Optimization of Multi-Agent Workflow for Human-Robot Collaboration in Assembly Manufacturing

Ronald J. Wilcox* and Julie A. Shah†

Massachusetts Institute of Technology, Cambridge, MA, 02139, USA

Human-robot collaboration presents an opportunity to improve the efficiency of manufacturing and assembly processes, particularly for aerospace manufacturing where tight integration and variability in the build process make physical isolation of robotic-only work challenging. In this paper, we develop a robotic scheduling and control capability that adapts to the changing preferences of a human co-worker or supervisor while providing strong guarantees for synchronization and timing of activities. This formulation is then expanded to optimize the workflow of a team of robots according to a set of qualitative and quantitative spatial and temporal constraints and performance objectives. We describe the Adaptive Preferences Algorithm that computes the optimal flexible scheduling policy for task completion that meets hard temporal constraints. We use APA within a mixed integer multi-agent optimization algorithm that assigns a flexible schedule of agents to tasks. We show that execution of the Advanced Preferences Algorithm is fast, robust, and adaptable to changing preferences for workflow and that the multi-agent optimization, while slower, is practically useful for important applications in multi-robot assembly of large structures for aerospace manufacturing. We specifically demonstrate the capability for quick reoptimization of a plan in response to temporal disturbances in the schedule and changing high-level guidance from a human supervisor.

I. Introduction

Substantial workflow benefits in assembly and manufacturing tasks can be realized if robotic teams have the ability to work in concert with each other or with a human worker. For example, a robotic team that is able to rearrange its schedule to account for a robot failure can mitigate productivity loss due to the breakdown. Traditionally, human workers and robots work in isolation from one another, but a large increase in efficiency may be achieved if humans and robots are allowed to work in the same vicinity. For example, quality assurance teams can inspect work being completed in real time if the robots have the ability to resequence work dynamically to keep at a safe distance from the people.

In this paper, we present a robotic scheduling and control capability for human-robot collaborative work that addresses several key challenges in the assembly manufacturing environment. First, introducing humans to a traditionally robot-only space on the factory floor also introduces a large degree of unpredictability to the system; many manual assembly and manufacturing processes in the aerospace industry, for example, grant freedom to the worker to decide how best to accomplish a task. A high level of adaptability and robustness must therefore be built into any robotic system that works in close collaboration with people.

Second, human and robotic work in manufacturing and assembly must meet hard scheduling constraints, including pulse rates between build stations and flow rates for end-to-end assembly. The changing preferences of a human co-worker or supervisor must be accommodated while preserving strong guarantees for synchronization and timing of activities.

Third, a centralized controller must schedule all agents effectively to meet qualitative and quantitative spatial and temporal requirements on workflow. For example, the system must guarantee for safety that there be a buffer region around each robot so that an unexpected malfunction will not harm a person or damage another robot. Work must be coordinated amongst various agents to maximize efficiency while satisfying these and other hard constraints.

Our technical approach generalizes from dynamic scheduling methods [2, 7, 12]. Dynamic scheduling is domain independent and has been successfully applied to scheduling within the avionics processor of commercial aircraft [12], autonomous air vehicles [11], robot walking [4], and recently, human-robot teaming [9, 10, 13]. We leverage

*Graduate Student, Department of Mechanical Engineering.

†Assistant Professor, Department of Aeronautics and Astronautics.

prior art that addresses efficient real-time scheduling of plans whose temporal constraints are described as Simple Temporal Problems (STPs) [2, 7, 12]. STPs compactly encode the set of feasible scheduling policies for plan events that are related through simple interval temporal constraints. Temporal flexibility in the STP provides robustness to disturbances at execution.

We make use of this simple yet powerful framework to model joint human-robot work as a Simple Temporal Problem with soft constraints (also called preferences). The soft constraints encode person-specific workflow patterns and human operator input for suggested workflow. Simple Temporal Problems with Preferences (STPPs) have been studied previously [5, 6, 8] for weakest-link optimization criteria, but these solution techniques do not generalize to optimization criteria relevant to manufacturing applications. Alternatively, an STPP with arbitrary objective function may be formulated and solved as a non-linear program (NLP), where the solution is an assignment of execution times to each event in the plan. This approach results in brittle solutions; any disturbance in execution time requires time-consuming re-calculation of the schedule. In this work, we leverage the strengths of STP and NLP solution methods: flexibility in execution and optimization of arbitrary objective functions, respectively. We review the Adaptive Preferences Algorithm (APA) [13] that uses the output of a non-linear program solver to compute a flexible optimal scheduling policy that accommodates temporal disturbance. The algorithm also supports on-the-fly optimization in response to changing preferences.

The Adaptive Preferences Algorithm is built on a solely temporal framework and thus cannot handle important applications containing spatial constraints or preferences; it also does not support task assignment, restricting useful applications to those of a human worker and a single robotic assistant. We therefore proceed to expand upon these foundations to create a system capable of scheduling multiple agents while providing temporal and spatial guarantees as well as optimizing various productivity-based objectives. We describe the Multi-Agent Temporal Optimization Algorithm (MATOA) that makes use of a mixed integer quadratic program (MIQP) to optimize according to objectives relevant to manufacturing and then leverages the flexibility of STP scheduling in a manner similar to APA to provide a flexible multi-agent schedule. Finally, we demonstrate in simulation that the integration of APA and MATOA allows for a controller capable of controlling multiple robots under an assortment of different objectives and constraints that provides the flexibility, adaptability, and robustness required for human-robot collaboration.

In Section II we discuss examples of human-robot interaction that motivate our work. Next, in Section III we briefly review STP dynamic scheduling, optimization of preferences, and basic mixed integer modeling techniques. In Sections IV and V we present both the Adaptive Preferences Algorithm (APA) and the Multi-Agent Temporal Optimization Algorithm (MATOA) and their evaluations. We show empirical results indicating both that APA provides significant robustness to temporal disturbance and that a robot using APA can adaptively schedule its actions over a horizon of 50 activities with sub-second speed; we also provide empirical results detailing that MATOA provides flexible, agent-assigned schedules with 10 activities in approximately 10 seconds, but that high computation time renders large activity sets infeasible for MATOA. Finally in Section VI, we demonstrate the rescheduling of a robot team in response to the breakdown of one robot and the ability of the robot team to respond to changing high level guidance from a human supervisor.

II. Motivating Examples

In this section, we discuss two applications that motivate our work: the malfunction of a single robot in a robotic team, and the disruption of a process by a Quality Assurance agent.

A. Robot Breakdown

We aim to develop a capability that supports efficient redistribution of work in response to a disturbance. In aerospace assembly manufacturing, many of the end-effectors equipped on robots are new, specialized technology and are prone to frequent breakdown. Often, the entire multi-robot system is halted to repair one robot, leading to work slowdowns and lost time. Instead, our approach enables the multi-robot system to respond to the malfunction by automatically shifting work and resequencing tasks among the remaining robots. Further, we aim to take advantage of data mining techniques that allows one to predict how long a robot will be down based on the type of malfunction which has occurred. This data can be used to predict a time window for the robot's return and plan accordingly to direct other robots to pick up the slack of the broken one.

B. Quality Assurance Interruption

We also aim for our capability to enable a single operator to direct a team of robots, while ensuring that hard scheduling deadlines such as mandated flow rates are met. Individually commanding robots is inefficient. Instead, we aim to develop a control system whereby an operator can add a preference on-the-fly to an existing plan to provide real-time high level guidance on the workflow. The robots would then reconfigure the task assignment and schedule while still guaranteeing that all hard temporal and spatial constraints are met.

Our approach supports the ability for a supervisor to specify, for example, that work on the aft part of the fuselage be delayed by a certain amount of time to provide a safe working environment for the quality inspection team. Work will be shifted according to operator preferences through fast re-computation of the robots' schedule, while minimizing the amount of lost time.

III. Background

The robotic scheduling and control capability we develop builds on previous work in STP dynamic scheduling, the optimization of STPs with Preferences (STPPs), and mixed integer programming.

A. Dynamic Scheduling of STPs

A Simple Temporal Problem (STP) [2] consists of a set of executable events, X. These events are connected via binary temporal constraints (intervals) b_{ij} that indicate a range for the amount of time passing between events X_i and X_j.

A *solution* to an STP is a time assignment to each event such that all constraints are satisfied. An STP is said to be *consistent* if at least one solution exists. Checking an STP for consistency can be cast as an all-pairs shortest path problem. The STP is consistent if and only if there are no negative cycles in the all-pairs distance graph. This check can be performed in $O(n^3)$ time by applying the Floyd-Warshall algorithm [2].

The all-pairs shortest path graph of a consistent STP is also a *dispatchable form* of the STP, enabling flexible real-time scheduling [7]. The dispatchable STP provides a compact representation of the set of feasible schedules. Dynamic scheduling of the dispatchable STP provides a strategy that schedules events online just before they are executed, with a guarantee that the resulting schedule satisfies the temporal constraints of the plan. Scheduling events on-the-fly allows the robot to adapt to temporal disturbance associated with past events through linear-time constraint propagation. More formally, a network is *dispatchable* if for each variable X_A it is possible to arbitrarily pick a time t within its timebounds and find feasible execution windows in the future for other variables through one-step constraint propagation of the X_A temporal commitment.

B. Prior art in STP Preference Optimization

An STP with Preferences (STPP) [6] is a Simple Temporal Problem with the addition of soft binary constraints, or preference functions, $f_{b_{ij}}(t)$, governing the temporal durations between events. The global preference function, F, of an STPP represents the objective function derived from the preference values based on a time assignment to each event. An optimal solution to the STPP is consistent with the binary temporal constraints b_{ij} and optimizes the global preference function F.

Preferences provide an expressive and natural framework for encoding human input. A supervisor may apply preference functions to specify the most effective timing for an activity without providing hard constraints that lead to schedule brittleness. STPPs were originally developed to perform scheduling for Earth observation satellites [5] where fairly allotting time to different research teams was a priority, leading to use of a weakest-link optimization criterion. Solution methods, including a slow constraint propagation technique and fast binary chop method [8], have been designed for this weakest link optimization criterion but do not readily generalize to other objective functions.

Fairness is not a concern in the optimization of a manufacturing process. It is acceptable to sacrifice one interval's preference value to improve the preference values for many other intervals (e.g. slow down one robot so that it does not block the path for the other robots). For example, for many manufacturing applications, an approach that optimizes the STPP with respect to the sum of preference values, $\sum_{b_{ij}} f_{b_{ij}}(t)$, is more appropriate. An STPP with arbitrary objective function may be formulated and solved as a non-linear program (NLP), where the solution is an assignment of execution times to each event in the plan. However, this approach results in brittle solutions; any disturbance in execution time requires time-consuming re-calculation of the schedule. In Section A, we briefly review a method for computing a temporally flexible optimal scheduling policy that leverages the strengths of STP and NLP

solution methods. The Adaptive Preferences Algorithm [13] (APA) computes a flexible optimal scheduling policy that accommodates fluctuations in execution time and supports robust online optimization in response to changing preferences. In Subsection 3, we discuss the integration of APA with a mixed-integer quadratic program to compute flexible multi-agent schedules.

IV. Modeling

A. The Adaptive Preferences Algorithm

In this section, we briefly review the Adaptive Preferences Algorithm [13] that leverages STP and NLP solution methods to compute a flexible, optimal scheduling policy. The Adaptive Preferences Algorithm (APA) takes as input a Simple Temporal Problem with Preferences (STPP), composed of

- a set of variables, $X_1, ...X_n$, representing executable events,
- a set of binary temporal constraints of the form b_{ij} that encode activity durations and qualitative and quantitative temporal relations between events X_i and X_j,
- a set of preferences functions of the form $f_{b_{ij}}(t)$ that encode preference values over the temporal interval b_{ij} , and
- a global objective criterion F defined as a function of the preferences functions $f_{b_{ij}}(t)$. We use $F = \sum_{b_{ij}} f_{b_{ij}}(t)$ for prototyping of the described manufacturing applications, although note APA generalizes to other forms of the objective function.

The output of the algorithm is a *dispatchably optimal* (DO) form of the STPP that supports fast dynamic scheduling. We define an STPP as dispatchably optimal if it is possible to maximize the global preference function F through the following procedure: for each variable X_i it is possible to arbitrarily pick a time t within the DO form's timebounds and find feasible execution windows in the future for other variables through fast one-step constraint propagation of the X_i temporal commitment.

Notice that the proposed problem may be formulated as a non-linear optimization problem to solve for event execution times. This approach provides a solution that is brittle to disturbance, requiring recomputation when an event does not execute at precisely the specified time. In contrast, our approach compiles a temporally flexible optimal scheduling policy that accommodates fluctuations in execution time. This method leverages the insight that there are potentially many schedules that are consistent with an optimal time assignment to preference functions. Subsection 1 reviews the compilation algorithm that computes the DO form for the STPP. Subsection 2 reviews the dispatcher algorithm that generates a schedule using the STPP DO form, and supports robust online reoptimization in response to changing preferences.

1. Compiler for STPP DO Form

The Compiler takes as input a STPP composed of events X_i, constraints b_{ij}, and preference functions $f_{b_{ij}}(t)$. The output is a DO plan, which encodes a flexible scheduling policy that maximizes the global preference function F subject to the given binary temporal constraints b_{ij}.

Pseudocode for the compilation algorithm is provided in Fig. 1. The given STPP is first reformulated for optimization through an all-pairs-shortest-path computation to expose all implicit constraints (Line 1). Redundant constraints are then removed (Line 2). The resulting network is provided as input to a nonlinear program; variables represent the timepoints of the network, and interval constraints relate the timepoints. The form of the objective function is chosen based on the requirements of the particular application. The optimization software then optimizes (Line 3) the given problem and outputs a set of timestamps that maximize the objective function subject to the constraints.

```
function APAcompilePlan(STPP plan)
1.  STP compiled_plan = perform APSP( plan)
2.  compiled_plan = prune redundant edges(compiled_plan)
3.  optimal_execution_times = new NLP Solver(compiled_plan)
4.  given_prefs = gather constraints with preferences (plan);
5.  for(each interval b'{ij} in compiled_plan)
6.    if( there exists a constraint relating events Xi and Xj in given_prefs)
7.      set b'{ij} to difference in optimal_execution_times[Xj-Xi];
8.    end if
9.  end for
10. perform APSP (compiled_plan);
11. return compiled_plan;
```

Figure 1. Pseudocode for the compilation algorithm

As mentioned, the timestamped solution provides brittle schedules, so APA instead uses the timestamps returned by the optimizer to guide a tightening of the original network while maintaining flexibility. In Lines 5-9, each interval with a preference function is tightened to the timestamp value with a small tolerance built in; Floyd Warshall's algorithm is then performed again to tighten the network around these optimal values while maintaining flexibility in most of the intervals without preferences (Line 10).

In the next section, we discuss the method for dispatching the DO form of the STPP.

2. *Dispatcher*

```
function STPPdispatch(STP compiled_plan, STPP orig_plan)
1.  Enabled = {first event}; Executed = {}
2.   current_time = 0
3.  new thread STPdispatch(orig_plan)
4.  while(size of Executed < number of events)
5.    if(new preferences or deviation from optimal schedule)
6.        switch execution control to STPdispatch thread
7.        orig_plan' = replace past intervals with rigid links(orig_plan)
8.        compiled_plan = compilePlan(orig_plan')
9         switch execution control to STPPdispatch
10.   end if
11.   for(each event e in plan)
12.     if(Executed does not contain e)
13.       if( robot signals event has been performed)
14.         add event and execution time to Executed
15.         remove event from Enabled
16.         event_executed = true
17.       end if
18.       if(event e is in Enabled)
19.         Interval bounds = extract 'liveness' bounds for e
20.         if( bounds lowerbound < current _time < bounds upperbound)
21.           signal robot to execute event e
22.         end if
23.       end if
24.       if(event_executed)
25.         event_executed = false;
26.         Enabled = gather enabled events
27.         propagate event commitment to compute liveness windows
28.         wait for next live event or until robot signals an executed event
29.       end if
30.     end if
31.   end for
```

Figure 2. Pseudocode for the dispatching algorithm

In this section, we present a dispatcher algorithm that supports two functions: the dispatcher (**function1**) generates a schedule using the STPP DO form, and (**function2**) supports robust online reoptimization in response to changing preferences.

The dispatcher takes as input an STP *compiled_plan* that encodes the DO form of an STPP S. The dispatcher schedules events on-the-fly just before they are executed while guaranteeing that the resulting schedule satisfies the temporal constraints of the plan. This guarantee is achieved through constraint propagation of temporal commitments to executed events. The output of the dispatcher is an assignment of event execution times that optimizes the STPP S global preference value F, subject to the given temporal constraints of S.

The dispatcher also supports robust online replanning of the DO form, in response to changing preference functions and disturbances in the optimal execution. In these situations, the DO form must be recompiled by calling the algorithm **APAcompilePlan** with the modified STPP S'. As discussed in Section V, this recompilation takes on the order of seconds for moderately-sized real-world problems.

Function1 of the dispatcher is achieved using the standard STP dispatching algorithm [7]. **Function2** is achieved by augmenting the STP dispatching algorithm with two additional methods. The first method triggers recompilation for changing preference functions or deviations from the optimal schedule, and the second method runs as a concurrent process to ensure the dispatcher makes progress during recompilation and that the execution schedule satisfies the hard constraints of the STPP S.

Fig. 2 presents the STPP dispatching algorithm. Augmentations to the standard STP dispatching algorithm are highlighted; these additions allow for the implementation of **Function2**. The reader is referred to [13] for a full description of the dispatching algorithm.

The resulting schedule maximizes the global preference value and satisfies the temporal constraints of the problem. The signal-and-response structure (signal in Line 21 and robot response in Line 13) provides robustness in execution by allowing for situations that prevent the robot from completing the task at precisely the specified time.

The compiler and dispatcher presented in Figs. 1 and 2 have been implemented and tested successfully. Section V presents an empirical evaluation of APA. Next, we describe the Multi-Agent Temporal Optimization Algorithm (MATOA); we show how to model useful performance objectives and spatial and temporal constraints in quadratic programming form and how to use the output to compute a robust, flexible schedule.

3. *Evaluation*

We now review the empirical evaluation of the compilation time of APA and the flexibility it allows in its solutions. For a full discussion of these results, the reader is referred to [13]. Cumulative compilation time for the inflexible NLP approach scales with the number of events in the plan, whereas the STPP DO approach scales with the number

of preference functions. The result is that the STPP DO form provides on average an 80% reduction in cumulative compilation time as shown in Figure 3.

Next, we compute a measure of the temporal flexibility in the STPP-DO form and the NLP solution, as compared to the temporal flexibility of the original STPP. We compiled 50 random problems and compared the resulting interval durations to the original problems' interval durations. This ratio then represents the percentage of flexibility retained from the original problem by the compiled DO form. Higher values of this ratio correspond to an increased robustness to disturbances during execution. We compare this to the flexibility ratio for the NLP-specified schedule on the same 50 problems. The DO form captures on average more than 75% of the temporal flexibility in the original plan, whereas the NLP solution captures less than 1%; these results are relatively independent of problem size. The DO form provides a marked improvement in robustness to disturbance over the NLP solution while achieving global optimization of the schedule. These results provide sufficient capability for one-to-one human-robot collaboration, indicating a robot can adaptively schedule its actions over a horizon of approximately 100 activities with sub-second speed.

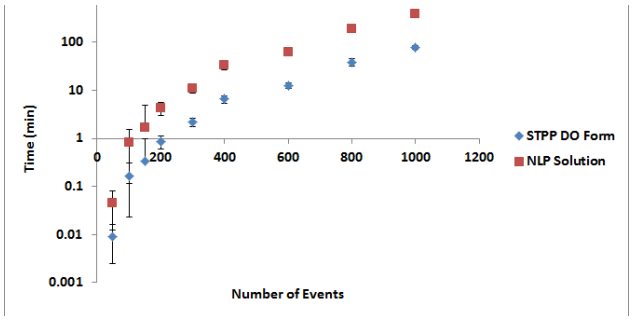

Figure 3. Cumulative Compilation Time as a function of the number of events in the plan

B. Multi-Agent Optimization Algorithm

In this section, we describe the schedule optimization of multiple agents according to various qualitative and quantitative spatial and temporal constraints and performance objectives, modeled as a mixed integer quadratic program. As input, our Multi-Agent Temporal Optimization Algorithm (MATOA) takes a STP encoding the temporal constraints of the problem, a spatial grid representing the positions of the various work packages, and a previous agent allocation detailing where the agents were assigned in the previous iteration of the algorithm. As output, MATOA returns an assignment of each work package to an agent and a flexible schedule of when each work package should be executed.

We have modeled objectives and constraints applicable to assembly manufacturing. The objective function, Obj, includes terms that minimize the difference from the previous agent assignment to the returned agent assignment, the number of spatial interfaces between work packages that two different agents have completed, and the overall idle time of the system. We create constraints ensuring that (1) temporal requirements are met, (2) each work package is assigned to one agent, (3) agent capabilities and limitations (in terms of temporal constraints on task completion) are taken into account, (4) agents maintain safe buffer distances between each other, and (5) that schedules produced are temporally consistent. Table 1 presents the binary and continuous decision variables of the model.

Next, we describe the objectives and constraints and their efficient, quadratic mathematical formulations.

Variable	Properties	Description
A_{aj}	Binary	Indicates whether agent a performs work package j
J_{ij}	Binary	Indicates whether work package i is performed before work package j
T_e	Continuous	Indicates at what time event e is performed

Table 1. Table 1: Descriptions of the Decision Variables used by MATOA

1. *Objective Modeling*

The objective function, Obj, show in Eq(1), is composed of a weighted sum of three terms, each corresponding to a different goal. The weights are arbitrary and allow the objectives to be valued differently based on the specific application.

$$Obj = \alpha \times D + \beta \times Int + \gamma \times Idle \tag{1}$$

In manufacturing environments with humans and robots working together, it is crucial to maintain predictability of the robotic system to support human worker trust and situational awareness. We therefore do not want the system oscillating between equally optimal solutions. For this reason, we minimize D, the difference between agent allocation, where P_{ai} is the value of A_{ai} from the previous solution:

$$D = \sum_{a,i} (A_{ai} - P_{ai})^2 \tag{2}$$

Inter-robot accuracy is challenging for multi-robot systems of standard industrial robots. For example, in robot painting, this can lead to gaps or overlaps at interfaces between work done by two different robots. Therefore we minimize the number of spatial interfaces using the following formulation, where S is the set of all work packages (i, j) that are spatially adjacent:

$$Int = \sum_{a} \sum_{(i,j) \in S} (A_{ai} - A_{aj})^2 \tag{3}$$

We next minimize the agent idle time. This both maximizes the efficiency of the robot system and, for mixed human-robot teams, is beneficial from a human factors perspective. A few intermediate variables are required; the first is a variable $Both_{aij}$ indicating whether agent a performs both work packages i and j, computed as the conjunction $A_{ai}\ AND\ A_{aj}$. The idle time between two work packages is the difference between the assigned time of the last event of the first work package and the assigned time of the first event of the second work package. This works well in the case where we know the ordering of the work packages (for example, if they have a required delay between them), but to include the idle time between work packages which are allowed to occur in any order, we create two new variables combining the two possible sequencing cases. We have $Combo_{aij} = Both_{aij} AND J_{ij}$ and $AltCombo_{aij} = Both_{aij} AND (NOT J_{ij})$, where J is the decision variable governing the order in which two unordered work packages are chosen to occur; $J = 1$ if work package i occurs before work package j. Using these variables, we can find the total idle time of the system. We separately sum the idle times between ordered (below, represented by set O) and unordered (below, represented by the set U) work packages in the following manner:

$$\begin{aligned} Idle = &\sum_{a} \sum_{(i,j) \in O} Both_{aij} \times (T_{starteventof j} - T_{finisheventof i}) + \\ &\sum_{a} \sum_{(i,j) \in U} Combo_{aij} \times (T_{starteventof j} - T_{finisheventof i}) + \\ &\sum_{a} \sum_{(i,j) \in U} AltCombo_{aij} \times (T_{starteventof i} - T_{finisheventof j}) \end{aligned}$$

The weighted sum of these three terms in Eq(1) composes the objective function which we seek to minimize.

2. *Constraint Modeling*

Constraints are included in the model to ensure that various requirements in the manufacturing environment are satisfied. The first of these are mandatory deadlines and delays, or temporal requirements. These are encoded within an STP and built into the model in an identical way to the Adaptive Preferences Algorithm, ensuring that the differences between all event times fall within the interval bounds in the STP. The second requirement states rather intuitively that all work packages must be completed, and that one agent executes each work package. This corresponds to the mathematical requirement that, for all work packages i:

$$\sum_{a} A_{ai} = 1 \tag{4}$$

We next take into account the capabilities and limitations of the various agents. The events associated with the start and finish of the work package must be assigned to occur within the times given by the agent's capabilities. Every agent input has an interval, $[a_{lower}, a_{upper}]$ indicating its capabilities. If this agent is assigned a work package, these bounds must be obeyed. We make use of the common bigM formulation of mixed-integer programming [3], where M takes on a very large (theoretically infinite) value to activate or relax the constraints based on the value of the binary decision variable. Thus, we add the following constraints to the model for all agents a and work packages i:

$$a_{lower} - M(1 - A_{ai}) \leq T_{finish\ event\ of\ i} - T_{start\ event\ of\ i} \tag{5}$$

$$a_{upper} + M(1 - A_{ai}) \geq T_{finish\ event\ of\ i} - T_{start\ event\ of\ i} \tag{6}$$

Safety for both robots and people dictates that there be a buffer zone around each robot that another agent cannot enter; this mitigates the effect of unexpected malfunctions or motions. For this reason, we include constraints preventing a robot from being assigned a task while another agent is working on a task directly adjacent to it. We allow an

agent to enter a space within a minimum time of $travel$ of another agent leaving it. We again use the set S of adjacent work packages to formulate, for all pairs of work packages i and j within S:

$$T_{start\,event\,of\,j} - T_{finish\,event\,of\,i} \geq travel - MJ_{ij} \tag{7}$$

$$T_{start\,event\,of\,i} - T_{finish\,event\,of\,j} \geq travel - M(1 - J_{ij}) \tag{8}$$

Finally, we formulate sequencing constraints to ensure that no agent is assigned to do two work packages at once. These constraints also assign an order to work packages that are originally unordered and include two big M terms, one to enforce which order work packages will occur, and another to apply the constraints only if the same agent is working on both work packages; these two conditions are similar to the objective formulation of the Idle time. We also again make use of the buffer time $travel$. We apply these constraints for all agents a and work packages i and j:

$$T_{start\,event\,of\,j} - T_{finish\,event\,of\,i} \geq travel - MJ_{ij} - M(2 - A_{ai} - A_{aj}) \tag{9}$$

$$T_{start\,event\,of\,i} - T_{finish\,event\,of\,j} \geq travel - M(1 - J_{ij}) - M(2 - A_{ai} - A_{aj}) \tag{10}$$

The optimizer minimizes the objectives subject to these constraints and returns the optimal agent allocation, sequencing, and time schedule. In a similar manner to the Adaptive Preferences Algorithm, we use these returned values to tighten the network while attempting to maintain as much flexibility as possible so as not to create brittle solutions. This is discussed in the next subsection.

3. Processing and Integration

In order to effectively use the STP dispatching algorithm outlined in above, we integrate the returned optimized variables from MATOA into the original STP. First, all intervals corresponding to the work packages are tightened to the capabilities of the agent assigned to that work package. For example, a painting work package required to be completed between 2 and 8 hours for quality reasons may be tightened to 4 to 6 hours to account for the fact that the robot assigned to it cannot complete it faster than 4 hours and will not take more than 6 hours. Agent assignments are associated with each work package interval and are output through the dispatcher when the start event of that work package is executed. The sequencing selected by the optimizer is then enforced, so any work packages that were unordered in the original STP are tightened to have a positive lower bound on their connecting intervals. Finally, an all-pairs-shortest-path computation is applied to expose implicit constraints in the network based on these modifications. This process outputs a dispatchable form of the network that provides a temporally flexible scheduling policy for the multi-agent system.

There are three options for combining the two algorithms developed in this work, APA and MATOA: (1) executing APA before MATOA, (2) after MATOA, or (3) integrating the preferences themselves into MATOA. Performing APA before MATOA (1) causes one to weight the preference functions higher than objectives built into MATOA since APA tightens the resulting network (which would be input to MATOA). This is a useful choice for some applications where the preferences drive the desired behavior, but in some circumstances it can lead to infeasibilities when the preferences are optimized to regions out of the agents' inherent capabilities.

An alternative method for integration performs MATOA before APA (2), ensuring feasibility of the agent selection process and then optimizing the preferences of the network around these sequencing choices. This leads to guaranteed feasibility of the agent allocation (given a feasibly designed input problem), but can lead to suboptimal solutions for the preferences since the sequencing decisions of MATOA can potentially lead the network to be tightened away from the true optimal of the input STPP.

The final choice for combination involves replacing the idle time objective in MATOA with preference functions (3), since idle time is essentially a preference to pull all intervals to their shortest possible duration. One can then tighten the network around the preferences, sequencing, and agent capabilities in the processing phase of the algorithm. All three possibilities have strengths and weaknesses and can be useful for different applications.

V. Multi-Agent Optimization Algorithm Evaluation

The Multi-Agent Optimization Algorithm schedules multiple agents to perform a set of tasks subject to various constraints and objectives. In this section, we evaluate the speed of compilation and the flexibility preserved by the post processing performed on the output from the third-party optimizer.

Empirical results are produced using a random problem generator that creates structured problems using as input the number n of work packages, the number of user-specified constraints between work packages c, and the number of agents a. The output of the generator is an STP, a spatial grid of work packages, a list of agents and their capabilities, and a set of previous agent assignments which are provided as input to the compiler. The MATOA compiler and random problem generator are implemented in Java, and non-linear programs are solved using the Java implementation of Gurobi [1]. Results are generated using an Intel Core i7-2620M 2.70 GHz Processor.

First we evaluated the optimization time for plans with a number of work packages varying in the range [5, 10]. Ten work packages was the maximum number that could reliably be schedule in less than 30 minutes. Next, we ran tests on plans with 10 work packages to determine the impact of the number of agents and the number of user-specified constraints between work packages. These results are presented in Figure 4. We find that the addition of constraints decreases the compilation time; this is because the sequencing decisions, which only occur for work packages without constraints between them, constitute the most substantive contribute to compilation time. We found the number of agents available to execute the work packages did not change the compilation time significantly, although theoretically more agents leads to higher compilation times because of the addition of binary variables to the MIQP.

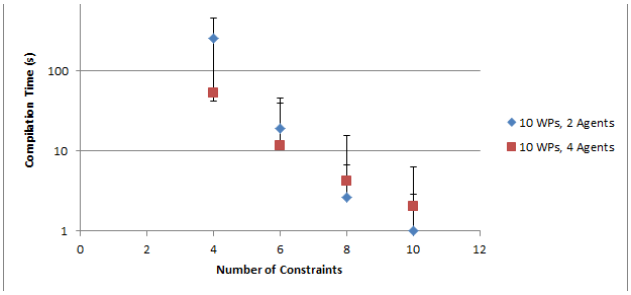

Figure 4. The effect on compilation time of temporal constraints for plans with 10 workpackages with 2 and 4 agents. There were 50 plans generated for each constraint data point; mean and standard deviation indicated

We also evaluated the flexibility gained by running the post-processing on the specific times yielded by the MIQP optimizer. We use a measure of flexibility similar to that used for APA where the percentage of flexibility is the sum of the amount of time in each interval of the compiled plan divided by sum of the amount of time in each interval of the original plan. Theoretically, because many intervals are changed from unordered to ordered because of single-agent sequencing limitations, we expect a decrease in flexibility compared to the APA results; however, the direct use of optimizer-returned times leads to much more brittle and rigid schedules than MATOA. We found that MATOA allows for 40% of the flexibility of the original problem as compared to less than 1% for the MIQP; this ratio is relatively constant through all problem sizes.

Although MATOA is still too slow for large problems, it does capture a reasonable amount of flexibility. In the next section, we demonstrate the ability of MATOA to schedule multiple agents for two important factory applications.

VI. Robotic Demonstration

A. Robot Malfunction

We demonstrate MATOA to optimize a system of two robots assembling a large structure. The plan involves executing six work packages. Figure 5 shows the setup of the system; work packages are denoted by the stripes and are numbered left to right. The first, third, fourth, and sixth work packages each take 5 seconds to perform, and the second and fifth work packages each take 2.5 seconds. Each robot takes 1 second to move between stations. The robots are commanded to finish all work packages within 20 seconds. The video of this execution can be found at a http://youtu.be/QGhlcK1kFB0. After 5 seconds the Left Robot breaks down and requires 8 seconds for repair. MATOA computes a new plan in response to this disturbance using the additional constraint that the Left Robot may not perform any activity during the next 8 seconds. MATOA then re-allocates work packages to robots and re-sequences the work to finish within the 20 seconds allotted.

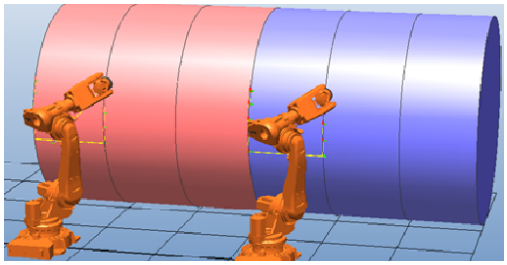

Figure 5. Demonstration Set-up

Specifically, the Right Robot is sent immediately to the second work package to ensure that it "picks up the slack" for the Right Robot while guaranteeing the robots maintain safe distances between each other. This complex behavior arises with the addition of a single constraint that prevents the Left Robot from performing any work for 8 seconds after a malfunction. and demonstrates that MATOA enables on-the-fly adjustment of agent assignments and schedules.

B. Quality Assurance Interruption

The Adaptive Preferences Algorithm (APA) and Multi-Agent Temporal Optimization Algorithm (MATOA) have been successfully integrated to create a system that is capable of quick reoptimization in response to changing operator preferences. In the following video http://youtu.be/3ewBl55llmc two robots work together to execute twelve work packages. Each work package take 10 seconds to complete. After the first work package is completed, a quality assurance agent adds a preference that no work be done on the left half of the work piece for the next twenty seconds, so s/he can inspect the progress. APA is run before MATOA to apply a preference that the robots vacate the left side of the fuselage for as close to twenty seconds as possible. Next, MATOA is applied to optimize the idle time of the multi-robot team. In this way, the command of a single operator is capable of modifying the behavior of an entire team of robots, as desired.

VII. Conclusions and Future Work

In this paper, we introduce the Multi-Agent Temporal Optimization Algorithm (MATOA) that creates an optimal agent allocation and schedule for a team of robots and review the Adaptive Preferences Algorithm (APA) that computes a flexible optimal policy for robot scheduling. We show through empirical evaluation that APA is fast, adaptable and robust to uncertainty in execution, and supports interaction with human co-workers and supervisors whose preferences about task completion are prone to change; MATOA is limited in its computation speed by the combinatoric growth of task sequencing. We also demonstrate the use of MATOA and APA in simulation and show that the robots successfully adapt to both malfunctions of a single robot and commands by a single operator. As future work, we aim to improve computation time of MATOA to support multi-robot orchestration problems involving hundreds of work packages.

VIII. Acknowledgements

This work was funded in part by Boeing Research and Technology.

References

[1] Gurobi optimizer version 5.0, April 2012.

[2] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artif. Intell.*, 49(1):61–91, 1991.

[3] F. Hillier and G. Lieberman. *Introduction to Operations Research*. McGraw-Hill, 1995.

[4] A. Hofmann and B.Williams. Robust Execution of Temporally Flexible Plans for Bipedal Walking Devices. In *Proc. ICAPS*, pages 386–389, 2006.

[5] L. Khatib, P. Morris, R. Morris, and F. Rossi. Temporal constraint reasoning with preferences. In *Proc. IJCAI*, pages 322–327, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-812-5, 978-1-558-60812-2. URL `http://dl.acm.org/citation.cfm?id=1642090.1642135`.

[6] P. Morris, R. Morris, L. Khatib, S. Ramakrishnan, and A. Bachmann. Strategies for global optimization of temporal preferences. In *Proc. CP*, pages 408–422. Springer, 2004.

[7] N. Muscettola, P. Morris, and I. Tsamardinos. Reformulating Temporal Plans For Efficient Execution. In *Proc. KRR*, 1998.

[8] F. Rossi, K. B. Venable, L. Khatib, P. Morris, and R. Morris. Two Solvers for Tractable Temporal Constraints With Preferences. In *Proc. AAAI workshop on preference in AI and CP*, 2002.

[9] J. Shah. *Fluid Coordination of Human-Robot Teams*. MIT PhD Thesis, Cambridge, Massachusetts, 2010.

[10] J. Shah, J. Wiken, B. Williams, and C. Breazeal. Improved Human-Robot Team Performance Using Chaski, a Human-inspired Plan Execution System. In *Proc. ACM/IEEE HRI*, pages 29–36, 2011.

[11] J. Stedl. Managing Temporal Uncertainty Under Limited Communication: A Formal Model of Tight and Loose Team Communication. Master's thesis, MIT, 2004.

[12] I. Tsamardinos and N. Muscettola. Fast transformation of temporal plans for efficient execution. In *Proc. AAAI*, 1998.

[13] R. Wilcox, S. Nikolaidis, and J. Shah. Optimization of temporal dynamics for adaptive human-robot interaction in assembly manufacturing. In *Proc. RSS*, 2012.