A Scalable Heuristic for Mission Planning of Mobile Robot Teams

Anders Lager [*,**] **Branko Miloradović** [**]
Giacomo Spampinato [*] **Thomas Nolte** [**]
Alessandro V. Papadopoulos [**]

[*] *ABB AB, Västerås, Sweden*
(e-mail:{anders.lager,giacomo.spampinato}@se.abb.com)
[**] *Mälardalen University, Västerås, Sweden (e-mail:{branko.miloradovic,thomas.nolte,alessandro.papadopoulos}@mdu.se)*

Abstract: In this work, we investigate a task planning problem for assigning and planning a mobile robot team to jointly perform a kitting application with alternative task locations. To this end, the application is modeled as a Robot Task Scheduling Graph and the planning problem is modeled as a Mixed Integer Linear Program (MILP). We propose a heuristic approach to solve the problem with a practically useful performance in terms of scalability and computation time. The experimental evaluation shows that our heuristic approach is able to find efficient plans, in comparison with both optimal and non-optimal MILP solutions, in a fraction of the planning time.

Keywords: Mobile Robotics, Task Planning

1. INTRODUCTION

Coordinating a fleet of robots to perform various tasks is a problem with high complexity, requiring the solving of many sub-problems. In this paper, we investigate how to efficiently plan a kitting application mission, including the selection of a robot team for this purpose. In a kitting application, a specified group of objects of different types shall be fetched from different locations and thereafter delivered to a specified location. To increase operational efficiency, some object types may be available at multiple alternative locations, e.g., if the demand for these objects is high. Robots available for the mission may be located at different locations, e.g., for charging. The targeted objective is to minimize the makespan, which promotes a balanced usage of the robots and reduces the aggregation time at the delivery location.

This problem belongs to the category of Multi-Robot Task Allocation (MRTA) (Khamis et al., 2015). It can also be categorized as a variant of the Capacitated Vehicle Routing Problem (VRP), first studied by Dantzig and Ramser (1959), and extensively studied during the latest decades to solve many related problem variants in different domains, e.g., logistic operations and production planning (Konstantakopoulos et al., 2020; Lahyani et al., 2015). For a basic VRP, the problem is to deliver a set of customer orders with a set of vehicles located at a single depot. The customers to be served are located at different locations and the vehicles need to return to the depot after all customers have been served. Vehicles have a limited order capacity. A solution to the problem will decide the number of vehicles and their routes, while the objective is to minimize the total cost of all routes. In our problem variant, vehicles represent mobile robots, and customers to be visited represent robot tasks to be done. Since the robots are assumed to have an unlimited capacity, the problem may also be categorized as a Multiple Travelling Salesperson Problem (MTSP) which covers a subset of VRPs without capacity constraints (Cheikhrouhou and Khoufi, 2021). The selection of a team of robots, each one having a different start location, can be categorized as a VRP with Multiple Depots (Montoya-Torres et al., 2015), here assuming there is only one robot at each depot. The problem includes AND-type precedence constraints (Bahalke et al., 2022), limiting the execution order of intra-schedule tasks (Korsah et al., 2013). It also includes separation constraints (Bortfeldt and Wäscher, 2013) to provide mutual exclusion of delivery tasks within the same route. These tasks are used to deliver aggregated objects. Additionally, the robots are not required to return to their starting depots to complete the mission, making it an Open VRP. In practice, the robots may immediately become available for a new mission and they may start moving towards a currently free charging location. The problem can also be categorized as a Balanced VRP since a minimum makespan objective promotes a balanced usage of the robot team, as long as the mission does not include dominating tasks (Miloradović et al., 2019a). The existence of sets of alternative tasks, in which only one task needs to be visited, motivates the categorization of the problem as a Multiple Generalized TSP (MGTSP). MGTSP and the single robot version GTSP (Ilavarasi and Joseph, 2014), are generalizations of MTSP and TSP respectively, where the tasks are divided into different subsets and at least one task in each subset needs to be visited. In our problem, these subsets are mutually exclusive and exactly one task needs to be visited. To

[⋆] This work was funded by The Knowledge Foundation (KKS) and by The Swedish Foundation for Strategic Research (SSF).

summarize, we have a Balanced, Open, Multi Depot, VRP/MGTSP with Precedence Constraints (PC) and separation constraints.

In this paper, we propose an interpretation and extension of the Robot Task Scheduling Graph (RTSG) (Lager et al., 2021) to model the addressed multi-robot problem. Moreover, we provide a problem formulation as a Mixed Integer Linear Program (MILP), that can be used by a MILP Solver to find optimal solutions for smaller problem instances, given enough time. We also propose a heuristic multi-step approach, targeting a reduced planning time and efficient solving of larger problem instances. In the first step, tasks are partitioned into clusters with a semi-supervised clustering approach based on K-medoids. Thereafter, the clusters are modeled as separate single-route scheduling problems using MILP, or as asymmetric Travelling Salesperson Problems. Finally, any remaining alternative tasks are removed and the computed schedules are balanced with a local search approach to further minimize the makespan.

The solution quality and the planning time of the heuristic approach are compared with a MILP solver, indicating an ability to generate high-quality solutions within a practically acceptable time frame. Other benchmark approaches were not found that can be applied to the full problem description in an obvious way. To the best of our knowledge, OR-type PC for VRP, (Bahalke et al., 2022), with *alternative* predecessors, have not been addressed in the VRP literature.

The remainder of this paper is organized as follows. Sect. 2 presents related works. Sect. 3 describes the problem and the assumptions made, while Sect. 4 gives a formal problem formulation as a MILP. Sect. 5 details the heuristic solution approach, and Sect. 6 provides the experimental evaluation. The study is concluded in Sect. 7.

2. RELATED WORK

Robot Task Scheduling Graph is an intuitive task modeling formalism proposed by Lager et al. (2021). It can be used by a domain expert to model an industrial mobile robot application while leveraging automated planning. Until this work, this modeling formalism has only been applied to single-robot planning.

For a VRP, a customer normally *must* be served and there is only one address to go to. However, a task to be performed by a robot can often be alternative or be performed at alternative locations. Examples of alternative customers (or task locations) in the VRP literature are sparse but can be found, e.g., in the work by Goel and Gruhn (2008), where the profit is maximized by deciding if a transportation request shall be assigned to a vehicle or bought. Mathew et al. (2015) used an MGTSP problem formulation to plan a team of charging robots to support a team of Unmanned Aerial Vehicles on the ground at alternative locations along planned flight trajectories. Touzani et al. (2021) targeted a combined sequencing and path generation problem for industrial robots, using a genetic algorithm to solve an MGTSP for the sequencing including a selection of alternative robot configurations.

Precedence constraints of AND-type have been used extensively in previous works, and OR-type PC was suggested recently for VRP where only one of a task's predecessors must precede the task in a plan (Bahalke et al., 2022; Roohnavazfar et al., 2022). In our problem, OR-type PC are modeled. However, different from the mentioned works, OR-type predecessors of a task are *alternative* in the sense that only one will exist in a plan.

The objective to minimize the makespan can often be improved by increasing the robot team size. However, robots are a limited resource that may be used in parallel for alternative missions or multi-mission problems (Miloradović et al., 2019b). Therefore, being able to specify the number of robots, i.e., the number of clusters for the proposed heuristic approach, can be considered an advantage for this problem. K-means is a popular method to partition nodes into K clusters where the total distance between the nodes and their Euclidian cluster center points is minimized (Jain, 2010). However, the routes for our problem, e.g., in a warehouse, are seldom linear and a Euclidian center point may not be close or even reachable from other cluster nodes. K-medoids (Park and Jun, 2009) is a more suitable approach where a node is identified as a center point. To consider separation constraints in the computation of clusters, a supervised clustering approach can be applied. Our supervised approach is based on the Variable Neighborhood Search (VNS) algorithm presented by Randel et al. (2019). Cluster algorithms that also identify the number of clusters sometimes referred to as *automatic clustering*, can be appropriate for some problem types. Several automatic clustering approaches for MRTA are listed by Ayari and Bouamama (2019).

Ayari and Bouamama (2019), addressed an MRTA problem where tasks shall be assigned to robots with the objective to minimize the total completion time given by a fitness function. They used a two-step approach with dynamic clustering based on Particle Swarm Optimization followed by a robot assignment to clusters with an approach including the solving of TSP problems for each combination. Xu et al. (2017) demonstrated a two-phase heuristic approach to compute solutions for a Balanced MTSP, where a balanced K-means was used to get clusters with evenly distributed nodes. This was followed by a genetic algorithm to compute routes. Murugappan et al. (2021) compared the performances of different clustering algorithms for solving a Balanced MTSP, where a convex hull TSP algorithm was used to generate the routes for each cluster. The modeling of alternative tasks for the single route problem within an asymmetric TSP has similarities with the approach for converting an asymmetric GTSP into an asymmetric TSP described by Laporte and Semet (1999).

3. PROBLEM DESCRIPTION AND ASSUMPTIONS

The problem is to fetch and deliver a set of objects while minimizing the mission makespan. To this end, a team of robots needs to complete a set of *tasks*. These are Single-robot Tasks and the robots are Single-task Robots in a Time-extended Assignment (Gerkey and Matarić, 2004). The team size is fixed beforehand. The problem has In-Schedule Dependencies but no Cross-Schedule Dependencies (Korsah et al., 2013). It is assumed there is no restriction in the number of fetch tasks that can be allocated to a single robot. Furthermore, the robots are homogeneous and initially located at different start locations. Each task is associated with a location and an action duration. A *schedule* is referred to as an ordered sequence of tasks to be executed by a single robot. The solution shall identify 1) the robots to be used for the mission, and 2) the schedules for the used robots. A mission work description can be modeled with an RTSG model (Lager et al., 2021). In this representation, exemplified in Fig. 1, rectangular nodes represent tasks. Directed edges and paths represent precedence constraints. A start state is represented by an S-node and the goal state is represented by a G-node. AND-Fork (&F) and AND-Join (&J) node pairs are used to split and rejoin edges. OR-Fork (||F) and OR-Join (||J) node pairs split a branch into alternative branches. In previous works, RTSG has exclusively been used with a single robot.

In our model (Fig. 1), a set of fetch tasks ($F_i, G_{j,k} : i, j, k \in \mathbb{N}$) need to be completed before the execution of delivery tasks ($D_l : l \in \mathbb{N}$). The model allows for defining a number of OR-pair groups, each one containing a set of alternative fetch tasks ($G_{j,k}$) where only one of them will be planned (j indicates the OR-pair group). There is one delivery task for each schedule. The delivery tasks can be co-located, but this is not a requirement for the solution approaches. We propose a few multi-robot interpretations and extensions of RTSG for the multi-robot problem at hand:

- Precedence constraints do only apply if the involved tasks become allocated to the same schedule.
- If a predecessor of a task is mandatory, it matches an AND-type PC for VRP in accordance with Bahalke et al. (2022), where all predecessors of a task must be planned before any successor.
- If a predecessor is alternative, it matches an OR-type PC for VRP, as proposed by Bahalke et al. (2022), where only one of the predecessors to a common task must precede it. Since the predecessors are alternative, only one of them will be planned for execution.
- A Separation Constraint (SC) indicates a group of tasks that needs to be separated into different schedules. In Fig. 1, the blue color is used to indicate a SC for the group of delivery tasks.

In the proposed form, the RTSG model is a flexible but constrained multi-robot work description.

4. PROBLEM FORMULATION

In this section, a MILP problem formulation is proposed that can be used to compute optimal solutions of the MRTA problem presented in Section 3.

4.1 Decision variables and objective

Let r be a robot schedule in the set of schedules R, where the number of total schedules is known *a priori*. The set of robot start locations is defined by S, representing available robots. For each start location $s \in S$, there is a corresponding goal state, $E(s) \in G$. $|S| = |G| \geq |R|$. The set of all tasks is denoted as A. For convenience, we

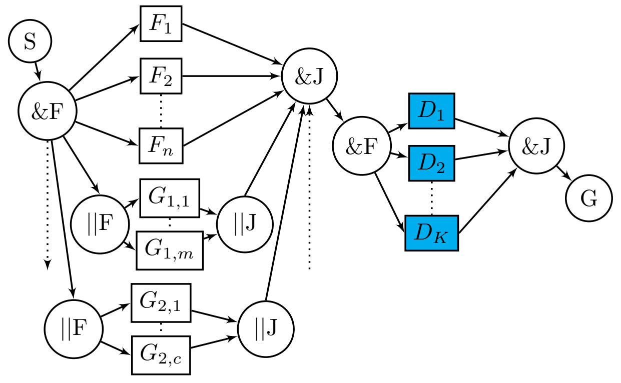


Fig. 1. A work description for the multi-robot mission modeled as a Robot Task Scheduling Graph.

indicate with $A^S = A \cup S$, with $A^G = A \cup G$, and with $\tilde{A} = A \cup S \cup G$. The set B is the set of all non-mandatory tasks, indicating they may, or may not be a part of a schedule. The set B is a subset of set A, i.e., $B \subseteq A$. Other tasks that belong to the set $A \setminus B$ are mandatory. The notation $j \prec k$ where $j, k \in \tilde{A}$, indicates task j must be scheduled before task k if they are assigned to the same schedule.

Decision variable $X_{j,k,r}$ is a binary variable, i.e., $X_{j,k,r} \in \{0, 1\}, \forall j, k \in \tilde{A}$, and $\forall r \in R$, where

$$X_{j,k,r} = \begin{cases} 1, & \text{if there is a scheduled direct transition} \\ & \text{from task } j \text{ to task } k \text{ within schedule } r, \\ 0, & \text{otherwise,} \end{cases}$$

where $X_{j,j,r} = 0, \forall j \in \tilde{A}, \forall r \in R$. The cost to perform task k after task j within the same schedule includes routing cost ($\tau_{j,k}$) and action cost (α_k):

$$C_{j,k} = \tau_{j,k} + \alpha_k. \tag{1}$$

There is no cost to reach the goal state, $C_{j,G} = 0, \forall j \in \tilde{A}$. The objective function used for optimization is a MiniMAX (Nunes et al., 2017). The objective function J is defined as:

$$J = \max_r \sum_{j \in A^S} \sum_{k \in A^G} (X_{j,k,r} \cdot C_{j,k}). \tag{2}$$

Since J is not a linear function in $X_{j,k,r}$ for Equation 2, the objective function is modeled as a constraint by including J as one extra decision variable where the size of J is limited by additional MiniMAX constraints for the total cost of each schedule:

$$-J + \sum_{j \in A^S} \sum_{k \in A^G} (X_{j,k,r} \cdot C_{j,k}) \leq 0, \qquad \forall r \in R, \tag{3}$$

4.2 General constraints

There are transitions from exactly $|R|$ robot start locations:

$$\sum_{j \in S} \sum_{k \in A^G} \sum_{r \in R} X_{j,k,r} = |R|. \tag{4}$$

If a transition from a robot start location goes directly to the goal state, i.e., $X_{j,k,r} = 1 : j \in S, k \in G, r \in R$, schedule r is considered empty and the robot at start location j will not be allocated. There is at most one transition from a robot start location and there are no transitions to a robot start location as defined by:

$$\sum_{k\in A^G}\sum_{r\in R} X_{j,k,r} \leq 1, \qquad \forall j \in S, \tag{5}$$

$$\sum_{j\in \tilde{A}}\sum_{r\in R} X_{j,k,r} = 0, \qquad \forall k \in S. \tag{6}$$

Similarly, there is at most one transition to a robot goal state and there are no transitions from a robot goal state:

$$\sum_{j\in A^S}\sum_{r\in R} X_{j,k,r} \leq 1, \qquad \forall k \in G, \tag{7}$$

$$\sum_{k\in \tilde{A}}\sum_{r\in R} X_{j,k,r} = 0, \qquad \forall j \in G. \tag{8}$$

One start location is included in each schedule:

$$\sum_{j\in S}\sum_{k\in A^G} X_{j,k,r} = 1, \qquad \forall r \in R. \tag{9}$$

The number of transitions from a robot start location in a schedule is the same as the number of transitions to the corresponding goal state:

$$\sum_{k\in A^G} X_{j,k,r} = \sum_{j'\in A^S} X_{j',E(j),r}, \quad \forall j \in S, \forall r \in R. \tag{10}$$

There is one transition from and one transition to mandatory tasks:

$$\sum_{k\in A^G}\sum_{r\in R} X_{j,k,r} = 1, \qquad \forall j \in A \setminus B, \tag{11}$$

$$\sum_{j\in A^S}\sum_{r\in R} X_{j,k,r} = 1, \qquad \forall k \in A \setminus B. \tag{12}$$

There is at most one transition from and one transition to non-mandatory tasks.

$$\sum_{k\in \tilde{A}}\sum_{r\in R} X_{j,k,r} \leq 1, \qquad \forall j \in B, \tag{13}$$

$$\sum_{j\in \tilde{A}}\sum_{r\in R} X_{j,k,r} \leq 1, \qquad \forall k \in B. \tag{14}$$

The number of incoming and outgoing transitions to non-mandatory tasks must be the same:

$$\sum_{k\in \tilde{A}}\sum_{r\in R} X_{j,k,r} = \sum_{j'\in \tilde{A}}\sum_{r\in R} X_{j',j,r}, \qquad \forall j \in B. \tag{15}$$

A task is entered and departed within the same schedule:

$$\sum_{j\in A^S} X_{j,k,r} = \sum_{k'\in A^G} X_{k,k',r}, \qquad \forall k \in A, \forall r \in R. \tag{16}$$

No cyclic sub-route are allowed:

$$\sum_{j\in V}\sum_{k\in V} X_{j,k,r} \leq |V| - 1, \quad \forall V \subseteq A : V \neq \emptyset, \forall r \in R. \tag{17}$$

Thus, to eliminate the sub-tours, it is required that for each nonempty subset $V \subseteq A$, the number of edges between the elements of V must be at most $|V| - 1$. Let O be a set of alternative tasks in the set of all sets of alternative tasks $\tilde{O}$, i.e., $O \in \tilde{O}$ where $O \cap U = \emptyset, \forall U \in \tilde{O} \setminus O$. Additionally, O is a subset of B, i.e., $O \subseteq B$. The number of transitions to and from a set of alternative tasks is limited to 1.

$$\sum_{s\in O}\sum_{j\in A^G}\sum_{r\in R} X_{s,j,r} = 1, \qquad \forall O \in \tilde{O} \tag{18}$$

$$\sum_{j\in A^S}\sum_{s\in O}\sum_{r\in R} X_{j,s,r} = 1, \qquad \forall O \in \tilde{O} \tag{19}$$

In order to constrain fetch tasks to be done before delivery tasks within the same schedule, as indicated by the edges of the RTSG in Fig. 1, general precedence constraints are introduced:

$$\sum_{j=1}^{|D|-1} X_{D_j,D_{j+1},r} \leq |D| - 2, \\ \forall D \subseteq \tilde{A} : |D| \geq 2,\ D_{|D|} \prec D_1,\ \forall r \in R, \tag{20}$$

where D is an ordered subset $D = \{D_1, \ldots, D_{|D|}\} \subseteq \tilde{A}$.

4.3 Delivery task constraints

Delivery tasks, $P \subseteq B$, cannot be allocated to the same schedule. The number of delivery tasks equals the number of computed schedules, $|P| = |R|$. Since schedules may become empty, delivery tasks are a subset of the non-mandatory tasks, B. At most one delivery task in P is allocated to schedule r, i.e., a separation constraint. No delivery task is allocated to r if the schedule is empty, meaning there is a direct transition between the start location and the goal state:

$$\sum_{j\in A^S}\sum_{k\in P} X_{j,k,r} \leq 1, \qquad \forall P \subseteq B, \forall r \in R, \tag{21}$$

$$X_{s,E(s),r} + \sum_{j\in A^S}\sum_{k\in P} X_{j,k,r} = 1, \quad \forall P \subseteq B, \forall s \in S, \forall r \in R. \tag{22}$$

5. HEURISTIC APPROACH

The general idea of the proposed heuristic approach is to partition all tasks into K mutually exclusive clusters, where K is the number of robots to use. Thereafter, a single route scheduling problem is solved for each cluster. Eventually, alternative tasks remaining in different routes are reduced, and the costs of the routes are balanced by transferring tasks between the routes.

5.1 Task Clustering

To reduce routing costs, it is assumed that tasks within the same cluster should be in proximity to each other. K-medoids partitions the nodes into K clusters while minimizing the total dissimilarities of the nodes and their *medoid*, i.e., a node assigned as the center for a cluster:

$$\sum_{l\in L}\sum_{i\in l} d_{m_l,i}, \tag{23}$$

where L is the set of clusters ($|L| = K$), $d_{x,y}$ is a dissimilarity measure of nodes x and y. The assigned medoid of cluster l is m_l.

For our problem, nodes represent robot tasks, and the elements of the dissimilarity matrix, $d_{x,y}$, represent routing costs between task x and task y. The separation constraints for the delivery tasks require them to end up in different clusters. These constraints are converted to *cannot-link constraints*, indicating tasks that must be separated in different clusters (Basu et al., 2008).

The clustering approach is detailed in Algorithm 1. It implements the Semi-Supervised K-medoids algorithm presented by Randel et al. (2019). It starts with an initial

Algorithm 1 Semi Supervised K-medoids

```
function ComputeClusters($A$,$k$)
  $CL \leftarrow$ K-medoids clustering of $A$ with $k$ random medoids
  $CL \leftarrow$ Repair($CL$)
  $ret \leftarrow CL$
  while Stop criterion not reached do
    $v \leftarrow 1$
    while $v \leq k$ do
      $CL \leftarrow$ Switch $v$ medoids randomly in $CL$
      $CL \leftarrow$ Repair($CL$)
      $CL \leftarrow$ LocalSearch($CL$)
      $CL \leftarrow$ Repair($CL$)
      if $CL$.cost() < $ret$.cost() then
        $ret \leftarrow CL$
      $v \leftarrow v + 1$
  return $ret$
```

random selection of K medoids, and the tasks in the set A are grouped with their closest medoid. In each iteration, a randomly selected subset of the medoids are exchanged and the tasks are regrouped. Thereafter, the *LocalSearch* algorithm, proposed by Resende and Werneck (2007), reduces the total cluster cost (23) by greedy modifications of the medoid selections and regrouping of tasks. After each modification of the medoid selection, a repair step adjusts the clusters to satisfy cannot-link constraints by moving conflicting, non-medoid tasks between clusters. Different from the original approach by Randel et al. (2019), the cost of the clusters to be minimized is set in this paper as the max cluster cost:

$$\max_{l \in L} \sum_{i \in l} d_{m_l,i} + \alpha_i \tag{24}$$

where $d_{m_l,i}$ and α_i represent routing cost and action cost.

5.2 Routing and robot selection

A MILP model is used to model the routing and robot selection problem for an individual cluster, having a significantly smaller complexity than the MILP model in Section 4. The decision variables are significantly less as we do not have dimension r. Since the clustering already provided the allocation of tasks to robots, we can now solve the sub-problems for each robot. Specifically, at least K such problems need to be solved in this approach. These smaller problems can be solved by a MILP solver to compute (sub-)optimal solutions for the sub-problems. Moreover, for the kitting application problem investigated in this work, the sub-problems are modeled as asymmetric TSPs making it possible to use off-the-shelf efficient TSP solvers, instead of using more general-purpose solvers to compute solutions of the downsized MILP problem presented in Section 4. The optimality of the overall solution will be affected by the clustering resulting from the semi-supervised K-medoids method, i.e., the solution of the asymmetric TSP will only optimize the route of the individual robots, but the task allocation is decided a priori through the clustering.

5.3 TSP modeling

As it is already mentioned in the sub-section above, the reduced routing and robot selection problem can be expressed as an asymmetric TSP problem. The reason for this conversion is the possibility to use efficient solvers dedicated to the TSP problem, which will have some performance advantage over a more general MILP solver. In order to use symmetric TSP solvers, e.g., Concorde (Applegate et al., 2011), the asymmetric TSP is transformed into a symmetric TSP with the approach by Jonker and Volgenant (1983). In a TSP problem, a salesperson needs to visit all assigned cities and return to the starting point. An optimal solution will find a visiting order that minimizes the total travel distance. A presumption for the presented conversion to TSP is a limited problem instance in terms of modeled precedence constraints, where one task, i.e., the delivery task d, is preceded by all other tasks and needs to be visited last. This is in accordance with the RTSG model in Fig. 1, where the objects can be fetched in any order before being delivered. The TSP problem can be specified with a cost matrix, $C_{j,k} : j,k \in A^S$:

$$C_{j,k} = \begin{cases} 0 & \forall j,k \in S \\ 0 & j = d, \forall k \in S \\ M & k = d, \forall j \in S : |A| > 1 \\ M & j = d, \forall k \in A \\ M & \forall j \in A, \forall k \in S \\ \tau_{j,k} + \alpha_k & \text{otherwise, where } j,k \in A^S \end{cases} \tag{25}$$

where M is a value big enough to block related transitions. A solution will put all start locations in a sequence, where the last indicates the selected robot. It is followed by all the tasks where the delivery task becomes last. Thereafter, the cost matrix is modified and extended to handle subsets of alternative tasks:

For a subset representing a set of alternative tasks, $O = \{O_1, \ldots O_n\} \subseteq A$, where n is the number of tasks in a corresponding OR-pair group of the RTSG model, one extra task $O_i^{'}$ is added for each $O_i, \forall i \in \{1,\ldots,n\}$, where $O^{'} = \{O_1^{'}, \ldots O_n^{'}\} \subseteq A \subseteq A^S$. The transition costs are arranged to make O_i represent transitions *to* this alternative task while the added $O_i^{'}$ represents transitions *from* the very same task. We indicate with $O^E = O \cup O^{'}$ and order the tasks of O and $O^{'}$ in closed loop sequences, so that $O_{i+n} = O_i, \forall i \in \{1,\ldots,n\}$ and $O_{i-n}^{'} = O_i^{'}, \forall i \in \{1,\ldots,n\}$. For each subset O, the related elements of the cost matrix are defined as:

$$C_{j,k} = \begin{cases} \tau_{j,k} + \alpha_k & \forall j \in A^S \setminus O^E, \forall k \in O \\ M & \forall j \in O, \forall k \in A^S \setminus O^E \\ \tau_{j,k} + \alpha_k & \forall j \in O^{'}, \forall k \in A^S \setminus O^E \\ M & \forall j \in A^S \setminus O^E, \forall k \in O^{'} \\ 0 & j = O_i, k = O_{i+1}, \forall i \in \{1,\ldots,n\} \\ 0 & j = O_i, k = O_i^{'}, \forall i \in \{1,\ldots,n\} \\ 0 & j = O_i^{'}, k = O_{i-1}^{'}, \forall i \in \{1,\ldots,n\} \\ M & \text{otherwise: } j,k \in O^E \end{cases} \tag{26}$$

where M is a value big enough to block related transitions. Within a solution of the TSP, all the tasks of a set O^E become grouped in a sub-sequence: $(O_i, O_{i+1}, \ldots, O_{i+n}, O_{i+n}^{'}, \ldots, O_{i+1}^{'}, O_i^{'})$ where $i \in \{1,\ldots,n\}$. Only the outer task pair, $O_i, O_i^{'}$, contributes a cost to the solution, while the inner transitions have zero cost. In turn, the outer task pair indicates the selected alternative task, and the intermediate tasks are removed from the solution. All sets of alternative tasks that were distributed over multiple clusters will remain with one task in each computed schedule. In this step, all of them are removed and bypassed, except the one whose removal causes the smallest cost saving.

5.4 Balancing

The balancing step is a local search algorithm, detailed in Algorithm 3. At each iteration, a task is selected and moved from the schedule with the highest cost into another sequence that does not increase the maximum cost. The selected task may not have a separation constraint from tasks in the receiving sequence. The selection criteria of the task, the receiving schedule, and the predecessor in the receiving schedule is critical to avoid sub-optimization. E.g., a prioritization of the largest overall cost reduction will often move tasks to the least expensive sequence, even if other scheduled sequences are much closer. Another important aspect is to minimize the intersections between different routes. Our selection criterion is based on maximizing a gain vs loss ratio, where the gain is the cost reduction of the sending sequence and the loss is the cost increase of the receiving sequence.

5.5 Algorithmic overview

A pseudo-code for the heuristic approach is given in Algorithm 2. The main function ComputeSchedules, takes as input arguments the set of tasks (A), the set of robot start locations (S), and the number of schedules (k) to be computed. First, the clusters are computed using the function *ComputeClusters*. Thereafter, a schedule and a robot selection are computed for each cluster, using the TSP model in Section 5.3. Since there is a chance that a group of multiple schedules may select the same robot, the schedule in the group with the highest cost will be used, while the other schedules are recalculated without this robot available. The worst case for this conflict resolution approach is $K(1+K)/2$ schedule computations, which may cause a planning efficiency problem if the given robot team size, K, is large. In the next step, redundant alternative tasks are removed. Finally, the schedules are balanced with Algorithm 3, *BalanceSchedules*.

Algorithm 2 Clustering-based heuristic approach.

```
function ComputeSchedules($A$,$S$,$k$)
    $CL \leftarrow$ ComputeClusters($A, k$)                    ▷ Clusters
    $SC \leftarrow \varnothing$                                ▷ Schedules
    $AR \leftarrow S$                                          ▷ Available robots
    for all $c \in CL$ do
        $s \leftarrow$ Schedule()                              ▷ New empty schedule
        $s.cl \leftarrow c$                                    ▷ Schedule has cluster $c$
        $s.seq \leftarrow \varnothing$                         ▷ Schedule has no sequence yet
        $s.robot \leftarrow \varnothing$                       ▷ Schedule has no robot yet
        $SC \leftarrow SC \cup s$
    while $|S| - |AR| < k$ do
        for all $s \in SC$ do
            if $s.robot = \varnothing \vee s.robot \notin AR$ then
                $s.robot, s.seq \leftarrow$ ComputeSchedule($s.cl, AR$)
        for all $s \in SC$ do
            if $s.robot \in AR$ then
                $y \leftarrow x \in SC : x.robot = s.robot, \max_x x.cost$
                $AR \leftarrow AR \setminus s.robot$
    $SC \leftarrow$ ReduceAlternativeTasks($SC$)
    $SC \leftarrow$ BalanceSchedules($SC$)
    return $SC$
```

Algorithm 3 Balancing of schedules

```
function BalanceSchedules($SC$)
    $SC$.Sort()                  ▷ Sort in order of decreasing routing cost
    while true do
        $costImproving \leftarrow$ false
        $maxRatio \leftarrow 0$
        for all $t \in SC[0].seq$ do              ▷ tasks in the longest route
            for all $s \in SC \setminus SC[0]$ do
                for all $p \in s.seq$ do
                    if NoSeparationConstr($p, SC[0].seq$) then
                        $loss \leftarrow$ CalcLoss($s.seq, t, p$)
                        if $s.seq.cost + loss < SC[0].seq.cost$ then
                            $gain \leftarrow$ CalcGain($SC[0].seq, t$)
                            $ratio \leftarrow gain/loss$
                            if $ratio > maxRatio$ then
                                $costImproving \leftarrow$ true
                                $maxRatio \leftarrow ratio$
                                $task \leftarrow t$
                                $predecessor \leftarrow p$
                                $receiver \leftarrow s$
        if $costImproving$ then          ▷ Move task into shorter route
            $receiver$.insert($task, predecessor$)
            $SC[0]$.remove($task$)
            $SC$.Sort()             ▷ Update the sorting for next iteration
        else
            return $SC$
```

6. EXPERIMENTS

The goal of the experiments is to compare the proposed heuristic planning approach with a MILP solver with respect to plan quality and planning performance. In our experiments, the operational area where the missions are planned is 40×50 m, containing 468 task locations and 72 robot locations. For each experiment, a mission work description is generated in the form of the RTSG model in Figure 1. The model is populated with a randomly selected subset of the tasks in the operational area, where the number of tasks is a controlled parameter. 20% of the tasks are modeled to be alternative, partitioned in randomly composed alternative sets with 2-4 tasks in each set. Delivery tasks are co-located. Eight randomized robot locations indicate possible robot team members to be selected, where the team size is a controlled parameter. Routing distances are Euclidian and routing costs, i.e., routing times, are estimated with a robot speed of 0.5m/s. Task action durations are randomized in the interval 5-15s. The *makespan* is minimized by the two compared planning approaches, while *planning time* is a corresponding performance measure. 10 experiments were run for each combination of team size and the number of tasks. The planning time was limited to a maximum of 30 minutes for the MILP solver. It can be noted that the problem is NP-hard, since it generalizes the NP-hard TSP problem (Ilavarasi and Joseph, 2014).

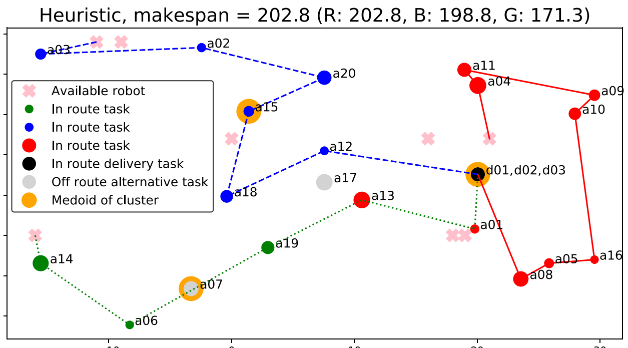
The generated plans for one such experiment is illustrated in Figure 2, where the diagrams indicate $2D$ locations in the operational area. Units of axes are meter. The heuristic solution is visualized in the upper diagram and the MILP solver solution in the lower diagram. Available robots are marked with pink X-markers and tasks are marked with O-markers. Each route starts from a selected robot (an

X-marker) and ends with one of the black-colored and co-located delivery tasks ($d01$,$d02$,$d03$). Action durations are indicated by the size of the O-markers. The color of non-selected alternative tasks is grey, i.e., $a07$ and $a17$ for the heuristic solution, and $a04$ and $a15$ for the MILP solution. For the heuristic approach, the medoids of the clusters are highlighted in orange and the colors of planned tasks indicate their clusters. The balancing step has moved two red tasks ($a1$,$a13$) from the red route to the green route.

The experiments were run on an Intel i5-4570 with 8 GB of RAM and Ubuntu 20.04.5 operating system. Gurobi (Gurobi Optimization, 2021) was used to compute MILP solutions. For the heuristic approach, Concorde (Applegate et al., 2011) was used to compute TSP solutions while the remaining algorithmic steps, e.g., clustering, were implemented in Python.

A comparison of the heuristic approach and the MILP solver is found in Fig. 3. The horizontal axes indicate the number of tasks. In the top graph, the average computed makespan is given for the MILP solver and for the heuristic approach for different team sizes. For the MILP solver, a few makespan values are marked with a red dot. They indicate problem instances where the MILP solver was able to find optimal solutions for at least 5 out of 10 runs. To give a quantitative indication of the optimality of the heuristic approach, only the runs with optimal MILP solver solutions are included in the data set for these problem instances. The middle graph indicates the average planning time, and the bottom graph gives the Planning-time to Makespan Ratio (PMR).

The heuristic approach generated solutions with a slightly higher makespan than the MILP solver for the smallest problem instances, while performing better than the MILP



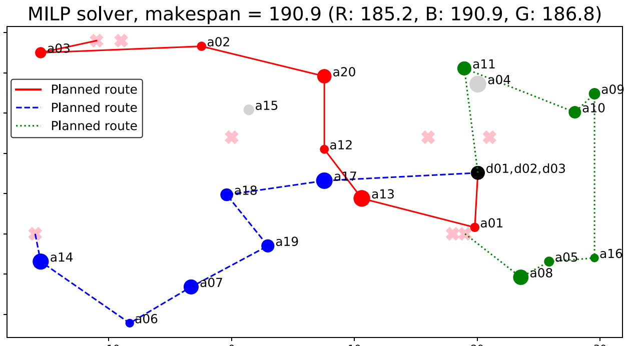


Fig. 2. Illustrated example of computed problem solutions.

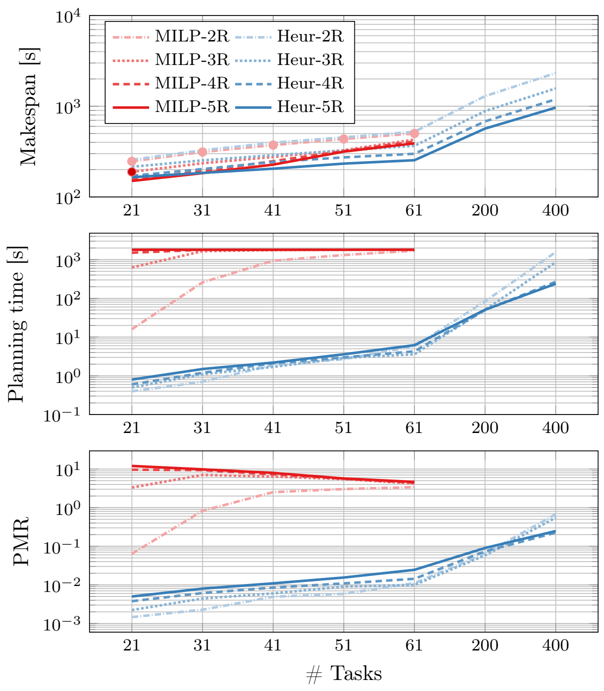


Fig. 3. Makespan and planning time for different sizes of robot teams, displayed with logarithmic scales to provide a compact view.

solver for the larger problem instances. The heuristic approach found solutions within a few seconds, while the MILP solver delivered an optimal solution in less than 30 minutes, or a sub-optimal solution at the 30 minute timeout. The PMR comparison in the bottom graph indicates that the MILP solver in general uses significantly more time for producing a plan compared to the makespan of that plan. This implies the MILP solver is a less suitable approach, especially in online planning scenarios where the planning time may have a direct impact on the mission time. On the other hand, the heuristic approach has a planning time that is a fraction of the makespan. With 200 tasks, the planning time is still reasonable with PMR $<$ 10%. Some quantitative data of the solution optimality for the heuristic approach can be indicated with a Makespan Optimality Ratio (MOR):

$$MOR = \frac{Makespan_{heur}}{Makespan_{opt}} \qquad (27)$$

$MOR \geq 1$, where a value of 1 indicates an optimal solution. MOR of the heuristic approach can be evaluated for the problem instances with optimal MILP solver solutions, i.e., $Makespan_{opt} = Makespan_{MILP}$. For problem sizes with two robots, MOR was $104\%, 105\%, 106\%, 104\%$ and 104% for $21, 31, 41, 51$ and 61 tasks, respectively. For 3 robots, MOR was 113% for 21 tasks. These are all sub-optimal solutions, but indicate an acceptable gap to optimality, especially when considering the superior planning time compared to the MILP solver. However, the evaluation is quantitative and we do not provide a guarantee on the optimality. For the largest problem instances with 200 and 400 tasks, the MILP solver was unable to find any feasible solution in the given time. With 400 tasks, the planning time of the heuristic approach increases with smaller robot teams. This is caused by TSP computations becoming the dominating sub-problem, where a smaller team size scales up the size of the TSP problems.

7. CONCLUSION

We have investigated a novel heuristic approach to select and plan a multi robot team for an industrial kitting application modeled with a Robot Task Scheduling Graph. It is benchmarked against a MILP model implemented in Gurobi, that is able to generate optimal solutions for smaller problem instances. The experiments confirm an ability to generate high quality solutions within a few seconds, i.e., a fraction of the time required by the MILP solver. Additionally, solutions can be generated within reasonable time for significantly scaled up problem instances.

Future extensions of this work may investigate, e.g., Cross-Schedule Dependencies and mission planning in a dynamic environment.

REFERENCES

Applegate, D.L., Bixby, R.E., Chvátal, V., and Cook, W.J. (2011). The traveling salesman problem.

Ayari, A. and Bouamama, S. (2019). ACD3GPSO: automatic clustering-based algorithm for multi-robot task allocation using dynamic distributed double-guided particle swarm optimization. *Assem. Autom.*, 40(2).

Bahalke, U., Hamta, N., Shojaeifard, A.R., Alimoradi, M., and Rabiee, S. (2022). A new heuristic algorithm for multi vehicle routing problem with and/or-type precedence constraints and hard time windows. *Op. Res. in Eng. Sciences: Theory & Applications*, 5(2), 28–60.

Basu, S., Davidson, I., and Wagstaff, K. (2008). *Constrained clustering: Advances in algorithms, theory, and applications.* CRC Press.

Bortfeldt, A. and Wäscher, G. (2013). Constraints in container loading: A state-of-the-art review. *Europ. J. Op. Res.*, 229(1), 1–20.

Cheikhrouhou, O. and Khoufi, I. (2021). A comprehensive survey on the multiple traveling salesman problem: Applications, approaches and taxonomy. *Computer Science Review*, 40, 100369.

Dantzig, G.B. and Ramser, J.H. (1959). The truck dispatching problem. *Management science*, 6(1), 80–91.

Gerkey, B.P. and Matarić, M.J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The Int. Journal of Robotics Research*, 23(9), 939–954.

Goel, A. and Gruhn, V. (2008). A general vehicle routing problem. *Europ. J. Op. Res.*, 191(3), 650–660.

Gurobi Optimization, L. (2021). Gurobi optimizer reference manual. URL `http://www.gurobi.com`.

Ilavarasi, K. and Joseph, K.S. (2014). Variants of travelling salesman problem: A survey. In *Int. Conf. on Inf. Comm. and Emb. Syst. (ICICES)*, 1–7.

Jain, A.K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8), 651–666.

Jonker, R. and Volgenant, T. (1983). Transforming asymmetric into symmetric traveling salesman problems. *Op. Res. Letters*, 2(4), 161–163.

Khamis, A., Hussein, A., and Elmogy, A. (2015). Multi-robot task allocation: A review of the state-of-the-art. *Cooperative robots and sensor networks 2015*, 31–51.

Konstantakopoulos, G.D., Gayialis, S.P., and Kechagias, E.P. (2020). Vehicle routing problem and related algorithms for logistics distribution: A literature review and classification. *Op. Res.*, 22(3), 2033–2062.

Korsah, G.A., Stentz, A., and Dias, M.B. (2013). A comprehensive taxonomy for multi-robot task allocation. *The Int. J. Rob. Res.*, 32(12), 1495–1512.

Lager, A., Papadopoulos, A., Spampinato, G., and Nolte, T. (2021). A task modelling formalism for industrial mobile robot applications. In *Int. Conf. on Adv. Rob. (ICAR)*.

Lahyani, R., Khemakhem, M., and Semet, F. (2015). Rich vehicle routing problems: From a taxonomy to a definition. *Europ. J. Op. Res.*, 241(1), 1–14.

Laporte, G. and Semet, F. (1999). Computational evaluation of a transformation procedure for the symmetric generalized traveling salesman problem. *INFOR: Information Systems and Op. Res.*, 37(2), 114–120.

Mathew, N., Smith, S.L., and Waslander, S.L. (2015). Multirobot rendezvous planning for recharging in persistent tasks. *IEEE Trans. Robotics*, 31(1), 128–142.

Miloradović, B., Cürüklü, B., Ekström, M., and Papadopoulos, A.V. (2019a). A genetic algorithm approach to multi-agent mission planning problems. In *Int. Conf. Op. Res. and Enterp. Syst.*, 109–134.

Miloradović, B., Frasheri, M., Cürüklü, B., Ekström, M., and Papadopoulos, A.V. (2019b). Tamer: Task allocation in multi-robot systems through an entity-relationship model. In *Int. Conf. Principles and Practice of Multi-Agent Systems*, 478–486.

Montoya-Torres, J.R., Franco, J.L., Isaza, S.N., Jiménez, H.F., and Herazo-Padilla, N. (2015). A literature review on the vehicle routing problem with multiple depots. *Computers & Ind. Eng.*, 79, 115–129.

Murugappan, E., Subramanian, N., Rahman, S., Goh, M., and Chan, H.K. (2021). Performance analysis of clustering methods for balanced multi-robot task allocations. *Int. J. Prod. Res.*, 60(14), 4576–4591.

Nunes, E., Manner, M., Mitiche, H., and Gini, M. (2017). A taxonomy for task allocation problems with temporal and ordering constraints. *Rob. & Aut. Syst.*, 90, 55–70.

Park, H.S. and Jun, C.H. (2009). A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2), 3336–3341.

Randel, R., Aloise, D., Mladenović, N., and Hansen, P. (2019). On the k-medoids model for semi-supervised clustering. In *Variable Neighborhood Search*, 13–27.

Resende, M.G.C. and Werneck, R.F. (2007). A fast swap-based local search procedure for location problems. *Annals of Op. Res.*, 150(1), 205–230.

Roohnavazfar, M., Pasandideh, S.H.R., and Tadei, R. (2022). A hybrid algorithm for the vehicle routing problem with and/or precedence constraints and time windows. *Computers & Op. Res.*, 143, 105766.

Touzani, H., Hadj-Abdelkader, H., Séguy, N., and Bouchafa, S. (2021). Multi-robot task sequencing & automatic path planning for cycle time optimization: Application for car production line. *IEEE Rob. & Autom. Lett.*, 6(2), 1335–1342.

Xu, X., Yuan, H., Liptrott, M., and Trovati, M. (2017). Two phase heuristic algorithm for the multiple-travelling salesman problem. *Soft Computing*, 22(19), 6567–6581.