Investigation of Real-Time Task Scheduling on Robot Fleets with Reconfigurable Actuators

Trevor Robin Smith†
Department of Engineering Physics
McMaster University
Hamilton, Canada
Email: tsmith@ieee.org

Spencer Ploeger†
School of Engineering
University of Guelph
Guelph, Canada
Email: sploeger@ieee.org

Benjamin Dyer
School of Engineering
University of Guelph
Guelph, Canada
Email: dyerb@uoguelph.ca

†Equal contribution authors.

2021 IEEE International Systems Conference (SysCon) | 978-1-6654-4439-2/20/$31.00 ©2021 IEEE | DOI: 10.1109/SysCon48628.2021.9447134

***Abstract*—Multi-Fleet Scheduling (MFS) is concerned with the issue of assigning tasks to a swarm of mobile robotic agents. In this paper, MFS of tasks using a novel class of mobile agents with reconfigurable modular actuators is proposed and analyzed. MFS is split into two regimes, static and dynamic, where the static regime does not allow real-time reconfiguration of agent actuators. Most pre-existing robotic agents are compatible with the static multi-fleet scheduling (S-MFS) regime, whereas the novel agents being investigated here are capable of using dynamic multi-fleet scheduling (D-MFS). Solutions to both problems are compared, and it is shown that in the worst case scenario, given some set of agents and tasks available at known start times, D-MFS finds the same optimal schedule as S-MFS, whereas D-MFS can be used to find more optimal solutions in some conditions. It is also shown that D-MFS may not always be optimal depending on the arrival of previously unknown a-periodic tasks, as D-MFS provides the optimal schedule for a specific fleet of robots accomplishing a set of tasks for some scheduling algorithm and cost function. By defining and exploring the D-MFS problem, this work paves the way for future investigations in task-prediction, efficient large-scale scheduling algorithms, and novel robot manufacturing capabilities.**

***Index Terms*—systems, scheduling, multi-fleet, reconfigurable, actuators, end-effectors**

I. Introduction

Adaptive manufacturing practices provide companies the flexibility to rapidly react to volatility in market conditions, workforce size, and other important decision-making factors. Such practices are often restricted by incompatibility with existing infrastructure as not all machines can easily be repurposed in real-time. One approach to achieving flexible manufacturing processes is implementing a fleet of reconfigurable robotic agents [1]. This work explores scheduling tasks for such a fleet, with and without the ability to change actuators in real-time for potentially faster task completion.

This work contributes to literature by highlighting a novel method to schedule fleets of mobile, reconfigurable agents through the reformulation of Multi-Fleet Scheduling (MFS) into two different regimes (static-MFS and dynamic-MFS), the development of preliminary methods to solve both problems, and insight towards further potential improvements. Previous research in fleet-scheduling and agile robotics are first discussed. A model for an agent is defined and real-time fleet scheduling is formulated as an optimization problem. A scheme for finding the optimal solution given some cost function and scheduling algorithm is found and a simulation use-case is shown to highlight potential advantages of D-MFS. While previous research has studied task-scheduling for a fleet of mobile robots, to the best knowledge of the authors, this is the first to investigate the scheduling of agents which can change their functionality in real-time through switching actuators.

II. Literature Review

A. Agile Robotics

The authors of [1] propose a complete solution for modular end-effector robots and matching docking stations to facilitate the exchange of end-effectors and charging. This solution is an ideal platform for investigating task scheduling in mobile, reconfigurable agents as it provides the essential building blocks for agile manufacturing processes. The focus of the aforementioned work was on the design of the modular end-effectors and docking stations. Task-scheduling in [1] was beyond the scope of the work, thus a First-In-First-Out (FIFO) priority queue was used for scheduling. This leaves significant opportunities to develop frameworks and algorithms for efficient scheduling of such robots in an agile manufacturing environment, as existing methods do not fully utilize the design flexibility.

B. Scheduling

To effectively utilize fleets of mobile agents, efficient scheduling algorithms must exist to find optimal task-schedules in real-time. Although the class of agents discussed in this work are novel - and as such, scheduling algorithms have not been investigated previously - significant work has been done in fleet-scheduling and scheduling in general.

Real-world scheduling problems are found in situations such as work force management and mobile robot fulfillment systems (MRFS). Many large distribution centers such as Amazon and Alibaba use MRFS in order to lift movable racks and transport them to work stations [2]. The use of MRFS as a warehousing system has been investigated using queuing models and was shown to be effective in a variety of workshop sizes and shapes [3]. Task scheduling can be applied to the service industries such as banks, police departments, fast food, and airport ground stations through the use of personnel cross-training in order to increase workforce flexibility [4]. Through the use of task scheduling, adaptive manufacturing processes can be investigated and implemented on large scale. Meng et. al. developed distributed flexible job shop scheduling processes (DFJSP) in order to represent a multi-factory environment in which each factory acts as a flexible job shop capable of changing tasks, while each task is given to exactly one factory [5].

A subset of scheduling research focuses specifically on fleet scheduling. Work by Li et. al. investigated multistage heterogeneous fleet scheduling while considering the size of the fleets that could be scheduled [6]. In doing so the scheduler is capable of integrating vehicle allocation and fleet size while considering the routing of the fleet and the fleet vehicle types. Nag et. al. developed schedule optimization for earth observations using a constellation of orbiting Cubesats. Through the use of optimal scheduling cubesats can be made to quickly and effectively make measurements at short notice while considering cloud cover predictions, ground down-link windows, or any other positional or temporal constraints on each Cubesat [7].

Xidias et. al. investigated an approach for task scheduling a fleet of vehicles in a factory environment. Their approach focused on representing the environment using a single mathematical entity which could then be combined with path planning and task scheduling to create an optimization problem [8]. The near optimal solution was found using a modified Genetic Algorithm in order to attain safe, efficient, and feasible paths for the fleet to follow while maintaining a desired work rate. Kousi et. al. also focused on the intra-factory environment developing a service-based control system executed by autonomous mobile units capable of transporting consumables from a warehouse to production stations [9]. de Matta et. al. developed work schedules for an inter-city transit system. In order to meet the daily service requirements of an inter-city bus transit firm, the best mix of primary and secondary jobs in short term work schedules utilizing multiple fleet types was investigated resulting in a larger solution space with potentially more optimal results [10].

III. Problem Formulation

In this section, the problem of flexible multi-fleet scheduling is explored and a framework is developed for initial investigation into the problem. An *agent* is a robot available as a resource to accomplish tasks in some defined environment. Building upon the design introduced in [1], agents are modelled as robots with interchangeable actuators using docking stations that dual as actuator storage.

The set of robots organized in a *fleet*, F, is defined as a list of n sets, where there are $n-1$ different actuator types. Each element of the list stores the robot list of a particular type, called *sub-fleet* i or F_i, where the ith element corresponds to the set of agents with actuators of type i and sub-fleet F_0 corresponds to the set of agents without actuators. A *task*, τ_j, is a single command that can be interpreted by a robotic agent, R_k - the kth agent in a fleet, such that R_k can autonomously complete τ_j given adequate available resources. Using a user-defined cost function, $C(R_k, \tau_j)$, the cost of R_k completing τ_j can be determined.

Scheduling a set of tasks, denoted with τ, is not a trivial task. If F_i is the sub-fleet corresponding to the necessary actuator type for completing τ_j, the optimal available agent in sub-fleet F_i can be assigned the task such that $R_k = \min\{C(F_i, \tau_j)\}$, where R_k is the optimal agent belonging to F_i. In this work, the approach mentioned previously will be referred to as *Static-MFS*, as the solution set is locally bounded to the F_i sub-fleet due to the static configuration of agent actuators. A classical Static-MFS algorithm is shown by Algorithm 1. In the event that $\min\{C(F, \tau_j)\} \neq \min\{C(F_i, \tau_j)\}$, expanding the search to include the global set F can yield more satisfactory results. Typically, such global searches are redundant if actuators are not re-configurable, as the task could not be accomplished with any amount of time unless the agent were to be initially configured with the required actuator. This leads to the introduction of a novel problem - *Dynamic-MFS*.

A. Dynamic-MFS Problem

The Dynamic-MFS problem can be formulated as follows: Given some set of tasks, τ, where $\tau = \{\tau_1, \tau_2, ..., \tau_k\}$, find some efficient schedule that satisfies all elements of τ such that $\Sigma C(\tau_i)$ is minimized. The differences between the static and dynamic regimes lie in the system constraints.

Constraints are imposed through a user-defined cost function. In the S-MFS regime, any agent without a task-compatible actuator is equivalent to having an infinite cost of completion. This makes it computationally redundant to search the entire fleet when looking for an optimal agent, as all agents excluded from the task-compatible sub-fleet will never be optimal - in the static regime all non-compatible actuator agents would have an infinite cost of completion. In the D-MFS regime, any agent can complete any task at the expense of a more complex cost function that incorporates a penalty for changing actuator types at a docking station. Assuming identical conditions at scheduling, the D-MFS will at worst find the same solution as the S-MFS, as the static regime's solution-space is a subspace of the dynamic regime's solution-space.

Although D-MFS has potentially many more solutions than it's static counterpart, finding these solutions is a challenging task. Solving the problem by brute-force can work on small-scale systems, but checking all possible schedule combinations may not be feasible if the number of agents in the sub-fleet is magnitudes less than that of the global fleet. For D-MFS to be feasible on the large-scale, efficient methods for optimizing agents on the large-scale must be investigated, developed and tested. This provides motivations for future work on the subject.

IV. Solution for Dynamic-MFS

A heuristic method has been developed (see Algorithm 2) to utilize the modular end-effectors for switching agents between sub-fleets in real-time, allowing for more optimal fleet scheduling in some cases. Additionally, the worst case scenario of the proposed method yields the equivalent result to the S-MFS solution. An example that illustrates this situation is when a task requiring a specific actuator-type arrives but there are no available agents.

In a static fleet, a task cannot be scheduled if no agents belonging to the corresponding sub-fleet are available. In this case, the task is scheduled to the agent in the sub-fleet that can complete the task fastest after completing their current task. This is the fastest solution if agents cannot be reconfigured, however, with reconfigurable agents, D-MFS may find a more optimal solution by expanding the agents being searched. The decision process for static fleets can be seen in Figure 1.

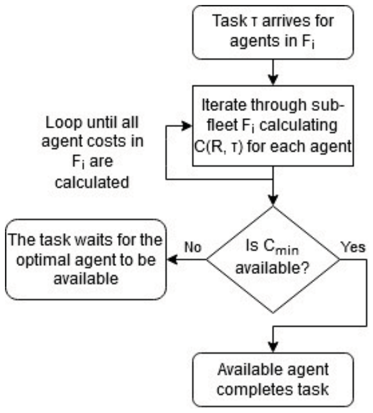
Task τ arrives for agents in F_i
Iterate through sub-fleet F_i calculating C(R, τ) for each agent
Loop until all agent costs in F_i are calculated
Is C_min available?
No
Yes
The task waits for the optimal agent to be available
Available agent completes task

Fig. 1. Flowchart Static-MFS

A dynamic fleet uses the same scheduler as static fleets, but differs in cost function. Static regime cost functions yield infinite cost for agents with incompatible actuators, whereas the dynamic regime uses a cost proportional to the time required to change actuators. This means that every agent R_i in the fleet F is evaluated in determining scheduling of tasks, regardless of whether they possess the necessary actuator. In the event that the optimal agent is not in the sub-fleet of actuator-type required by the task, the optimal agent moves to the docking station, switches actuators to that required of the task, and performs the task. This only occurs when the cost algorithm determines an actuator-change is optimal rather than waiting for an already-configured agent to complete the task, as the algorithms are identical except S-MFS is a local search of D-MFS. This algorithm capitalizes on robots that would otherwise sit idle by switching their actuator in order to complete tasks in a way to minimize total cost of the system. The decision process for dynamic fleets can be seen in Figure 2.

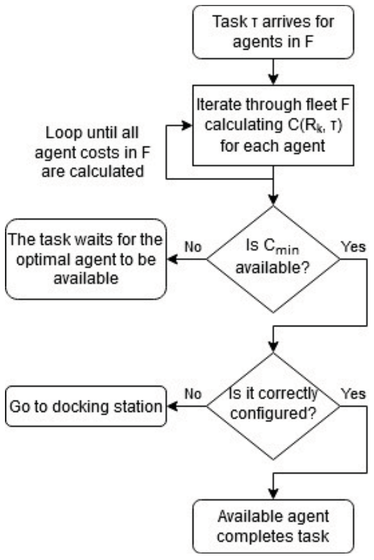
Task τ arrives for agents in F
Iterate through fleet F calculating C(R_k, τ) for each agent
Loop until all agent costs in F are calculated
Is C_min available?
No
Yes
The task waits for the optimal agent to be available
Is it correctly configured?
No
Yes
Go to docking station
Available agent completes task

Fig. 2. Flowchart Dynamic-MFS

V. Results

Two different sets of tasks, displayed in Tables I and II, have been simulated to highlight the potential effects of using S-MFS and D-MFS in the fixed environment in Figure 3.

A simple cost function based on time was used, where $C(R_k, \tau_i) = C_{switch} + C_{move} + C_{task}$. Agent motion is modelled as ideal straight paths with a cost of 1 unit distance per unit time ($C_{move} = distance$ units), and the cost to change actuators was fixed to 3 time units ($C_{switch} = 3$ units).

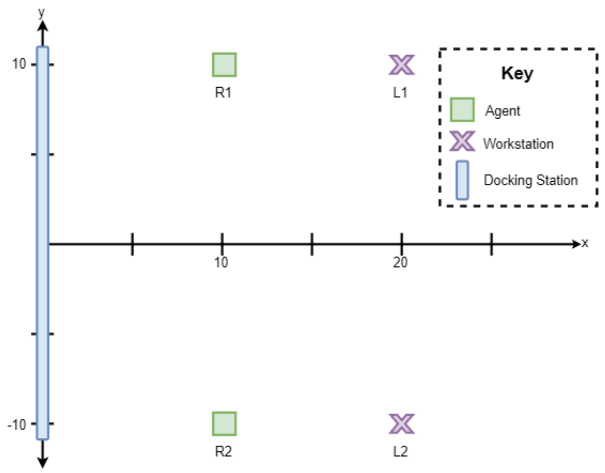

Fig. 3. Robot environment used for simulations

Task Number	T1	T2	T3	T4	T5
Time of Arrival	0	0	15	35	40
Actuator Type	A	A	B	A	B
Workstation	L1	L2	L1	L2	L1
Task Duration	20	20	15	20	15

TABLE I
Set 1: List of tasks and requirements

Task Number	T1	T2	T3	T4	T5
Time of Arrival	0	10	25	50	60
Actuator Type	A	A	A	B	B
Workstation	L1	L2	L1	L2	L1
Task Duration	20	20	20	15	15

TABLE II
SET 2: LIST OF TASKS AND REQUIREMENTS

Set 1 has been chosen to highlight an important situation; when S-MFS results in a more efficient solution than D-MFS. This results from the fact that D-MFS can only ensure optimal scheduling for tasks known at the time of scheduling. T3, T4, and T5 arrive after one round of scheduling has already occurred, however, S-MFS and D-MFS have already deviated from one another and thus, the two different problems have different initial conditions at $t = 15$, leading to D-MFS performing worse. Set 2, contrary to Set 1, yields a faster schedule in D-MFS, as can be seen in Table III.

This is worth noting, as it gives motivation to exploring potential task-anticipation schemes for improving the reliability of D-MFS over S-MFS. With accurate predictions - or when dealing with periodic tasks - this is not problematic, as D-MFS can consistently perform at worst the same as S-MFS, but a-periodic tasks have the potential to make D-MFS perform worse than S-MFS.

A heuristic search algorithm is displayed below to find the optimal agent to complete a task given some fleet. This method assesses the optimal cost agent for a task given some fleet, as D-MFS searches all possible solutions for the optimal schedule, not simply evaluating the schedules for sub-fleet F_i. In this algorithm described, it is assumed that an ideal scheduler picked a task as to be completed next, and the algorithm must find which agent can complete the task optimally.

Algorithm 1: Classical Static-MFS Solution

```
Result: $R_{min}$ in $F_i$ for a S-MFS Task $\tau_i$
$C_{min} = \infty$ ;        ▷ Initialize as infinity
for $R_k$ in $F_i$ ;   ▷ Loop through all agents
 of sub-fleet
do
    if $R_k$ is available;        ▷ If agent with
     desired actuator is free
    then
        Calculate $C_k for R_k$;
        if $C_k < C_{min}$;       ▷ New current best
        then
            $C_{min} = C_k$;
            $R_{min} = R_k$;
        end
    end
end
```

Algorithm 2: Preliminary Dynamic-MFS Solution

```
Result: $R_{min}$ in $F$ for a D-MFS task $\tau_i$
$C_{min} = C_{wait}$ ;    ▷ Initialize as infinity
for $R_k$ in $F$ ;          ▷ Loop through fleet
do
    Calculate $C_{switch} for R_k$;
    if $C_{switch} < C_{min}$;      ▷ New current best
    then
        $C_{min} = C_{switch}$;
        $R_{min} = R_k$;
    end
end
```

VI. CONCLUSION

This work reformulates the Multi-Fleet Scheduling (MFS) problem into two different regimes: Static-MFS and Dynamic-MFS. Static-MFS (S-MFS) is for environments where sub-fleets of different types cannot be reconfigured, whereas in Dynamic-MFS (D-MFS), agents can change sub-fleets. It is shown that S-MFS is a restricted form of D-MFS, where a single sub-fleet is searched instead of the entire fleet. This means that, given some cost function and set of tasks with arrival times, D-MFS will at worst find the same schedule as S-MFS, but at best find a more optimal schedule.

It is also shown that although D-MFS yields at worst an equivalent schedule to S-MFS, D-MFS cannot guarantee optimal results in the event of non-predicted, a-periodic tasks. Resulting from potentially non-identical initial conditions when the schedule is re-calculated to include the new task, further research is needed to investigate methods for efficiently solving the Dynamic Multi-Fleet Scheduling problem with a-periodic tasks.

	Set 1		Set 2	
Time	**Static**	**Dynamic**	**Static**	**Dynamic**
10	R1 begins T1	R1 begins T1 R2 arrives at dock	R1 begins T1	R1 begins T1
13		R2 leaves dock with A		
20				R2 arrives at dock
23				R2 leaves dock with A
30	R1 completes T1	R1 completes T1	R1 completes T1	R1 completes T1 R1 begins T3
33		R2 arrives at L2 with A R2 begins T2		
37.36	R2 begins T3			
43				R2 begins T2
50	R1 begins T2	R1 arrives at dock	R1 begins T2	R1 completes T3
52.36	R2 completes T3 R2 begins T5			
53		R1 leaves dock with B R2 completes T2 R2 begins T4		
60			R2 arrives at L2 R2 waits	
63				R2 completes T2
67.36	R2 completes T5			
70	R1 completes T2 R1 begins T4		R1 completes T2 R2 begins T4	R1 arrives at dock
73		R1 begins T3 R2 completes T4		R1 leaves dock with B
83				R2 arrives at dock
85			R2 completes T4	
86				R2 leaves dock with B
88		R1 completes T3 R1 begins T5		
90	R1 completes T4		R1 begins T3	
93				R1 arrives at L1 R1 begins T5
103		R1 completes T5		
105			R2 arrives at L1, waits	
106				R2 arrives at L2 R2 begins T4
108				R1 completes T5
110			R1 completes T3 R2 begins T5	
121				R2 completes T4
125			R2 completes T5	

TABLE III
SIMULATION RESULTS FOR R1 AND R2 COMPLETING TASK SETS 1 AND 2

REFERENCES

[1] T. R. Smith, B. Thompson, J. Balfour, and A. Taher, "Modular End-Effector on Mobile Robot with Automated Change Station," *2020 4th International Conference on Robotics and Automation Sciences, ICRAS 2020*, pp. 34–38, 2020.

[2] W. Yuan and H. Sun, "A task scheduling problem in mobile robot fulfillment systems," *12th International Conference on Advanced Computational Intelligence, ICACI 2020*, pp. 391–396, 2020.

[3] T. Lamballais, D. Roy, and M. B. De Koster, "Estimating performance in a Robotic Mobile Fulfillment System," *European Journal of Operational Research*, vol. 256, no. 3, pp. 976–990, 2017.

[4] E. Peters, R. de Matta, and W. Boe, "Short-term work scheduling with job assignment flexibility for a multi-fleet transport system," *European Journal of Operational Research*, vol. 180, no. 1, pp. 82–98, 2007.

[5] L. Meng, C. Zhang, Y. Ren, B. Zhang, and C. Lv, "Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem," *Computers and Industrial Engineering*, vol. 142, no. February, p. 106347, 2020.

[6] B. Li, X. Yang, and H. Xuan, "A Hybrid Simulated Annealing Heuristic for Multistage Heterogeneous Fleet Scheduling with Fleet Sizing Decisions," *Journal of Advanced Transportation*, vol. 2019, 2019.

[7] S. Nag, A. S. Li, and J. H. Merrick, "Scheduling algorithms for rapid imaging using agile Cubesat constellations," *Advances in Space Research*, vol. 61, no. 3, pp. 891–913, 2018.

[8] E. Xidias, P. Zacharia, and A. Nearchou, "Path Planning and scheduling for a fleet of autonomous vehicles," *Robotica*, vol. 34, no. 10, pp. 2257–2273, 2016.

[9] N. Kousi, S. Koukas, G. Michalos, and S. Makris, "Scheduling of smart intra–factory material supply operations using mobile robots," *International Journal of Production Research*, vol. 57, no. 3, pp. 801–814, 2019.

[10] R. de Matta and E. Peters, "Developing work schedules for an inter-city transit system with multiple driver types and fleet types," *European Journal of Operational Research*, vol. 192, no. 3, pp. 852–865, 2009.