Co-Optimizating Multi-Agent Placement with Task Assignment and Scheduling

Chongjie Zhang and **Julie A. Shah**
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
{chongjie,julie_a_shah}@csail.mit.edu

Abstract

To enable large-scale multi-agent coordination under temporal and spatial constraints, we formulate it as a multi-level optimization problem and develop a multi-abstraction search approach for co-optimizing agent placement with task assignment and scheduling. This approach begins with a highly abstract agent placement problem and the rapid computation of an initial solution, which is then improved upon using a hill climbing algorithm for a less abstract problem; finally, the solution is fine-tuned within the original problem space. Empirical results demonstrate that this multi-abstraction approach significantly outperforms a conventional hill climbing algorithm and an approximate mixed-integer linear programming approach.

Introduction

Task assignment and scheduling are central problems for multi-agent systems, as they determine which agent should execute which task and at what time in order to optimize system performance. This multi-agent coordination problem with temporal and spatial constraints can be formulated as a mixed integer linear programming (MILP) [Brucker, 2001] and is NP-Hard [Bertsimas and Weismantel, 2005]. Various approaches have been proposed to address this challenging problem [Hooker, 2009; Korsah *et al.*, 2012; Gombolay *et al.*, 2013]; however, these existing approaches do not scale well for many practical problems, which usually incorporate more than 20 agents and 200 tasks.

Fortunately, many large-scale problems commonly possess a nearly-decomposable structure, where agents can be grouped according to their capabilities and tasks can be clustered by their spatial or temporal constraints so that constraints between tasks in different clusters can be aggregated and modeled as high-level constraints between clusters. For example, in a production-line factory as shown in Figure 1, agents (e.g., robots and human workers) are grouped by their specialties and tasks are naturally clustered according to the workplaces at which they are located. For the sake of simplicity, we use *workplaces* to refer to task clusters. Constraints between tasks in different workplaces are modeled as precedence constraints between workplaces.

To exploit such nearly-decomposable structure and enable large-scale multi-agent coordination, this paper abstracts a high-level *multi-agent placement* problem, which determines how many agents of each type are allocated to each workplace in order to optimize overall system performance. The solution to this problem is used to decompose a large multi-agent system into a group of smaller subsystems, for which the task assignment and scheduling problem has exponentially less complexity and can be solved using existing approaches. However, this multi-agent placement problem is challenging because how agents are placed to workplaces depends on how tasks of each workplace are assigned and scheduled to them.

The contribution of this paper is twofold. First, we formulate multi-agent coordination as an integrated, multi-level optimization problem. This problem consists of agent placement (i.e., how agents are allocated to workplaces), task assignment (i.e., how tasks are assigned to the agents at each workplace), and task scheduling (i.e, how an agent performs its tasks). Second, we develop a multi-abstraction search approach (MASA) for co-optimizing multi-agent placement with task assignment and scheduling. MASA consists of three phases. It first begins with a highly abstract agent placement problem and rapidly computes an initial solution. This solution is then improved upon using a hill climbing algorithm by considering how task assignment at each workplace affects agent placement. Finally, MASA fine-tunes the improved agent placement solution by considering the effects from both task assignment and scheduling.

Search in abstract problems is much faster, as it considers much fewer variables and constraints. In addition, abstract problems may have smoother search space surfaces with less local optima. Therefore, the first two phases prepare a suitable initial solution for the final phase and allow it to avoid inferior local optima, converge to a better solution and improve the search speed. Experimental results indicate that MASA yields a more than 15% and 25% reduction in makespan compared with an MILP-based approximate approach and a conventional hill climbing algorithm, respectively. MASA also improves the convergence speed by more than 10% on average over the conventional hill climbing.

Multi-Level Optimization Problem

This section illustrates multi-agent coordination with temporal and spatial constraints in the manufacturing domain and

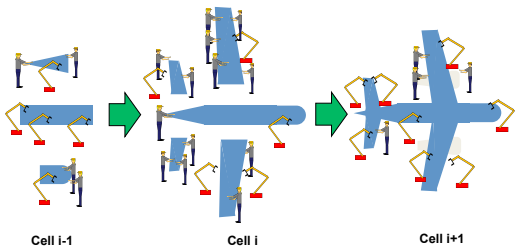

Figure 1: An assembly line with robots and human workers

formulates it as a multi-level optimization problem.

Illustrative Domain

Robotic systems are now commonplace on many manufacturing assembly lines. Figure 1 depicts a segment example of a cellular assembly line with a group of robots and human agents performing tasks. This assembly line consists of a sequence of workplaces, each of which consists of a set of tasks. There are precedence constraints between neighboring workplaces. A product will move to a next workplace when all tasks associated with the current workplace are completed and the product is not fully assembled. There are upper and lower bounds on task completion and temporal constraints relating tasks (e.g. "the first coat of paint requires 30 minutes to dry before the second coat may be applied"). The different agent types have different capabilities for performing tasks (e.g., how fast to complete a task). In addition, there are spatial restrictions on agent proximity (e.g. robots must always operate at least 5 feet from other agents) to support safe and efficient multi-robot or human-robot collaboration. The goal is to maximize the assembly line's throughput.

Problem Formulation

We formulate such multi-agent coordination problem as a three-level optimization problem: agent placement (i.e., allocating agents to workplaces), task assignment (i.e., assigning tasks to the agents in each workplace), and task scheduling (i.e, scheduling tasks for each agent to perform). The inputs of this multi-level optimization problem include:

- τ: A set of agent types
- $\langle n_1, \ldots, n_{|\tau|} \rangle$: The number of agents of each type
- W: A set of workplaces that agents are allocated to
- C: The set of agent capabilities specifying tasks each agent a may perform and its minimum $lb_{a,k}$, maximum $ub_{a,k}$, and expected time to complete each task k, and
- $P_w = \langle T_w, \Gamma_w, S_w \rangle$: A task problem for each workplace $w \in W$, where:
  - T_w: The set of all tasks to be performed.
  - Γ_w: a Simple Temporal Problem (TSP) [Dechter *et al.*, 1991], describing the temporal constraints relating tasks to one another at workplace w, and
  - S_w: A set of task pairs separated by less than the allowable spatial proximity, specifying spatial constraints.

The objective of this multi-level optimization problem is to find a solution (specifying agent placement, task assignment and task schedules for agents at each workplace) wherein all constraints are satisfied and the maximum makespan among workspaces is minimized. For an assembly line, this objective is equivalent to maximizing its throughput.

We formulate the multi-agent placement problem as below:

$$min_R \quad max(f_1^*(R_1, P_1), \ldots, f_{|W|}^*(R_{|W|}, P_{|W|})) \tag{1}$$

subject to

$$\sum_{w \in W} R_{w,x} \leq n_x, \forall x \in \tau \tag{2}$$

where $R_{w,x}$ is a decision variable representing the number of agents with type x assigned to workplace w, and $f_w^*(R_w, P_w)$ represents the *optimal makespan* of workplace w given a set of agents R_w. Equation 2 ensures that agent placement satisfies resource constraints.

To compute $f_w^*(R_w, P_w)$, we need to solve the task assignment and scheduling problem at workplace w with given agents R_w. In this paper, we mainly focus on the multi-agent placement problem and its interactions with lower-level task assignment and scheduling problems. For the sake of brevity, we formulate the task assignment and scheduling problem for each workplace w as a single MILP [Gombolay *et al.*, 2013]:

$$min\ f_w(A, J, t_{(\cdot)}^S, t_{(\cdot)}^E, R_w, P_w) \tag{3}$$

subject to

$$\sum_{a \in R_w} A_{a,k} = 1, \forall k \in T_w \tag{4}$$

$$lb_i \leq t_m - t_n \leq ub_i, \forall i \in T_w, n, m \in \Gamma_w \tag{5}$$

$$t_k^E - t_k^S \geq lb_{a,k} - M(1 - A_{a,k}), \forall k \in T_w, a \in R_w \tag{6}$$

$$t_k^E - t_k^S \leq ub_{a,k} + M(1 - A_{a,k}), \forall k \in T_w, a \in R_w \tag{7}$$

$$t_j^S - t_i^E \geq M(J_{i,j} - 1), \forall i, j \in S_w \tag{8}$$

$$t_i^S - t_j^E \geq -MJ_{i,j}, \forall i, j \in S_w \tag{9}$$

$$t_j^S - t_i^E \geq M(J_{i,j} - 1) + M(A_{a,i} + A_{a,j} - 2), \forall i, j \in T_w \tag{10}$$

$$t_i^S - t_j^E \geq -MJ_{i,j} + M(A_{a,i} + A_{a,j} - 2), \forall i, j \in T_w \tag{11}$$

$A_{a,k} \in \{0, 1\}$ is a binary decision variable for the assignment of agent a to task k. $J_{i,j}$ is a binary decision variable specifying the relative sequencing of two tasks, i and j ($J_{i,j}$ = 1 implies task i occurs before j). Decision variables t_k^S and t_k^E specify the start and end times, respectively, of task k. R_w is the set of all agents assigned to workplace w. M is a large positive number used to encode conditional constraints.

Equation 4 ensures that each task is assigned to a single agent. Equation 5 ensures that the temporal constraints are met, where bounds lb_i and ub_i are described in TSP Γ_w. Equations 6 & 7 ensure that agents are not required to complete tasks more quickly or slowly than they are capable. Equations 8 & 9 sequence actions to ensure that agents maintain safe buffer distances from one another while performing tasks. Equations 10 & 11 ensure that each agent only performs one task at a time. Note that Equations 8 and 9 couple the variables relating task sequencing, spatial locations and task start and end times, resulting in tight dependencies among agents' schedules.

The objective function $f_w(A, J, t^S_{(\cdot)}, t^E_{(\cdot)}, R_w, P_w)$ represents the makespan of workplace w. For all workplaces $w \in W$, the optimal makespan $f^*_w(R_w, P_w) \equiv min\ f_w(A, J, t^S_{(\cdot)}, t^E_{(\cdot)}, R_w, P_w)$ with all constraints satisfied.

From the formulation described above, we can observe the interdependency between multi-agent placement and task assignment and scheduling. The solution of the agent placement problem instantiates the task assignment and scheduling problems at workplaces, while computing the optimal agent placement solution depends on optimal solutions of the task assignment and scheduling problems at workplaces. In the next section, we will present a multi-abstraction approach to solve this nested problem.

Multi-Abstraction Search Approach (MASA)

The key challenge of the multi-level optimization problem is that optimizing multi-agent placement depends on solutions of a combinatorially explosive number of task assignment and scheduling problems, each of which is NP-hard and computationally expensive, even when being solved approximately [Lipovetzky *et al.*, 2014]. One conventional approach is to create an easily-computed objective function to approximate the objective function (1), detaching the agent placement problem from lower-level task assignment and scheduling problems. This top-down decomposition approach is computationally feasible, but can produce a very suboptimal or even an infeasible solution (i.e., agent placement resulting in no feasible solution for the task assignment and scheduling problem in one or more workplaces), because it does not consider how agents are used in each workplace.

To address this challenge, we propose a multi-abstraction search approach (MASA). MASA optimizes multi-agent placement by incrementally considering effects of lower-level optimization problems. MASA consists of three phases: 1) it begins with a highly abstract agent placement problem that does not consider the effects of task assignment and scheduling, and uses MILP to compute an initial solution to this problem; 2) it then improves upon this solution using a hill climbing algorithm, which considers the effects of task assignment, in a less abstract problem; and 3) MASA fine-tunes the agent placement solution by considering the effects of both task assignment and scheduling at workplaces.

All three phases of MASA are critical. The first phase generates a suitable initial solution for the hill climbing algorithm. Because solving a task assignment problem is much faster than solving a combined task assignment and scheduling problem, the introduction of the second phase significantly improves the search speed. Moreover, by considering much fewer variables and constraints, search in an abstracted solution space at the second phase helps avoid inferior local optima. The final phase is indispensable because, by co-optimizing agent placement with both task assignment and scheduling, it not only improves the agent placement solution but also helps avoid an infeasible solution generated from previous two phases. In addition, this phase generates a complete solution for agent placement, task assignment, and task scheduling.

The following subsections will discuss in greater detail these three phases of MASA.

Phase 1: finding initial agent placement

This phase aims to quickly find a suitable initial solution of agent placement, building a good start for the hill climbing algorithm. To achieve this goal, we construct an approximate agent placement problem, which does not consider how tasks will be assigned and scheduled to agents. This approximate problem aims to balance agent allocation based on task loads at workplaces and the capabilities of the agent. Specifically, its objective is to minimize the maximum average task load of agents at all workplaces:

$$min_R \quad max_{w \in W, x \in \tau} \sum_{t \in T_{w,x}} D_{wxt}/R_{w,x} \tag{12}$$

where $R_{w,x}$ is a decision variable representing the number of agents of type x assigned to workplace w, $T_{w,x}$ is a set of tasks that can be best performed by an agent of type x at workplace w and D_{wxt} is the expected time to complete task t by an agent of type x. Note that this objective function does not take task assignment or scheduling into consideration.

Note that the objective function (12) is not linear, because the decision variables are denominators. To formulate this problem as a MILP, we reverse the objective and convert it to a maximization problem, maximizing the minimum of the inverse of the average task load of agents at all workplaces. We then introduce a proxy variable, v, to represent this minimum inverse value and to linearize the new objective function. The mathematical MILP formulation of this converted problem is presented below:

$$max_R \quad v \tag{13}$$

subject to

$$v \leq R_{w,x} / \sum_{t \in T_{w,x}} D_{wxt}, \quad \forall w \in W, x \in \tau \tag{14}$$

$$\sum_{w \in W} R_{w,x} \leq n_x, \quad \forall x \in \tau \tag{15}$$

where Equation 15 ensures that agent placement satisfies resource constraints. This MILP can be solved using existing optimizers [Gurobi Optimization, 2015], which return the agent placement matrix.

Phase 2: improving agent placement with task assignment

At the second phase, MASA aims to improve the initial agent placement solution, computed in Phase 1, by taking into consideration how task assignment at workplaces affects agent placement. We develop a hill climbing algorithm for improving the agent placement solution by minimizing the maximum makespan among workplaces. Note that the optimal makespan of a workplace with given agents depends on both task assignment and scheduling. As MASA does not consider task scheduling at this phase, we construct an approximate task assignment problem, detached from task scheduling, to estimate the optimal makespan of a workplace with assigned agents. This approximate problem does not consider temporal and spatial constraints, so it can be solved much faster and enables the rapid hill climbing search.

Algorithm 1 Hill Climbing Algorithm

```
1: procedure HILLCLIMBING(P_{1,...,|W|}, R_{1,...,|W|})
2:     loop
3:         for all w ∈ W do
4:             V[w] = eval(P_w, R_w)
5:         end for
6:         b ← argmax_{w∈W} V[w]
7:         makespan ← V[b]
8:         OW ← increasingly sort W\{b} by values V
9:         for i ← 1, |OW| do
10:            w ← OW[i]
11:            L ← neighbors(⟨R_w, R_b⟩)
12:            for all ⟨R'_w, R'_b⟩ ∈ L do
13:                v_max ← max(eval(P_w, R'_w), eval(P_b, R'_b))
14:                if makespan > v_max then
15:                    makespan ← v_max
16:                    ⟨R_w, R_b⟩ ← ⟨R'_w, R'_b⟩
17:                end if
18:            end for
19:            if makespan < V[b] then
20:                break
21:            end if
22:        end for
23:        if makespan == V[b] then return R_{1,...,|W|}
24:        end if
25:    end loop
26: end procedure
```

The hill climbing algorithm minimizes the maximum makespan of workplaces by iteratively transferring agents from other workplaces to the workplace with the maximum makespan. Algorithm 1 depicts the hill climbing pseudocode. Its input is a set of workplaces, $P_{1,\ldots,|W|}$, and the initial agent placement solution, $R_{1,\ldots,|W|}$. Lines 3-5 computes the optimal makespan of each workplace with assigned agents. In our implementation, as the computation of function $eval$ is not trivial, we cache the computed value for each pair (P_w, R_w) to be retrieved in the future. Line 6 identifies the bottleneck workplace b with the maximum value, which is the makespan. Line 8 sorts workplaces in increasing order of their makespans, such that a workplace with a lower makespan has a higher priority to transfer its agents to the bottleneck workplace. The *for* loop at Line 9 is to iterate through the order list and find a workplace that can transfer its agents in order to improve the makespan. Line 11 generates a set of possible agent placement solutions by transferring agents from w to b. To limit the neighboring solution size, we restrict the number of agents to be transferred at each iteration to two or fewer. Lines 12-18 determine the best agent placement. Line 19 ensures that we only need to find one workplace at each iteration for transferring its agents. Line 23 determines whether any other solution improves the makespan; if not, it returns the current solution.

At this phase, the evaluation function $eval(P_w, R_w)$ in the hill climbing algorithm requires solving the task assignment problem at workplace w with agents R_w. We construct approximate versions of the MILP for task assignment and scheduling presented in the Problem Formulation section. The objective of these approximate task assignment problems is to minimize the maximum total expected time that all agents would require to complete their tasks, assuming those tasks had no temporal and spatial constraints. We introduce a proxy variable, v_w, as this approximate objective and formulate the problem at each workplace w as a MILP:

$$min_A \quad v_w \tag{16}$$

subject to

$$v_w \geq \sum_{j \in T_w} D_{a,j} A_{a,j}, \forall a \in R_w \tag{17}$$

$$\sum_{a \in R_w} A_{a,j} = 1, \forall j \in T_w \tag{18}$$

where $A_{a,j} \in \{0, 1\}$ is a binary decision variable for the assignment of agent a to task j, $D_{a,j}$ is the expected time of task j performed by agent a, and T_w and R_w are a set of tasks and agents available at workplace w, respectively. Equation 17 defines the objective v_w and Equation 18 ensures each task is assigned to only one agent. The optimal value v^*_w can be computed using an existing MILP solver [Gurobi Optimization, 2015]. The hill climbing algorithm uses these optimal values to evaluate its target function, that is, $eval(P_w, R_w) = v^*_w$.

Phase 3: fine-tuning agent placement with task assignment and scheduling

The final phase of MASA operates similarly to the second phase, by applying the hill climbing algorithm to improve the agent placement solution. The difference is that, at this phase, the hill climbing algorithm is applied to optimize the original multi-agent placement objective function (1), instead of an approximate one. In other words, this phase fine-tunes the multi-agent placement solution by taking both task assignment and scheduling at workplaces into account. As the numbers of tasks and agents at each workplace in practice are often relatively small, we can use existing approaches [Hooker, 2009; Castro and Petrovic, 2012; Korsah *et al.*, 2012; Gombolay *et al.*, 2013] for solving multi-agent task assignment and scheduling for each workplace. At this phase, the hill climbing algorithm uses the computed objective value $f^*_w(R_w, P_w)$ of the task assignment and scheduling problem for workplace w with agents R_w to evaluate the candidate agent placement solution (R_w, P_w), i.e., $eval(R_w, P_w) = f^*_w(R_w, P_w)$. When MASA converges, it returns an agent placement solution as well as task assignments to agents at each workplace and their task schedules.

In our implementation, we use the Tercio algorithm [Gombolay *et al.*, 2013] to solve task assignment and scheduling problems and compute nearly optimal makespans for workplaces, as the MILP for task assignment and scheduling is often computationally intractable for relatively large problems (e.g., problems involving more than five agents and 50 tasks). Tercio receives a task assignment and scheduling problem as input and returns a flexible temporal plan that contains task assignment and schedules of agents, if one can be found. The input problem is decomposed into a task allocation and a task sequencing problem. Tercio combines MILP (for solving the task allocation problem) with a fast, satisficing, incomplete multi-agent task sequencer (inspired by real-time processor scheduling techniques). It works by iterating through agent allocations until a schedule can be found that satisfies the maximum allowable makespan for the problem.

Comparison	#Better makespan	Makespan reduced	#Same makespan	#Failures
MASA vs. A-MILP	99	$15.3 \pm 8.2\%$	1	0 : 4
MASA vs. HC-R	97	$25.6 \pm 17.3\%$	3	0 : 12
MASA vs. HC-S	52	$6.9 \pm 5.1\%$	41	0 : 0
HC-S vs. HC-R	91	$23.1 \pm 17.7\%$	2	0 : 12
HC-S vs. A-MILP	94	$13.1 \pm 7\%$	6	0: 4

Table 1: Performance comparison across 100 problems

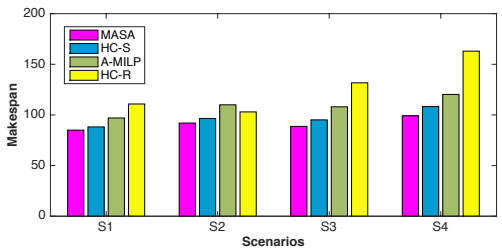

Figure 2: Solution quality for different scenarios

Experiments

In this section, we empirically evaluate the multi-abstraction search approach using the illustrative manufacturing domain, where the multi-agent task assignment and scheduling problem has temporal and spatial-proximity constraints.

Experimental Setup

We generated multi-agent task assignment and scheduling problems that simulate multi-agent construction of a cellular assembly line in a manufacturing domain, such as airplane production. There were two possible types of agents, and the number of workplaces on the production line and the number of agents of either type varied across the problems. We used a task generator similar to that used in [Gombolay *et al.*, 2013] for each workplace, but varied the number of tasks and the proportion of tasks with temporal constraints at different workplaces. At each workplace, task times were generated from a uniform distribution in the interval $[1, 10]$. The proportion of tasks, which were related via a nonzero wait duration (lowerbound constraint) drawn from the interval $[1, 10]$, varied at different workplaces, uniformly drawn from $\{0, \frac{1}{4}, \frac{1}{2}\}$. Approximately $\frac{1}{8}$ of the tasks were related via an upperbound temporal deadline generated randomly to another task. The upperbound of each intra-task and task deadline constraint was drawn from a normal distribution with a mean set to the tightest possible bound. For values lower than the mean, we simply drew a new deadline value. The physical locations of a task were drawn along a single dimension from a uniform distribution $[1, m]$, where m is the total number of tasks at a workplace. While an agent performed a task at resource location (x), we required that no other agent could work on a task at location (u) if $x - 1 \leq u \leq x + 1$.

We ran experiments on 10 scenarios, each with 10 different problems, for a total of 100 problems. Here are four example scenarios:
S1: Three workplaces, 140 tasks and 24 agents
S2: Four workplaces, 200 tasks and 32 agents
S3: Five workplaces, 250 tasks and 40 agents
S4: Six workplaces, 360 tasks and 48 agents

Results and Discussion

To our best knowledge, MASA is the first work to co-optimize multi-agent placement with task assignment and scheduling. We compared MASA with an approximate MILP approach (A-MILP) and a hill climbing algorithm with a random initial solution (HC-R). All approaches used the Tercio algorithm [Gombolay *et al.*, 2013] to solve lower-level task assignment and scheduling problems. The A-MILP method is essentially a traditional top-down, decomposable approach that incorporates as its basis the same MILP for approximate agent placement as that used in the first phase of MASA. It computes the five best solutions through the application of solution exclusion constraints [Tsai *et al.*, 2008] and selects the optimal one. The HC-R approach is a standard single-phase hill climbing algorithm, which begins with a random initial agent placement solution and improves upon it using the same hill climbing algorithm as that used in the third phase of MASA. In order to evaluate the second phase of MASA, we also designed a hill climbing algorithm with a smart initial solution (HC-S), which is similar to MASA but without the second phase.

Table depicts the results of comparisons between two different approaches across 100 problems. Four measures were used to compare the methods: the number of problems for which the first approach generated better solutions than the second one, the average makespan decrease for these problems, the number of problems with the same makespan and the number of failures.

As expected, MASA significantly outperformed the other three approaches with regard to solution quality, identifying feasible solutions for all 100 problems. MASA yielded lower makespan than A-MILP in 99% of the problems, with a mean makespan reduction of 15.3%, and an identical makespan in the remaining 1% of problems. In addition, A-MILP failed to identify feasible agent placement solutions for four problems. The A-MILP approach separates the agent placement problem from task assignment and scheduling by optimizing an approximate objective, the optimal solutions for which do not necessarily yield good or even feasible agent placement. This comparison implies that it is important to consider the effects of task assignment and scheduling when solving the agent placement problem.

MASA performed better than HC-R for 97% of the problems, yielding 25.6% lower makespan on average. HC-R failed to find feasible solutions for 12 % of the problems. As HC-R directly searches in the original problem space beginning with a random solution, this comparison implies that agent placement problems often have inferior local optima and conducting searches in abstract spaces using MASA avoids these inferior solutions, improves the likelihood of finding a feasible solution and converges to a better solution.

The comparison between MASA and HC-S is intended to evaluate the contribution of MASA's second search phase. From Table , we can observe that MASA outperformed HC-S in 52% of the problems, yielding 6.9% lower makespan

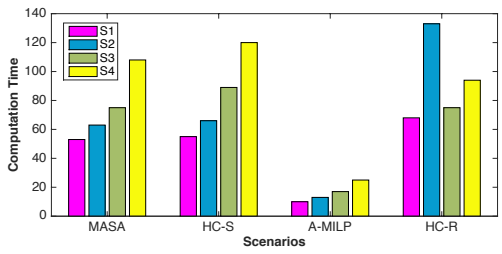

Figure 3: Computation speed for different scenarios

on average (which is significant for such tightly-constrained optimization problems). Although the two methods generated solutions with the same makespan in 41% of problems, MASA converged faster than HC-S, with an 11.7% shorter computation time on average. This improvement in solution quality is due to the abstract search method used at the second phase, while the speed improvement is attributable to cheaper computation of evaluating the objective function.

We compared HC-S with A-MILP and HC-R in order to evaluate the other two phases of MASA. While HC-R begins with a random solution, HC-S uses MASA's first phase to initialize its solution. The superior performance of HC-S over HC-R implies the importance of MASA's first search phase, while the superior performance of HC-S over A-MILP indicates the significance of the third phase.

Next, we discuss how the different approaches perform as the number of workplaces increases. Figure 2 depicts the average makespan yielded by four approaches over 10 problems in scenarios S1, S2, S3, and S4, which incorporated three, four, five and six workplaces, respectively. One key observation is that the improvement of MASA over other approaches widens as the number of workplaces increases and the problems become more complex. For example, MASA yielded 14%, 18%, 20% and 22% lower makespan than A-MILP and 3.8%, 4.9%, 7.2% and 9.2% lower makespan than HC-S for scenario S1, S2, S3 and S4, respectively.

Figure 3 indicates how computation time changed for each of the four approaches as the number of workplaces increased. The computation time for all approaches seemed to increase polynomially with the number of workplaces, with the exception of HC-R. HC-R had unexpectedly large average computation time in scenario S2, because there are several cases where HC-R started with "bad" random solutions and took an unusual number of iterations to converge. MASA converged faster than HC-S for all scenarios, and faster than HC-R for the first three scenarios. In scenario S4, HC-R often converged quickly to very poor solutions, with more than 40% higher makespan on average compared with MASA. Although A-MILP ran faster than MASA, due to its top-down decomposition and much less frequent calling for the Tercio algorithm, it is clear to choose MASA over A-MILP in real-world applications, because solution quality is often the most important factor and the computation times of these two approaches are of the same order of magnitude.

Related Work

There is a wealth of prior work in multi-agent task assignment and scheduling with temporal and spatial constraints. Korsah *et al.* provide a comprehensive taxonomy [Korsah *et al.*, 2013] for the multi-agent task allocation and scheduling problem. One line of research is the development of hybrid approaches that combine MILP with other techniques, such as constraint programming [Hooker, 2004; 2009; Jain and Grossmann, 2001; Korsah *et al.*, 2012] or integrating heuristic schedulers [Tan, 2000; Castro and Petrovic, 2012; Gombolay *et al.*, 2013; Lipovetzky *et al.*, 2014]. Although these approaches do not scale well for large problems, they often yield nearly optimal solutions to relatively small problems and can work with our multi-abstraction method for solving subproblems of multi-agent task assignment and scheduling.

Another line of research focuses on distributed approaches. For example, Cicirello and Smith propose wasp-like multi-agent systems for distributed factory coordination [Cicirello and Smith, 2004]. Auction algorithms have also been proposed for allocating tasks with time constraints [Ponda *et al.*, 2010; Nunes and Gini, 2015]. Amador *et al.* apply a market clearing approach to task allocation [Amador *et al.*, 2014]. Ramchurn *et al.* develop a coalition formation technique for task allocation to maximize the number of tasks completed before hard deadlines [Ramchurn *et al.*, 2010]. These works improve computational performance by decomposing plan constraints among agents. However, these methods often break down when agents' schedules become tightly inter-coupled, as they do when multiple robots operate in close physical proximity to one another.

Auction algorithms have been used for resource allocation [Huang *et al.*, 2008]. In fact, our hill climbing algorithm is a centralized implementation of an auction algorithm if Lines 19-20 (used to reduce the number of times to evaluate the objective function) are removed. Metaheuristic algorithms, such as simulated annealing [Aerts and Heuvelink, 2002], hill climbing with random restarts [Poole and Mackworth, 2010], have also been used for resource allocation. We did not use these algorithms in our method, however, because they require solving a much larger number of task assignment and scheduling problems.

Conclusion

This paper formulates large-scale multi-agent coordination as a multi-level optimization problem requiring multi-agent placement and task assignment and scheduling. We develop a multi-abstraction search technique for solving this optimization problem. This approach initiates the search with an abstract problem, which often prevents the convergence to inferior local optima and improves convergence speed. Empirical evaluation of the contribution of each abstraction verifies the performance advantages of this multi-abstraction approach with regard to the quality, speed, and likelihood of convergence. Significant outperformance of this approach compared with a top-down decomposition method also implies the importance of incorporating feedback from lower-level optimization for high-level optimization.

References

[Aerts and Heuvelink, 2002] Jeroen CJH Aerts and Gerard BM Heuvelink. Using simulated annealing for resource allocation. *International Journal of Geographical Information Science*, 16(6):571–587, 2002.

[Amador *et al.*, 2014] Sofia Amador, Steven Okamoto, and Roie Zivan. Dynamic multi-agent task allocation with spatial and temporal constraints. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1495–1496. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[Bertsimas and Weismantel, 2005] Dimitris Bertsimas and Robert Weismantel. *Optimization over integers*, volume 13. Dynamic Ideas Belmont, 2005.

[Brucker, 2001] Peter Brucker. *Scheduling Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition, 2001.

[Castro and Petrovic, 2012] Elkin Castro and Sanja Petrovic. Combined mathematical programming and heuristics for a radiotherapy pre-treatment scheduling problem. *Journal of Scheduling*, 15(3):333–346, 2012.

[Cicirello and Smith, 2004] Vincent A Cicirello and Stephen F Smith. Wasp-like agents for distributed factory coordination. *Autonomous Agents and Multi-agent systems*, 8(3):237–266, 2004.

[Dechter *et al.*, 1991] Rina Dechter, Italy Meiri, and Judea Pearl. Temporal constraint networks. *AI*, 49(1):61–91, 1991.

[Gombolay *et al.*, 2013] Matthew C. Gombolay, Ronald Wilcox, and Julie A. Shah. Fast scheduling of multi-robot teams with temporospatial constraints. In *Robotics: Science and Systems*, 2013.

[Gurobi Optimization, 2015] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2015.

[Hooker, 2004] John N. Hooker. A hybrid method for planning and scheduling. In *Proc. Carnegie Mellon University Research Showcase*, 2004.

[Hooker, 2009] John N. Hooker. An improved hybrid MILP/CP algorithm framework for the job-shop scheduling. In *Proc. IEEE International Conference on Automation and Logistics*, 2009.

[Huang *et al.*, 2008] Jianwei Huang, Zhu Han, Mung Chiang, and H Vincent Poor. Auction-based resource allocation for cooperative communications. *Selected Areas in Communications, IEEE Journal on*, 26(7):1226–1237, 2008.

[Jain and Grossmann, 2001] Vipul Jain and Ignacio E. Grossmann. Algorithms for hybrid MILP/CP models for a class of optimization problems. *Journal on Computing*, 13(4):258–276, 2001.

[Korsah *et al.*, 2012] G Ayorkor Korsah, Balajee Kannan, Brett Browning, Anthony Stentz, and M Bernardine Dias. xbots: An approach to generating and executing optimal multi-robot plans with cross-schedule dependencies. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 115–122. IEEE, 2012.

[Korsah *et al.*, 2013] G Ayorkor Korsah, Anthony Stentz, and M Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, 2013.

[Lipovetzky *et al.*, 2014] Nir Lipovetzky, Christina N Burt, Adrian R Pearce, and Peter J Stuckey. Planning for mining operations with time and resource constraints. In *Proceedings of 24th Int. Conf. on Aut. Planning and Scheduling, ICAPS*, 2014.

[Nunes and Gini, 2015] Ernesto Nunes and Maria Gini. Multi-robot auctions for allocation of tasks with temporal constraints. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[Ponda *et al.*, 2010] Sameera Ponda, Josh Redding, Han-Lim Choi, Jonathan P How, Matt Vavrina, and John Vian. Decentralized planning for complex missions with dynamic communication constraints. In *American Control Conference (ACC), 2010*, pages 3998–4003. IEEE, 2010.

[Poole and Mackworth, 2010] David L Poole and Alan K Mackworth. *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.

[Ramchurn *et al.*, 2010] Sarvapali D Ramchurn, Maria Polukarov, Alessandro Farinelli, Cuong Truong, and Nicholas R Jennings. Coalition formation with spatial and temporal constraints. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 3-Volume 3*, pages 1181–1188. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[Tan, 2000] Wei Tan. Integration of process planning and scheduling - a review. *Journal of Intelligent Manufacturing*, 11:51–63, 2000.

[Tsai *et al.*, 2008] Jung-Fa Tsai, Ming-Hua Lin, and Yi-Chung Hu. Finding multiple solutions to general integer linear programs. *European Journal of Operational Research*, 184(2):802–809, 2008.