to avoid duplicate actions in the new individual.

To perform the mutate operation, the first step is to *select one individual as a parent.* The parent is determined as the highest fitness individual amongst three randomly selected individuals from the current population. The new individual proceeds to *inherit all search plans* from the parent. Then the individual *randomly removes actions* from the inherited search plans. Each action is removed with a probability that linearly decreases from 85% at the start of the planning duration to 10% at the end.

After using either the recombine or mutate operator, some actions in UA_ω may be unassigned. Thus, the subsequent step is to *assign any outstanding actions.* This assignment process is performed by checking the effect of adding the search action at every position of all plans. The position-plan pair that minimizes the increase to the selected plan length, without exceeding the longest plan, is selected as the destination for the search action. If all position-plan pairs result in the selected plan length exceeding the longest plan, the position-plan pair that minimizes the new longest plan is selected.

After all 100 individuals in the new population are generated, we *optimize the highest fitness individual* using sequential variable neighborhood descent (seq-VND) [38]. Seq-VND modifies the individual's current team plan by attempting to move a sequence of one to three consecutive actions from any search plan to any position in any other search plan. The sequence is placed in the first position found that would reduce the individual's fitness. Sequences are iteratively moved until the fitness cannot be reduced.

By setting the newly generated population as the current population, the process of generating new populations is continued for the remainder of the planning duration. Proposed team plans are generated by solving the LBAP to *assign robots* for the highest fitness individual in each population. We then *select the best team plan* as the proposed team plan which minimizes Eq. (10) to output as TP_ω.

G. Validation Test for Planning Duration

As the team search action allocation module has the highest computational complexity, we validated that our clustering action allocation method and the three alternative allocation methods could plan in real-time.

Search queries: we considered a small subset of the experiment search queries in order to determine a planning duration for our comparison. This included all combination of: *environment size* = {30, 36, 42} shared rooms, *search duration* = {15, 45, 75} minutes, *number of target users* = {1, 10, 20}, *planning duration* = {1, 10, 100} milliseconds, and *search start time* = {10:00, 14:00, 18:00} on a 24-h clock for a total of 243 trials. We conducted this comparison for the case of 9 robots as these scenarios have the highest computational complexity. The MMST versus planning time is presented in Fig. G1, where each point is an average across 81 trials.

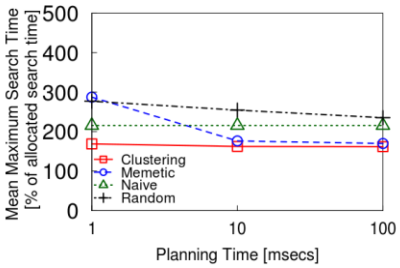

Fig. G1. Mean maximum search time with respect to planning duration.

For each of the three methods, we conducted a non-parametric Kruskal-Wallis test (α=0.05) with a Bonferroni correction (α=0.0125) to determine if there were any statistically significant differences in MMST across the three planning durations. No statistically significant differences were found in the MMST across planning durations for our clustering allocator, $\chi^2(2) = 0.79, p = 0.67$; the naïve allocator, $\chi^2(2) = 0.00, p = 1.00$; and the random allocator, $\chi^2(2) = 6.48, p = 0.039$. However, a statistically significant difference existed for the memetic allocator, $\chi^2(2) = 35.4, p < 0.0001$. A post-hoc Dunn's test (α=0.05) with a Bonferroni correction (α=0.0167) showed that for the memetic allocator a statistically significant difference existed between the MMST of 1 ms and 10 ms, $Z(81) = 5.26, p < 0.0001$, and 1 ms and 100 ms, $Z(81) = 5.77, p < 0.0001$, but not between 10 ms and 100 ms $Z(81) = 0.73, p = 0.233$. Therefore, providing more than 10 ms of planning time does not result in any additional benefit in reducing the MMST for any allocator. This analysis validates that all four methods could plan in real-time as their performance was maximized with only 10 ms of planning time and that 10 ms was sufficient for the experimental comparisons in the paper.

H. Alternative Multirobot Person Search Systems For Experiment #2

In the paper, we compare the mean success rate of our aware 2-MRPSS with strong coordination with respect to two alternatives methods: 1) a segmented approach (unaware with no coordination), and 2) a sequential approach (aware with weak coordination). Below, we detail the implementation of both alternatives.

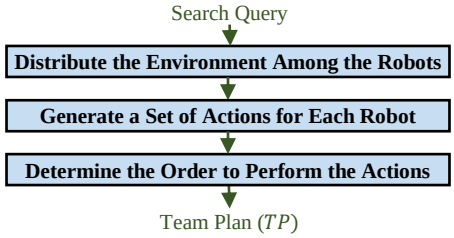

Fig. H1. Flow chart of the unaware approach.

1) Alternative Approach #1: Segmented Approach

The segmented approach is based on the method in [14], where an environment is segmented into an area for each robot to search. Although the robots are unaware of each other and

do not coordinate, this approach prevents redundant actions between the robots while minimizing travel time. We have integrated this with our two-stage approach, Fig. H1.

The first step is to *distribute the environment among the robots*. This is achieved by determining the total number of cells in the environment, Γ, using the grid representation of the regions from the local search method presented in Part B above. In [14], the environment was an open field and was therefore easily divided into rectangles of equal size for each robot. As our environment has walls between regions, we use an ad-hoc technique to generate a unique segmentation of each scenario environment that assigns an area to each robot such that the maximum travel time between any pair of regions in an area is minimized. The ad-hoc technique first assigns an ordering to all cells in the environment and then sequentially assigns each robot a contiguous set of cells from that ordering. The number of cells assigned to each robot was selected to be as close to uniform as possible using the following procedure. Let v be the maximum number of cells each robot can search in a uniformly distributed workload and κ be the cells that remain unsearched:

$$v + \frac{\kappa}{B} = \frac{\Gamma}{B}, v \in \mathbb{Z}, \kappa \in \mathbb{Z}, 0 \leq \frac{\kappa}{B} < 1. \quad \text{(H1)}$$

Then, the first κ robots have $v + 1$ cells and the rest have v cells. The second step is to *generate a set of actions for each robot* by solving the CMPMKP in Section IV.D of the paper to generate UA for a single robot team, e.g. $B = 1$, in an environment containing only the assigned cells. Finally, to generate a search plan for each robot we *determine the order to perform the actions* by solving TSP_2 for each robot in each time period. The culmination of the robot search plans is the team plan executed by the team.

2) Alternative Approach #2: Sequential Approach

The sequential approach is based on the approach presented in [17], where each robot plans in a sequence while considering the actions of the robots that have already generated their plans. This sequential approach prevents redundant actions between the robots. The approach is also combined with our two-stage approach, Fig. H2.

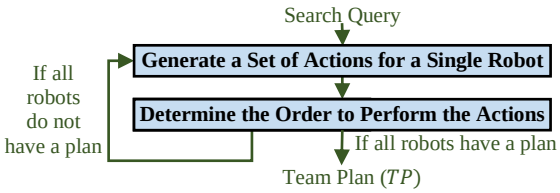

Fig. H2. Flow chart of the weakly coordinated approach.

For the sequential approach, we start by considering the first robot and *generate a set of actions for a single robot* by solving the CMPMKP in Section IV.D of the paper for a team of $B = 1$ robot. Then, to generate a search plan for this robot, we *determine the order to perform the actions* by solving TSP_2 in each time period. Each subsequent robot generates its plan similarly to the first, however it updates the edge reward as in Eq. (9) of the paper, which are used to solve the CMPMKP, to account for the actions performed by robots which had already planned. To achieve this, the edge rewards are updated as follows:

$$W\left(E^i_{t(a^*_{i,\omega})}\right) = -\sum_{z=1}^{Z} P\left(\bigcup_{\omega'=1}^{\Omega} \theta_{z,i}^{AA_{i,\omega'}}\right) + \sum_{z=1}^{Z} P\left(\bigcup_{\omega' \in [1,\Omega]\setminus\omega} \theta_{z,i}^{AA_{i,\omega'}}\right) + \sum_{z=1}^{Z} P\left(\theta_{z,i}^{PA_{i,\omega}}\right). \quad \text{(H2)}$$

$AA_{i,\omega'}$ is a set containing actions $UA_{i,\omega'}$ planned by the current robot and actions $PA_{i,\omega'}$ planned by other robots. As such, the team search time $tq(a^*_{i,\omega})$ in $AA_{i,\omega'}$ is equal to the sum of the team search times $t(a^*_{i,\omega})$ in $UA_{i,\omega'}$ and $q(a^*_{i,\omega})$ in $PA_{i,\omega'}$:

$$tq(a^*_{i,\omega}) = t(a^*_{i,\omega}) + q(a^*_{i,\omega}). \quad \text{(H3)}$$

Once each robot has generated a search plan using the modified rewards, Eq. (H3), the resulting set of search plans form the team plan and are executed by the robots.