Robot Rostering: Coalition Formation for Long-Term Missions with Work Shifts

Eduardo Feo Flushing[1], Luca M. Gambardella[1], Gianni A. Di Caro[2]

***Abstract*— The mission time for a large multi-agent team including multiple robots, as well as human agents or even animals, can easily extend over long time spans, from hours to weeks. For instance, this can be the case of post-disaster missions. Since, for a number of different reasons, not all the agents can be operational 24h/24h, it is necessary to organize the workforce in time shifts. While, in general, this approach is common in personnel management, it was not considered before taking into account a team including multiple robots. In this work, we first introduce the *robot rostering problem* and formalize it using an integer programming model. The specific characteristics and challenges of the problem are discussed. A number of sampling-based constructive heuristics are proposed to deal with the computational intractability of the problem and are studied in extensive simulation experiments.**

I. Introduction

In a typical search and rescue or post-disaster scenario, it is expected that the operational team (including humans, multiple robots, dogs, etc.) operates over *extended temporal scales*, that can be of hours, days, or even weeks or months. Increasing the temporal scale of multi-agent missions raises up the issue of defining the schedules for shift work. Since no agent can keep working 24h/24h (e.g., because of battery limitations, because of getting tired, because of environmental changes, because of work regulations), it is necessary to divide the whole mission time into *shifts*, periods of time during which different subsets of agents are operational.

This approach is common in Operations Research (OR) for *personnel management* [1], [2], especially in healthcare [3] and sustained industrial production, and goes under the category of "rostering problems". Up to our knowledge, similar considerations have never been applied before to heterogeneous multi-robot teams. Therefore, here we first introduce the term *robot rostering problem* (RRP) to refer to the scheduling problems including multi-robot agents when work shifts have to considered.

We have modeled the RRP using an *Integer Programming (IP) formulation*. Binary decision variables are used to select the composition of the individual rosters given a list of shifts to be performed. The use of each agent/robot is associated to limitations and costs. A number of *rostering constraints* are used to model restriction selections both at the level of single shifts and across the shifts. Each shift has so-called *coverage constraints*, that model the service requirements of a shift. These are based on the total budget allocated to the mission and on the expected efficiency of a roster, intended as the ratio between *utility* (performing missions' tasks) and *costs* of a roster. In particular, since both budget allocation and efficiency per shift are usually defined by a strategic layer, in our model we allow the flexible definition of different *budget and efficiency profiles*. For instance, when looking for survivors after a disaster, it could make sense to spend more of the effort in the early stages of the mission, that would result in allocating mode budget and requiring more efficiency in the early shifts, and then progressively decreasing the expenditure. Instead, in the case of a mission of environmental monitoring, it could make sense to have a flat allocation of the budget over the shifts for the prescribed time horizon of the mission.

The objective of solving the IP model consists in finding for each shift a feasible roster that maximizes the expected output (completion of mission's tasks) while being compliant with shifts budget and efficiency requirements. In order to provide a practical ground to our work while not losing generality, we consider the use of a heterogeneous team for tackling a mission that is composed of a set of spatially distributed tasks that can be dealt by incrementally over an extended time period (the *Spatial Task Allocation and Scheduling Problem in Heterogeneous mobile Multi-Robot* teams (STASP-HMR) described in Section III). For instance, in a search and rescue mission, each task would correspond to a patch of the mission area to be inspected to look for survivors. For each shift, based on the selected roster, a system-level multi-agent planner solves the STASP-HMR instance for the *mission state* corresponding to the shift. Following the directives of the planner, each agent executes its plan, possibly with some deviations, depending on the nature of the tasks and of the environment. In practice, this means that to find a solution to the RRP, an instance of the STASP-HMR needs to be solved for each candidate roster in a shift. Moreover, the mission state needs to be updated at the end of each roster to account for potential deviations during execution, a scenario that has to be expected when using robots. This sets RRP far apart from the personnel management problems commonly considered in the OR literature, in which all information about the shifts is available beforehand and the service demand of a shift does not depend on previous rosterings. In our case, the mission state, and shift's service demands, change from shift to shift, depending on the previous rosterings and their executions.

[1] Eduardo Feo Flushing and Luca M. Gambardella are with the Dalle Molle Institute for Artificial Intelligence (IDSIA), Lugano, Switzerland, {eduardo,luca}@idsia.ch

[2] Gianni Di Caro is with the Department of Computer Science, Carnegie Mellon University (CMU), gdicaro@cmu.edu.

This work was partially supported by the Swiss National Science Foundation (SNSF) through the National Centre of Competence in Research (NCCR) Robotics.

Overall, these characteristics, make the RRP problem computationally intractable, preventing from solving it to optimality. To cope with this, we have used an *iterative sampling-based approach* [4] to *incrementally construct feasible solutions*. A *heuristic function* is used instead of the STASP-HMR planner, allowing a rapid estimate of the goodness of a roster. Considering realistically sized missions for a 2-weeks time duration for a total of 56 shifts, we have performed a number of experiments in simulation, to study the computational feasibility of the approach for solving the RRP, the impact of using different heuristic strategies and parameters, and the application of different profiles for budget and costs.

In summary, the *contributions of the paper* include the following: (i) Introduction and formalization of the robot rostering problem; (ii) discussion of the challenges that result from including work shifts into multi-robot missions; (iii) application of the robot rostering problem in the context of search and rescue missions, including operational constraints derived from the OR literature and adapted to real-world, long-term, multi-robot mission scenarios; (iv) definition and evaluation of multiple constructive heuristics dealing with the computational issues of the robot rostering problem.

The rest of the paper is organized as follows. In Section II we introduce the basic concepts and notation that is used to formulate the robot rostering problem. In Section III we present the mission planning model for the STASP-HMR problem, on top of which the rostering problem is defined. In Section IV we formally present the robot rostering problem and the heuristic methods that we use as a first attempt to solve it. In Section V an experimental analysis of the solution methods with realistically sized mission scenarios is reported. Finally, we draw conclusions in section VI.

II. Basic concepts and notation

In this section we introduce the concepts of the robot rostering problem. We also present the notation used in our model formulation.

A. Time horizon and shifts

We assume that the mission has a maximum duration of H days, and that each of these days is composed of M non-overlapping time periods or shifts (e.g., morning, afternoon, night). Thus, there are a total of $H \times M$ shifts during which the agents can be scheduled to work on the mission. We use $d \in \{1, \ldots, H\}$ to denote each day, and $t \in \{1, \ldots, M\}$ to denote the position of a shift within a day.

Shifts occur in sequence, starting from day 1. We use $s \in \{1, \ldots, H \times M\}$ to denote the position of a shift within the mission time span. Using this notation, we can refer to a specific shift by both its absolute position s within the mission time span , or by the day and the position within the day (d, t).

B. Rosters and agents

We assume that the set of agents $\mathcal{A}$ from which agents are selected and assigned to any shift of the mission is known beforehand. A *roster* is a mapping which assigns to each shift s a subset of $\mathcal{A}$ that represents the team that has been selected to work on the mission during the shift s. We denote by $\mathcal{R}$ the set of all possible rosters that can be considered for the specific mission. Note that $\mathcal{R} \subseteq \mathcal{P}(\mathcal{A})^{HM}$, where $\mathcal{P}(A)$ is the power set of $\mathcal{A}$. For a given roster $r \in \mathcal{R}$, $r_s, r_{dt} \subseteq \mathcal{A}$ denote the team assigned to a specific shift s or (d, t).

C. Utility, cost, and efficiency of a roster

The goal of a roster is to work on a single mission in an incremental manner. Therefore we assume that the progress in the mission along the timeline is measured by the amount of work done, also called the *utility* achieved. For a specific roster r, we let $\mathcal{U}_{rs}$ to denote the utility achieved by roster r during a specific shift s.

Furthermore We consider that the inclusion of an agent a into a specific shift s incurs a certain positive cost $c_{as} > 0$. Given this, we can also estimate the *cost* of a given roster r for a specific shift, which is denoted by $\mathcal{C}_{rs} = \sum_{a \in r_s} c_{as}$.

Using the concepts of utility and cost we define the *efficiency of a roster* during a specific shift as the utility-cost ratio: $\frac{\mathcal{U}_{cs}}{\mathcal{C}_{cs}}$. For sake of formulation, we assume that when $\mathcal{C}_{rs} = 0$ (i.e., $r_s = \emptyset$), then the efficiency is 0.

D. Rostering and budget constraints

Personnel rostering problems are typically subject to a variety of constraints which can be broadly categorized as *domain*, *teaming* or *time-related* constraints [1].

Domain and teaming constraints define the possible team assignments for each shift on each day. Domain constraints are related to single agents and can be used to model skills, thus enforcing an agent to be assigned to a shift for which it is qualified (e.g., robots that rely on vision for navigation and cannot work at night time). Teaming constraints define inter-dependencies inside each shift. They can be used to model synergies that are required to enable the agents to cope with the specific shifts. Without losing generality, in this work we do not consider domain nor teaming type of constraints. However, we remark that both domain and teaming constraints can be easily accommodated within our model by explicitly defining the set of agents that are available at each shift.

Time-related constraints are the most common in personnel rostering problems and they can be used to limit the number of appearances of a specific agent or a combination of agents within a certain period (e.g., during a day or the entire mission), or over consecutive shifts. We consider that time-related constraints are the most relevant ones to the applications of interest. Here we consider four types of time-related constraints, all of which relate to a single agent a:

- maximum number of shifts in which the agent can participate in, during the entire mission (N_a);
- maximum number of shifts that the agent can perform in a single day (N_a^d);
- maximum number of consecutive shifts that the agent can perform (M_a);

- minimum number of shifts that the agent should be *out of* roster (m_a).

These constraints are motivated by the limitations of real-world missions involving multi-robots systems. For instance, the values of M_a^d and M_a can be determined by the battery lifetime of a mobile robot. The value of N_a can relate to the maximum operating time of a robot before it must undergo a maintenance phase that would exclude it from the rest of the mission. The value of m_a can be related to the minimum time that is required for a robot to recharge, or for a human to rest.

Lastly, we introduce constraints that limit the cost of a single shift in the roster. In this way it is possible to consider scenarios where the budget is defined per shift basis. We let $\mathcal{B}_s$ be the budget available for shift s.

E. Coverage constraints

Most of the problems seen in the personnel rostering field attempt to cover certain work demand in each period of the planning horizon. Such demand can be specified as required number of agents or set of skills, and can also be time-dependent. These requirements are enforced through the so called *coverage constraints*. These types of constraints are aligned with the typical applications that they consider such as the nurse rostering where shifts are meant to tackle cyclic tasks whose demand and requirements are known beforehand. Therefore, they are often used to define the number of personnel of each category needed to staff the ward.

In our work we consider a novel type of coverage constraints that are related to both budget and efficiency. In practice we set what is the expected service by satisfying at the same time budget limitations. At the strategic layer, we allow to express a desired efficiency profile that specifies, for each shift, the required minimum efficiency that a roster must provide. This approach is flexible enough to model missions where the required efficiency is time-dependent. For instance, in search and rescue missions, efficiency matters can be deferred to the final stage of the mission. To this end, we let Γ_s be the minimum efficiency required for a roster during shift s.

III. Spatial task allocation & scheduling in heterogeneous teams

In this section, we provide a brief description of the *spatial task allocation and scheduling problem in heterogeneous mobile multi-robot teams*, or STASP-HMR in short. This problem has been formally introduced in our previous work [5], [6], with a specific application to search and rescue mission [7].

We consider a mission that has been decomposed into a set $\mathcal{T}$ of *spatially distributed, location-dependent tasks*. Let $\mathcal{A}$ be a team of heterogeneous mobile agents that is available to perform the tasks. Individual tasks are independent from each other, and the success of the mission depends on how which tasks are selected and on the quality each task is performed, as it is explained in the following.

From the point of view of the mission, the complete execution of each task provides an overall *utility*. The total amount of utility which is provided by a task depends on how important the task is for the mission. For instance , in a search and rescue mission, some locations might be more important to search accurately than others based on some a priori knowledge.

The spatial layout of tasks is captured by a *traversability graph* that defines how agents can move between tasks. In the general case, graph is complete and, in this way, we are assuming that all tasks are independent from each other. However, the traversability graph can be used to accommodate constraints over the sequences of tasks that can be executed. For instance, when some tasks cannot be designated immediately after others, e.g., due to mobility constraints, or when specific tasks must be serviced immediately before servicing others, e.g., unblocking a road to reach other parts of the area.

Since we deal with heterogeneous teams, different agents may demonstrate different levels of performance in accomplishing the same task, due to their potentially different skills in relation to the specific local characteristics of the task. This difference in performance among the agents is modeled through the *task model* $\varphi : \mathcal{A} \times \mathcal{T} \mapsto \mathbb{R}$, which relates the amount of effort (measured as service time) to the progress in completion of each agent-task pair.

In the STASP-HMR, tasks are allowed to be non-atomic. Remarkably, while atomic tasks are assumed to be completed once they are assigned, non-atomic tasks can be carried out incrementally. Particularly, in a staged planning scenario (e.g., the mission is carried out in shifts, and plans are computed in an iterative manner), at every stage the model needs to identify the *current completion level* of tasks whose workload has been partially addressed in the previous stages. To this end, we introduce a *completion map* $C_m : \mathcal{T} \mapsto [0, 1]$ which expresses the remaining workload that each one of the tasks $\tau \in \mathcal{T}$ still require. For instance, a value of 0 for a task τ indicates that τ has been completed and, therefore, no further effort from the agents is required. If an agent is assigned to further deal with τ, no additional utility is achieved, which amounts to a waste of time and resources.

Based on the above notions and specifications, the STASP-HMR can be stated as follows. Given a set of heterogeneous agents, each characterized by its task performance model, a set of assignable tasks and a traversability graph, and a given *limited time budget* T, the STASP-HMR consists in determining joint plans for the activities of the agents in the environment which enable the timely selection of the tasks to perform, and aiming to maximize the overall mission utility (i.e., the sum of all gathered rewards). A *solution to the STASP-HMR* (i.e., a mission plan) consists of time-constrained sequences of task, one for each agent, that define how much effort (i.e., devoted time) each of the selected tasks will receive. An optimal solution defines plans for each one of the agents that maximize the mission utility.

Under a linearity assumption of the task models (φ) the STASP-HMR can be formulated by means of a mixed-integer

linear program (MIP).We followed this approach in [5], and proposed a solution method based on the combination of standard solvers and ad hoc heuristics that was able to retain *anytime* properties and formal guarantees on solution quality while speeding up computations.

Yet, these methods can be time-consuming and involve a significant amount of computation. Alternatively, pure heuristic methods can be used to find sub-optimal solutions in an efficient way. In this work we use an *incremental greedy heuristic* that, one agent at a time, incrementally builds a plan by selecting each time the task that provides the maximal utility among the possible moves.

IV. Rostering model

Using the concepts and notation introduced before, we provide a general mathematical model for the *robot rostering problem*. The decision variables x_r indicates that roster $r \in \mathcal{R}$ is chosen or not. We also introduce binary parameters p^r_{adt} and p^r_{as} that take value 1 if the agent a is included in a roster r inside shift (d,t) or s respectively.

The objective function maximizes the workload achieved during the mission time span:

$$\text{maximize} \quad \sum_{r\in\mathcal{R}} \sum_{1\le s\le S} x_r \mathcal{U}_{rs} \tag{1}$$

subject to

$$\sum_r x_r = 1 \tag{2}$$

$$\sum_{r,t} x_r p^r_{adt} \le N^d_a \qquad a \in \mathcal{A}, d \tag{3}$$

$$\sum_{r,s} x_r p^r_{as} \le N_a \qquad a \in \mathcal{A} \tag{4}$$

$$\sum_r x_r \sum_{s\le S-M_a} \sum_{s\le s'\le s+M_a} p^r_{as'} \le M_a \qquad a \in \mathcal{A} \tag{5}$$

$$\sum_{r,2\le s\le S-m_a} x_r (p^r_{a,s-1} - p^r_{a,s-1} p^r_{a,s})(m_a - \sum_{s\le s'\le s+m_a} p^r_{as'}) \le 0 \qquad a \in \mathcal{A} \tag{6}$$

$$x_r \mathcal{C}_{rs} \le \mathcal{B}_s \qquad r \in \mathcal{R}, s \tag{7}$$

$$x_r \Gamma_s \mathcal{C}_{rs} \le \mathcal{U}_{rs} \qquad r \in \mathcal{R}, s \tag{8}$$

$$x_r \in \{0,1\} \qquad r \in \mathcal{R} \tag{9}$$

Constraint (2) ensures that only one roster is selected. Constraints (3) and (4) ensure that a roster respects the maximum number of shifts for each agent. Constraints (5) ensure the limit on the maximum number of consecutive shifts that an agent performs. Constraints (6) ensure that each agent, once idle, remains in that state for a minimum number of shifts. Constraints (7) sets the budget restrictions for each shift. The efficiency of a roster is controlled by (8). Lastly, (9) set the integer requirements on the variables.

The solution to the integer programming formulation presented in (1)- (9) is a feasible roster that maximizes the expected output (tasks completion) while being compliant with each shifts' budget and efficiency requirements. In practice, this means that to find a solution to the RRP, the multi-agent planner has to be called for each candidate coalition in a shift, and for updating the problem state at the end of each shift. This is computationally intractable, and makes the de facto unfeasible to solve our RRP to optimality.

A. Rostering for the STASP-HMR

One of the distinctive aspects of our problem regards the need of estimating the workload achieved by a roster. While in typical rostering problems the objective is to cover certain work demand within each shift, in the case of extended multi-robot missions one of the objectives is to precisely maximize the workload done.

One issue that complicates the rostering problem for the STASP-HMR is that in order to estimate how much of the mission a specific team could do during shift s we should know how much of the mission is left to be done by that time. In other words, the estimation of the amount of work that can be done in a shift not only depends on the skills of the agents assigned to that shift but also on the state of the mission at the beginning of the shift (e.g., remaining tasks). the latter in turn depends on the performance of teams assigned to previous shifts.

To further illustrate this issue, let us consider a search and rescue mission where, among many types of agents (e.g., ground, human rescuers, flying robots), unmanned aerial vehicles (UAVs) are used to detect victims on the ground using downward-facing video cameras. Let us assume that we are preparing a roster and we face the decision of whether or not to include a some of the UAVs on later shifts (e.g., on the next day). Deciding the inclusion of UAVs in those shifts should consider the amount of the sparsely vegetated areas that will still require visual inspection, from the air, by that time. If we ignore this aspect, we could end up in a situation where the still unexplored region has no places that can be effectively searched from air (e.g., the unexplored region is densely vegetated). Therefore, if we would have scheduled the participation of UAVs in these later shifts they would not be able to provide any added value to the mission and therefore degrade the efficiency of the roster.

Heuristic estimation for the STASP-HMR: The estimation of the workload that could be achieved by a roster is based on the best use of the resources allocated to each shift with respect the expected state of the mission at that time. In other words, we must solve an instance of the STASP-HMR for each one of the shifts, using the mission state resulting from the previous shift. The STASP-HMR, which we use as a case study, is part of the NP-hard class of routing and scheduling problems [5]. Therefore it is computationally intractable to obtain an optimal mission plan for the given shift. Hence, we the heuristic method described in Section III to compute a mission plan that allows to estimate the performance of a team for a single shift. In general, most of the resource allocation problems in multi-robot missions are intractable (e.g., [8]), and we propose the heuristic estimation approach as a way to deal with the computational issues inherent to these problems.

Henceforth, we consider the heuristic method as a black-box function $H : \mathcal{R} \times S \mapsto \mathbb{R}$ that estimates the utility of a roster during a specific shift. The computation of $H(r,s)$ is based on the condition that the state of the mission at the start of shift s is available. This condition holds as long as

$H(r,1),\ldots,H(r,s-1)$ have been previously computed.

B. Sampling-based constructive heuristics for RRP

The dependency of the objective function on the mission state prevents the solution of the RRP in an analytic way. Furthermore, it is impractical to use of exact methods based on exhaustive search due to the combinatorial structure of the problem. We also remark that the evaluation of the objective and of the constraints (8) is based upon the heuristic procedure $H(r,s)$, which in practice may involve a significant computation time. Other approaches such as embedding the STASP-HMR in the rostering formulation, would be computationally very expensive.

In this work we introduce the first attempts to solve the rostering problem for STASP-HMR based upon constructive heuristic methods. The heuristics use a sampling-based strategy to construct feasible and efficient rosters. We propose two constructive methods, the *single pass shift-sampling* (SPSS) and the *clustered shift-sampling* (CSS).

Constructive heuristics are widely useful for solving combinatorial optimization problems. Apart for their simplicity, they can also be combined or be integrated in more complex solution methods. For instance, constructive heuristics can be used to initialize a pool of solutions in population-based heuristics such as Genetic Algorithms, or as initial solutions of local search methods.

1) Single pass shift-sampling: The SPSS heuristic constructs a roster by selecting a team for each shift, one shift at the time, starting from 1 up to S. Since the SPSS is a single pass heuristic, it does no backtrack to search for feasible rosters, therefore it may fail when rosters are difficult to construct. The algorithm is described in Algorithm 1. The main steps of the algorithm are explained in the following.

First, when working on a shift s, the SPSS determines a set of agents $\mathcal{A}_s^{VALID}$ that can b assigned to s while respecting the rostering and budget constraints. We note that in order to construct a roster that respects the rostering constraints described in Section II-D, is it sufficient to check these constraints at any single shift, and looking at all the previous shifts that have been already filled in. The rostering constraints are independent of each other, hence, we can check their compliance one by one. In addition, each rostering constraint relates to one single agent, therefore we can include a specific agent into the current shift if it complies with all the rostering constraints that relate to it.

The next step of SPSS when working on a shift s is to *sample* the set of configurations that can be assigned to s. This step is represented by the procedure $MakeSamples$ in Algorithm 1 and consists of sampling a set $\mathcal{A}_s^{SAMP}$ from the power set of the valid agents $\mathcal{A}_s^{VALID}$. We remark that we always include the empty shift $\emptyset$ in $\mathcal{A}_s^{SAMP}$, i.e., it is always possible to leave a shift empty. Under these conditions, the heuristic guarantees the generation of rosters that always comply with the rostering constraints.

In the evaluation we consider a random sample approach where each possible configuration has the same probability of being selected. Since the cardinality of $\mathcal{A}_s^{SAMP}$ has an impact on the performance of the heuristic, we set a limit on the size of $\mathcal{A}_s^{SAMP}$. In the evaluation we study the effect of this parameter on the quality of the solutions provided.

Algorithm: Single-pass shift-sampling

Input: number of samples n
Initialize $roster$ with empty shifts
foreach $1 \leq s \leq S$ **do**
  Determine $\mathcal{A}_s^{VALID}$ based on the current $roster$
  $\mathcal{A}_s^{SAMP}$ = `MakeSamples(`$\mathcal{A}_s^{VALID}$`,n)`
  foreach $team \in \mathcal{A}_s^{SAMP}$ **do**
    Temporary assign $team$ to shift s of $roster$
    Evaluate $H(roster, s)$
    Determine efficiency and validity of $roster$ until shift s
  end
  $team^*$ = Select a *valid* team in $\mathcal{A}_s^{SAMP}$
  Assign $team^*$ to shift s of $roster$
end
return $roster$

Algorithm 1: Pseudo-code of the single-pass shift-sampling heuristic

After the definition of $\mathcal{A}_s^{SAMP}$, the algorithm estimates the performance of each $team \in \mathcal{A}_s^{SAMP}$. For each $team$, it proceeds with the estimation of the utility that would be obtained, during the current shift s, given the inclusion of the $team$ in the roster. This step creates a temporary roster that extends the current roster with $team$ allocated to s. Then, it applies $H(roster, s)$ to estimate the utility and the efficiency resulting from the inclusion of $team$ into the roster.

After the assessment of all samples in the set $\mathcal{A}_s^{SAMP}$, the heuristic selects one of these sample configurations, and fixes it into the roster. In this step, the heuristic also checks the compliance of the coverage constraints (i.e., regarding the efficiency). Given that the utilities of each sample configuration have already been estimated, we combined with their cost, and finally use these two values to determine their efficiency. Only teams that comply with the coverage constraints are selected. When no teams can be found to comply with the coverage constraints, then the shift is left empty.

The selection step can use different criteria. For instance, it can perform a greedy selection, thus selecting the team that maximizes the objective function 1. However, doing so may exhaust the resources and force the heuristic to leave empty many shifts afterwards. On the contrary, selecting a team that provides low utility (but still complying with the coverage constraint), may result into a roster with poor performance. As the definition of this step is non-trivial, in the evaluation we compare different techniques to perform this step. Finally, the selected team is added into the roster and the SPSS proceeds with the next shift in the sequence.

The number of evaluations of the function $H(r,s)$ is bounded by nS. Since this is the most expensive step in the algorithm in terms of computation time, the number of evaluations can serve to estimate the computational require-

ments of the SPSS.

2) Clustered shift-sampling: Contrary to the SPSS heuristic, the CSS constructs a roster by selecting teams for a cluster of consecutive shifts at each time. The algorithm is described in Algorithm 2 and its main steps explained in the following.

Let c be the size of a cluster. From the first shift, CSS considers clusters of shifts $< 1, \ldots, c >$, $< c+1, \ldots, 2c >$, ... For convenience, a cluster $< i, \ldots, j >$ is denoted by the starting and ending points $< i, j >$. Sequentially, for each cluster $< i, j >$, the CSS randomly samples *sequences of configurations* that assign a team of agents to each shift $i \leq s \leq j$. The sampling step is represented by the procedure $MakeSamples$ in Algorithm 2.

We denote $\mathcal{A}^{SAMP}_{<i,j>}$ the set of random samples of sequences that are defined for a given cluster $< i, j >$. The set $\mathcal{A}^{SAMP}_{<i,j>}$ is determined using a randomized depth-bounded tree traversal with a constant branching factor. This step is represented by the recursive function $clusterSamplingRec$ in the algorithm.

The recursive procedure that determines $\mathcal{A}^{SAMP}_{<i,j>}$ operates in the following way. We consider a tree where each level represents a shift. Starting from the root node that represents shift i, the algorithm samples n configurations and selects the f most promising ones (i.e., with higher utility). From these selected configurations, the tree traversal continues until arriving at the depth of shift j. Upon arriving to a node located in the last level, a sequence of configurations, including the configuration at that node and those selected in its predecessors in the tree, is added to $\mathcal{A}^{SAMP}_{<i,j>}$

The number of evaluations of the function H in the CSS depends on parameters c, n, and f, and is bounded by the expression $\sum_{0 \leq i \leq c-1} nf^i$. In the evaluation part we investigate the effect of different values for these parameters.

Algorithm: Clustered shift-sampling

Input: cluster size c, number of samples n
Input: branching factor f
Initialize $roster$ with empty shifts
s=1
while $s \leq S$ **do**
  $t = \min(s + c, S)$
  $\mathcal{A}^{SAMP}_{<s,t>}$ = `ClusterSampRec(a,b)`
  foreach $sample \in \mathcal{A}^{SAMP}_{<s,t>}$ **do**
    Temporary assign teams in $sample$ to shifts $[s, t]$ of $roster$
    Evaluate $H(roster, t)$
    Determine efficiency and validity of $roster$ until shift t
  end
  $sample^*$ = Select a *valid* sample in $\mathcal{A}^{SAMP}_{<s,t>}$
  Assign teams in $sample^*$ to shifts $[s, t]$ of $roster$ $s = t$
end
Function `ClusterSampRec(`*roster, s, i, t*`)`**:**
  if *i=t* **then**
    return $\{< roster_s, \ldots, roster_t >\}$
  else
    Determine $\mathcal{A}^{VALID}_i$ based on the current $roster$
    $\mathcal{A}^{SAMP}_i$ = `MakeSamples(`$\mathcal{A}^{VALID}_i$`,`n`)`
    foreach $team \in \mathcal{A}^{SAMP}_i$ **do**
      Temporary assign $team$ to shift i of $roster$
      Evaluate $H(roster, i)$
      Determine efficiency and validity of $roster$ until shift i
    end
    T = Select at most f *valid* teams in $\mathcal{A}^{SAMP}_i$
    C={ }
    foreach $team \in T$ **do**
      Temporary assign $team$ to shift i of $roster$
      C = C $\cup$ `ClusterSampRec(`*roster,s,i+1,t*`)`
    end
    return C
  end

Algorithm 2: Pseudo-code of the clustered shift-sampling heuristic

V. Experimental Evaluation

In this section we present an experimental evaluation of the proposed solution scheme to the RRP for STASP-HMR. All the algorithms were implemented in C++. The numerical results presented here were obtained with an AMD Opteron® Processor (2.0 GHz, 4GB RAM). Only one core was involved in the computational process of each experiment.

A. RRP instances

We consider missions that can last up to 14 days, with 4 shifts a day, for a total of 56 shifts. Each shift of the day represents a problem instance of the STASP-HMR. We consider a mission scenario where the locations of the tasks composing the mission are obtained through a cellular grid decomposition of the area with a one to one correspondence between tasks and cells. The grid size is 20×20, for a total of four hundred tasks. We adopt a dimensionless space, yet we consider that robots must move through adjacent cells: those that share at least one vertex on the grid. This condition is translated into the traversability graph and imposes routing constraints on the set of feasible sequences of tasks that are considered to estimate the utility of a shift.

We consider $|\mathcal{A}| = 32$, and four types of agents, each characterized by the performance in accomplishing the tasks. As introduced in Section III, a task model φ defines the fraction of workload of a task that an agent can tackle during a single time step. Each instance of the STASP-HMR considers a model φ where an agent type exhibits one out of five performance levels for each task, ranging from inefficient ($\varphi \triangleq 1\%$) up to highly efficient ($\varphi \triangleq 10\%$).

Each shift has a duration of 10 mission steps. We remark that, if all agents are used in each shift, a total number of $32 \cdot 10 \cdot 56 = 17,920$ mission steps could be performed in a mission. However, a set of rostering and coverage constraints may restrict the possible coalitions that can be used within each shift. We generate the set of rostering constraints using random values for N^d_a, N_a, M_a, m_a and consider 80 different

sets of rostering constraints.

We consider that the budget and coverage constraints are often correlated in the applications of interest. For instance, we expect that when the budget of the mission increases, the efficiency requirements remain constant or increase as well. Thus, we define a set of *time-dependent requirement profiles* that establish values for both the budget and coverage constraints simultaneously. Specifically, we define *flat*, *linear*, and *periodic* profiles. A flat profile defines budget and coverage constraints that remain constant along the mission, that is, $\mathcal{B}_s$ and Γ_s are the same for all shifts s. A linear profile defines scenarios in which the values of $\mathcal{B}_s$ and Γ_s monotonically increase or decrease along the mission. A periodic profile defines scenarios in which the values of $\mathcal{B}_s$ and Γ_s increase and decrease, synchronously and periodically, along the mission. The specific values are based on expected performances of the team given $\mathcal{A}$ and φ.

It is important to remark that, for the specified number of shifts and the given scenario setup, it is possible that the resulting rosters do not complete all tasks composing the mission. Nevertheless, when we compare different rosters we consider as a metric the utility that they provide in terms of the amount of workload tackled (i.e., the percentage of the mission that was completed). We believe that our performance analysis holds also in the cases when the mission setup do enable the system to complete all tasks.

B. Comparison of different selection criteria for SPSS

We consider four different selection criteria in the SPSS heuristic: greedy, $\epsilon-$greedy, minimum cardinality, and random. The greedy selection chooses the sample configuration with maximal utility. The $\epsilon-$greedy selects a random configuration with $\epsilon = 0.5$ probability, and makes a greedy choice with $1 - \epsilon$ probability. The minimum cardinality selects the configuration with the least number of agents. Finally, we also consider a random choice. The above strategies, albeit trivial, serve as a baseline for our comparison.

We consider different sizes of the sets $\mathcal{A}_s^{SAMP}$ using $n \in \{1000, 5000, 10000\}$. In the heuristics, the number of evaluations is the main factor that determines the computation time. In the case of the SPSS, this number is defined by the size of $\mathcal{A}_s^{SAMP}$. Figure 1 shows how the computation time relates to the number of evaluations. Here, we also note that, contrary to the SPSS, the CSS methods do not have a fixed number of evaluations. While an upper bound can be derived, the CSS may incur in less evaluations if the number of valid samples in the recursive step is less than the branching factor. From the results in Figure 1 we note that these situations arise very often because the number of evaluations are rather dispersed.

In Figure 2 (top) we rank each strategy according to the utility achieved by their rosters. Results indicate that the greedy selection clearly outperforms the other strategies. However, as mentioned in Section IV-B.1, if we seek to obtain a maximal utility at each shift we may quickly exhaust the resources and be forced to leave some shifts empty (i.e., uncovered). Therefore, we also analyze the number

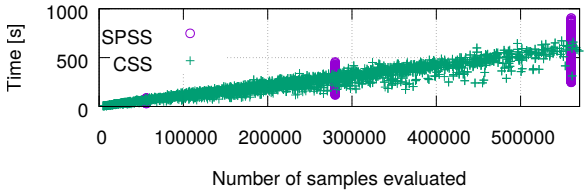

Fig. 1. Computation time vs. number of samples evaluated.

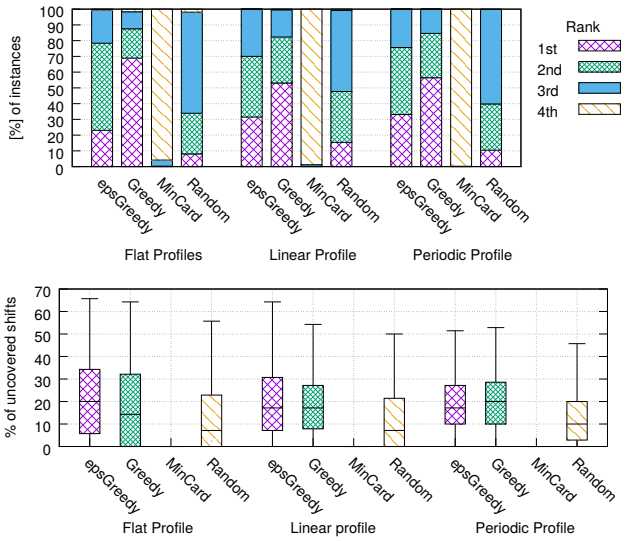

Fig. 2. Relative ranking (top) and percentage of uncovered shifts (bottom) using different selection criteria for the SPSS heuristic under various roster profiles.

of uncovered shifts that each strategy incurs. Results in Figure 2 (bottom) illustrate the problematic issues arising from a greedy selection of teams. Both greedy and ϵ-greedy strategies incur a significant fraction of shifts left uncovered, while selecting small teams (minimum cardinality strategy) enables more freedom in the construction of rosters that cover all shifts. Of course, the impact of uncovered shifts is application-dependent. Nevertheless, if a roster provides a good overall utility but contains a small number of shifts that remain uncovered, some repair actions could be applied to the roster in order to fulfill its empty shifts while retaining its good performance.

C. Effect of branching factor and of # of samples on CSS

Next, we evaluate the performance of the CSS using different values for the branching factor f, and number of samples n. To simplify our analysis we consider a single cluster size value of $c = 3$. Larger values of c would incur in a significant increase of the number of samples to be evaluated and thus increasing the computation time required for the experimental analysis. Therefore, here we only study the effects of the branching factor f and the sample size n for a fixed c. We consider the setups indicated in Table I.

Figures 3 and 4 show the relative ranking and the percentage of uncovered shifts of the CSS, respectively. The results indicate that the number of samples n dominates f. We also observe that the CSS outperforms the SPSS both in utility

f	n	max. # of eval.	avg. eval. x shift
5	310	143220	2557
5	615	284130	5073
5	1225	565950	10106
10	80	145760	2600
10	155	282410	5043
10	310	564820	10086
15	35	142870	2551
15	70	285740	5102
15	140	571480	10205

TABLE I

PARAMETERS USED FOR THE CSS HEURISTIC.

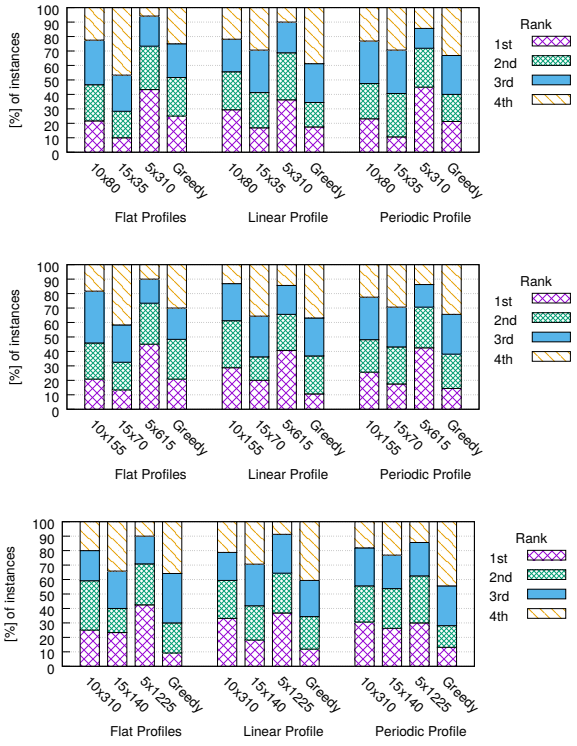

Fig. 3. Relative ranking of the CSS heuristic using different values for the branching factor f and number of samples n and the SPSS greedy heuristic (Greedy) using a similar number of samples. Each combination of parameters for the CSS is denoted as $f \times n$. We grouped methods that have a similar bound on the number of samples considered (see Table I).

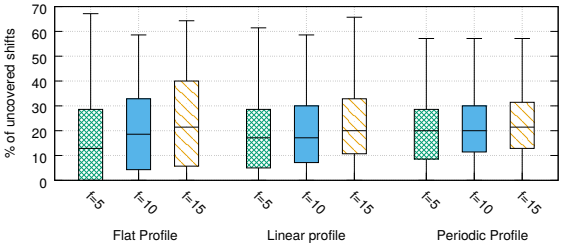

Fig. 4. Percentage of uncovered shifts using different branching factors.

and number of uncovered shifts.

VI. CONCLUSIONS

We have introduced the robot rostering problem, that accounts for the use of heterogeneous multi-agent and multi-robot teams in mission scenarios that need sustained operations for long time spans and require work shifts. A formal model of the robot rostering problem has been provided based on a multi-constrained integer programming formulation. We have discussed the reasons that make the robot rostering problem different from related problems addressed in the operations research literature, and we have shown that these same reasons result into a problem which is computationally intractable. To face the computational challenges we have proposed a number of sampling-based heuristics to support an incremental, constructive approach to generate feasible solutions. In our model a solution consists of a sequence of rosters that are jointly compliant with coverage, budget, and exclusion/inclusion constraints. The goal is to find solutions that are also optimized in the sense of selecting effective rosters for each time shift to cover. First, based on extensive simulation experiments, we could observe, that our heuristic approach is computationally feasible, allowing to compute a roster solution in matter of seconds or minutes. Moreover, the computed roster resulted to be feasible in practice (i.e., when the roster is issued and put into action) in about the 80-85% of the cases, on average. We have also studied the impact of different strategic choices, such as different time profiles to allocate the global mission budget over the entire mission time, and the use of more or less samples and lookahead when constructing the solution. These studies gave us useful information about how to design and shape the solution heuristics and the model to obtain good solutions in practice.

REFERENCES

[1] J. Van den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, and L. De Boeck, "Personnel scheduling: A literature review," *European Journal of Operational Research*, vol. 226, no. 3, pp. 367–385, 2013.

[2] A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier, "Staff scheduling and rostering: A review of applications, methods and models," *European Journal of Operational Research*, vol. 153, no. 1, pp. 3–27, 2004.

[3] E. K. Burke, P. De Causmaecker, G. V. Berghe, and H. Van Landeghem, "The State of the Art of Nurse Rostering," *Journal of Scheduling*, vol. 7, no. 6, pp. 441–499, 2004.

[4] P. Langley, "Systematic and Nonsystematic Search Strategies," in *Proc. of the First Int. Conf. on Artificial Intelligence Planning Systems*. Morgan Kaufmann Publishers Inc., 1992, pp. 145–152.

[5] E. Feo Flushing, L. M. Gambardella, and G. A. Di Caro, "A mathematical programming approach to collaborative missions with heterogeneous teams," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2014, pp. 396–403.

[6] E. Feo Flushing, L. Gambardella, and G. A. Di Caro, "On decentralized coordination for spatial task allocation and scheduling in heterogeneous teams," in *Proceedings of the 15th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, 2016, pp. 988–996.

[7] E. F. Flushing, M. Kudelski, L. M. Gambardella, and G. a. Di Caro, "Connectivity-aware planning of search and rescue missions," in *Proc. of the IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2013, pp. 1–8.

[8] G. Steinbauer and A. Kleiner, "Towards CSP-based mission dispatching in C2/C4I systems," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2012, pp. 1–6.