Auton Agent Multi-Agent Syst (2017) 31:1424–1466
DOI 10.1007/s10458-017-9367-7

Hybrid mission planning with coalition formation

Anton Dukeman[1] · Julie A. Adams[1]

Published online: 8 May 2017
© The Author(s) 2017

Abstract The increase in robotic capabilities and the number of such systems being used has resulted in opportunities for robots to work alongside humans in an increasing number of domains. The current robot control paradigm of one or multiple humans controlling a single robot is not scalable to domains that require large numbers of robots and is infeasible in communications constrained environments. Robots must autonomously plan how to accomplish missions composed of many tasks in complex and dynamic domains; however, mission planning with a large number of robots for such complex missions and domains is intractable. Coalition formation can manage planning problem complexity by allocating the best possible team of robots for each task. A limitation is that simply allocating the best possible team does not guarantee an executable plan can be formulated. However, coupling coalition formation with planning creates novel, domain-independent tools resulting in the best possible teams executing the best possible plans for robots acting in complex domains.

Keywords Multi-agent planning · Complex mission planning · Temporal planning · Continuous planning

1 Introduction

The domains robots can operate in is rapidly expanding as robotic capabilities increase. Some robotic domains, such as mass casualty response, will require close coupling between the human and robot responders in order to successfully complete the mission. A taxonomy for categorizing multi-agent system problems includes the types of robots (single-task robots vs. multi-task robots), the number of robots per task (single-robot tasks vs. multi-robot tasks),

✉ Anton Dukeman
anton.dukeman@vanderbilt.edu

Julie A. Adams
julie.a.adams@vanderbilt.edu

1 Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN 37240, USA

and when information is available (instantaneous allocation vs. time-extended allocation) [20]. This manuscript addresses centralized, domain-independent planning for multi-task robot, multi-robot task, instantaneous allocation missions.

The planning problem uses an initial state and a set of actions and constraints to derive a plan to achieve a goal state. The planning problem complexity is partially a function of the number of robots and tasks and partially a function of the domain model complexity [16]. Planning for expressive real-world models with durative actions, joint actions, concurrently executing actions, continuous variables, and continuous effects is much harder than planning domains with instantaneous actions and boolean variables.

Consider a mass casualty response scenario after a tornado, such as the EF-4 tornado in Tuscaloosa, AL on April 27, 2011. The immediate response involved hundreds of responders from local government agencies and required coordination to complete several tasks, including clearing impassable roads, securing prohibited items, triaging the wounded, and locating victims in the disaster area. Moving about the environment, clearing debris from roads, and all other actions require time and are not instantaneous actions. Agents must be able to coordinate actions and work concurrently. Real-world domains include continuous variables, such as fuel level, that must be considered. The heterogeneous agents, complex tasks, and complex environment complicate this difficult planning problem.

One method to address planning complexity is factored planning, which splits the goal into lower complexity subgoals. Multi-agent planning approaches using factored planning focus on how the entire coalition [4,14] or an individual agent [46] can solve subproblems. Factored planning as part of single-agent planning does not address multiple agents [7,23]. Solutions for path-planning [47] and target tracking [24] exist, but are domain-dependent. The tools presented in this manuscript address domain-independent planning problems with multiple, heterogeneous agents and multiple, complicated tasks in complex domains with durative actions, concurrently executing actions, continuous variables, and continuous effects by factoring the problem by both tasks and coalitions of agents.

Coalition formation can manage problem complexity by allocating a team of agents to each task. Coalition formation uses the capabilities of the agents and the capability requirements of the tasks to form teams of agents that can accomplish a set of assigned tasks, while optimizing an objective function (e.g., utility, cost, or number of tasks completed) [40]. The factoring produced by coalition formation generates several smaller problems from the original problem; however, coalition formation cannot guarantee allocated coalitions will be able to execute the task to which they are assigned to complete.

Current coalition formation research does not address the nonexecutable coalition problem (a coalition which cannot complete its assigned task) and current planning research does not perform task allocation on the scale of coalition formation. Factored planning is a popular approach to decomposing the problem into manageable subproblems, but existing factored planning algorithms do not consider agents. Multi-agent planning performs goal allocation, but typically does not consider how multiple agents working together simultaneously on a single task affects plan quality. Three novel tools incorporating both coalition formation and planning are presented: coalition formation then planning, relaxed plan coalition formation, and task fusion. The coalition formation then planning approach is used as the basis for the other tools that utilizes the output from the coalition formation problem as the inputs to the planning problem. Tasks are planned separately by the allocated coalitions and the results are merged into a single solution for the original problem. The coalition formation then planning tool derives satisficing solutions quickly, but raises two problems: nonexecutable coalitions and suboptimal solutions. If coalition formation allocates a coalition to a task that the coalition cannot complete, then a new coalition must be allocated. Relaxed plan coalition formation augments coalition formation then planning by performing iterative planning, relaxed planning, and coalition formation until a valid executable plan is identified. The second problem with coalition formation then planning is suboptimal solutions. Limiting the agents available for planning limits the problem complexity, but reduces the size of the problem solution set; thus, leading to suboptimal solutions. Task fusion balances solution quality with problem complexity by planning for tasks and coalitions together for which higher solution quality outweighs increased problem complexity.

Section 2 provides an overview of related research. Section 3 presents a formal definition of the problem. Three experimental domains are given in Sect. 4. Section 5 presents the experimental design used to evaluate the presented planning tools. Section 6 presents the tools and the results for each tool solving problems in each experimental domain followed by a discussion of how the results motivate the next tool. Finally, a conclusion and future work is presented in Sect. 7.

2 Related work

2.1 Coalition formation

Coalition formation is a subclass of the task allocation problem without constraints on the number of agents (robot or human) allocated to each task nor the number of coalitions to which an agent is a member. The coalition formation problem is *NP*-complete [37], is difficult to approximate [40], and represents a multi-task, multi-robot problem that incorporates algorithms for both instantaneous allocation [42,50] and time extended allocation [25,34]. The goal is to form teams of agents that are together more capable than the team's individual agents and can accomplish a set of assigned tasks, while optimizing an objective function (e.g., utility, cost, tasks completed). The general coalition formation problem assumes a grand coalition of n agents, $\Phi = \{\phi_1, \ldots, \phi_n\}$, and a set of m tasks, $V = \{v_1, \ldots, v_m\}$. A solution is a map of each task, $v \in V$, to a coalition assigned to the task, $\Phi_v \subseteq \Phi$ [37].

Agents and tasks are modeled as their capabilities offered or capabilities required, respectively. Two different capability models are used, the resource model and the service model. The resource model treats each agent as a set of available resources (e.g., chemical sensor, camera, laser) and each task as a set of required resources. Let Res be a vector of possible resource types, where Res_i is the ith resource type. Each agent, ϕ, is modeled as a resources available vector, Res^{ϕ}, and a coalition, $\Phi_i \subseteq \Phi$, is modeled as a vector equal to the sum of the available resource vectors of the constituent agents, $Res^{\Phi_i} = \sum_{\phi \in \Phi_i} Res^{\phi}$. A task, v, is similarly defined as a resources required vector, Res^{v}. All elements of resources available vectors and resources required vectors must be non-negative, and at least one element in each vector must be non-zero. A coalition, Φ_j, is a candidate coalition for a task, v, if and only if it has available at least as many of each resource type as the task requires, $\forall i,\ Res_i^{\Phi_j} \geq Res_i^{v}$. Only a candidate coalition for v can be allocated to v.

The service model associates a set of functions that each agent can perform with the particular agent (e.g., box-pushing, mapping, sentry-duty). Let Ser be a vector of possible service types, where Ser_i is the ith service type. An agent, ϕ, has a services available vector indicating whether or not each service is offered by the agent, $Ser_i^{\phi} \in \{0, 1\}$, where Ser_i^{ϕ} is 1 if ϕ offers service i and 0 if not. A coalition, Φ, has a services available vector equal to the sum of the services available vectors of its constituent agents, $Ser^{\Phi} = \sum_{\phi \in \Phi} Ser^{\phi}$. A task, v, is modeled as a services required vector, Ser^{v}, where $Ser_i^{v} \in \mathbb{N}$ is a non-negative integer representing the number of services of type Ser_i required to satisfy v and $\exists j, Ser_j^v > 0$. A coalition, Φ, is a candidate coalition for a task, v, if and only if has available at least as many services as the task requires, $\forall i, Ser_i^{\Phi} \geq Ser_i^v$.

There are many heuristic-based coalition formation algorithms, each with its own strengths and weaknesses. Greedy algorithms can derive solutions quickly, but make no guarantees on the solution quality [40,42,44,51]. Approximation algorithms provide solution quality guarantees, but suffer from poor worst-case run-time complexity, which can render them inappropriate for real-time applications [27,33]. Market-based techniques offer fault-tolerance for a distributed system, but have high communication processing requirements [10,41,43,50]. Biologically inspired ant colony optimization algorithms have been applied to several *NP*-complete problems, including coalition formation [36,38]. Different coalition formation algorithms provide different solutions with differing performance. For example, selecting a market-based algorithm with high communications requirements for use in a communications constrained environment results in poor performance. The intelligent Coalition Formation for Humans and Robots system was developed to autonomously reason over the specified mission constraints in order to select a subset of coalition formation algorithms to apply to a particular allocation problem [39].

2.2 Planning

Classical planning results in a satisficing plan that contains a sequence of actions that achieves a goal state [18]. While classical planning is for single-task robots executing single-robot tasks with an instantaneous allocation, variants of classical planning span the Gerkey and Matarić taxonomy. The extensions of classical planning most applicable to this research are temporal planning, continuous planning, and multi-agent planning. Temporal planning admits durative actions to the action set and allows concurrently executed actions in the solution, continuous planning incorporates continuous variables in the state space and continuous effects in the actions, and multi-agent planning models multiple agents executing actions, rather than a single agent, as in classical planning.

Temporal planning incorporates durative actions and concurrent action execution. The model for durative actions expands the classical action model to include a duration and temporal specifications for action conditions and effects. Action duration specifies the length of time required to execute the action. The temporal specifications indicate when conditions must be satisfied (at the beginning, at the end, or over the entire action duration) and when the effects are applied (at the beginning or at the end of action execution). A solution to the temporal planning problem is a satisficing plan that combines a set of actions with execution constraints to achieve a goal state. Temporal planning solutions can be classified as single-task agents executing single-agent tasks in an instantaneous allocation. State-space based search is a popular method for planning, such as Yet Another Heuristic Search Planner (YAHSP) [49], but other methods such as SAT-based planners (ITSAT [35]) also exist.

Approaches to managing problem complexity include subgoal partitioning and state-based decomposition. Subgoal Plan solved large problems by creating a subgoal partitioning through goal constraint analysis [7]. The subproblems were solved by Metric Fast Forward [21] and were significantly easier to solve than the original problem. The time to solve a problem exponentially decreased as the subproblems' size was linearly reduced. Divide-and-Evolve, similar to Subgoal Plan, used a preprocessing step to decrease problem complexity before an encapsulated satisficing planner was used to solve the problem [3]. Divide-and-Evolve used a state-based decomposition strategy to find a sequence of intermediate states that collectively solve the problem. Divide-and-Evolve with YAHSP [49] as the encapsulated planner solved significantly more problems than YAHSP alone.

Continuous planning incorporates continuous variables in the state space and expands the action model to include continuous effects. Classical planning models require continuous variables to be discretized; however, real-world models are more accurate when state variables, such as fuel level and temperature, can be modeled as continuous variables. Continuous effects must be combined with temporal planning and durative actions can have effects applied over the entire action duration, known as a continuous effect. For example, an accurate real-world model of aircraft flight must include a continuously decreasing fuel level. If continuous effects are not admitted, then the change in fuel level over the entire action duration must be applied instantaneously. A solution to the continuous planning problem is a satisficing plan that combines a set of actions with execution constraints to achieve a goal state. Continuous planning solutions can be classified as single-task agents executing single-agent tasks in an instantaneous allocation.

Some planners, such as Temporal Fast Downward (TFD) [17], support continuous variables, but not continuous change. Zeno was the first planner to allow continuous change in planning problems [31]; however, it was unable to handle concurrent continuous effects, such as are required for an accurate model of in-flight refueling. COLIN (COntinuous LINear) extended state space search techniques to manage continuous effects [8]. Other continuous planning algorithms include IxTeT [26], Sapa [12], and dReal[5]. Accommodating domains with non-linear continuous change allows real-world domains to be more accurately modeled, but is only supported by dReal.

Multi-agent planning explicitly considers multiple agents executing actions by substituting a set of agents for a set of actions in the classical planning definition, where each agent is modeled as a set of actions that the agent can execute. The solution is a plan specifying a set of actions with execution constraints and an associated agent responsible for executing each action. Multi-agent planning solutions exist for single-task and multi-task agents, single-agent and multi-agent tasks, and instantaneous allocation.

Factored planning approaches are natural multi-agent planning solutions. One of the first such algorithms used individual agent planning to generate a heuristic for use in global planning for the original problem [14,15]. Another factored planning approach performed distributed planning followed by agent voting [46]. Agents modified a base plan and distributed it to the other agents. A new base plan was selected from the set of agent plans by voting. If all agents indicated the base plan satisfied their task, then planning ended, otherwise another iteration of planning and voting occurred.

Deriving plans individually requires a plan merge step to integrate the plans to a single global solution. Plan merge allows agents to take advantage of side products, the unused product of other agents' actions, to eliminate redundant actions in plan merge steps [52]. Another approach treats the problem as a plan-space search problem in which incremental changes are made until the plan is valid [9]. Part of the plan merge problem requires satisfying all temporal constraints. Simple temporal networks have been applied by encoding the temporal constraints of the individual plans and finding consistent variable assignments representing a valid plan merge [1].

The domain complexity and the method used for factoring the problem are the differentiating aspects addressed by the presented tools. Existing multi-agent planning solutions focus on instantaneous actions and discrete state spaces or assume that tasks are executable by a single agent [4,11,13,30]. Developing real-world domain models requires durative actions in continuous state spaces for tasks that require multiple agents.

2.3 Integrated task allocation and planning

Chance-constrained task allocation is an example approaches that incorporate task allocation with aspects of planning [32]. Each agent estimated the utility of it completing each task. Allocation utility was a function of the agent allocated to the task, when the agent will be able to execute the task, and a predefined model of problem uncertainty. However, real-world problems do not consist exclusively of single-agent tasks.

Approaches to coalition formation in which tasks have temporal and spatial constraints can address task allocation and scheduling, but do not produce plans for how agents will execute their allocated tasks when they reach the task location [25,34]. Both approaches are for multi-task agents executing multi-agent tasks in a time-extended allocation and assumed all agents capable of reaching the task location were able to contribute to the task, an invalid assumption in some missions (e.g., an agent without a camera cannot assist an imaging task).

Auction style coalition formation algorithms allow agents to perform task planning prior to allocation and typically grant exclusive ownership of tasks, which can be detrimental when agents fail to complete their task and no method for informing the other agents of the failure exists. A method based on bounty hunters and bail bondsmen allows for nonexclusive task execution [53]. Following the bail analogy, agents act as bounty hunters and auctioneers as bail bondsmen. The bail bondsmen increase the value of each task until it has been completed. Agents commit to a task and announce to the other agents that they have committed to the task. Agents can plan how to complete each available task, but can only commit to a single task. Agents receive the task utility only upon completing the task. If an agent fails to complete a task, the system adapts by incentivizing other agents to complete the task through increasing task value. This approach lacks collaboration among the agents, a key feature in real-world problems representative of multi-robot task domains. These auction approaches are for multi-task agents executing single-agent tasks.

The Automatic Synthesis of Multi-robot Task solutions through software Reconfiguration (ASyMTRe) system used connected agent and task schemas to allocate coalitions to tasks [45]. Agents were modeled by perceptual, motor, and communication schema and tasks were modeled as a set of motor schema requirements. ASyMTRe connected robot schemas to develop a joint schema capable of accomplishing assigned tasks. For example, if robot r_i knows its position relative to robot r_j and r_j knows its position in a global reference frame, then r_i can derive its position in the same global reference frame. Connecting the various robotic schemas determined which agents were capable of jointly completing a task, but did not plan how the robots completed the task. A similar system, Remote Object Control Interface, considered robots as nodes offering expanded functionality dependent on the other nodes in the system [6]. Both systems fail to produce executable plans for the assigned tasks. These two systems are both for multi-task agents executing multi-agent tasks.

One domain-dependent integration of task allocation and planning that has been extensively studied is the multi-robot task allocation and path planning problem [2,29,54]. Simultaneously considering the task allocation and the path to the task allows for collision-free trajectories to be developed more efficiently than if the two problems were considered independently. One application incorporates two agents that swap tasks when a collision is detected [47,48]. The new trajectories for the agents are guaranteed not to collide with one another. A search and destroy problem with attack UAVs developed a plan offline to determine an optimal search pattern to locate mobile targets whose locations were unknown a priori; thus, the decision regarding which UAVs will perform the attack must be made online [24]. A distributed probabilistic approach considered the path for each UAV to reach the target, the UAV's attack capability, and the probability of target destruction. These approaches are feasible for the specific domains, but many, diverse multi-agent domains exist and developing a different solution for each domain is impractical.

Task allocation and planning are closely coupled problems, but there is minimal existing literature that addresses the interaction between the two problems. Planning affects task allocation via the developed plans, as the plan constrains the set of agents available by requiring agents to perform actions at specific times. Task allocation affects planning by determining which agents are available when developing a plan. If an agent is not allocated to a task, then the planning algorithm will not use the agent to develop a plan. Tools for coupling domain-independent task allocation and planning will facilitate solving planning problems consisting of multi-task agents executing multi-agent tasks.

3 Formal definition

The presented tools are for planning for multi-task robot, multi-robot task, instantaneous allocation problems [20]. This Hybrid Mission Planning with Coalition Formation problem couples coalition formation with planning to facilitate solving complex problem instances with heterogeneous multi-task robots executing multi-robot tasks.

Definition 1 (*Hybrid Mission Planning with Coalition Formation*) The *hybrid mission planning with coalition formation* problem is defined as a tuple, $\langle S, I, \Phi, V, C\rangle$, where:

- S is the state space,
- I is the initial state,
- $\Phi = \{\phi_1, \phi_2, \ldots, \phi_m\}$ is the grand coalition of agents,
- $A = 2^{\Phi} \rightarrow 2^{Act}$ is the coalition-action set mapping,
- $V = \{v_1, v_2, \ldots, v_n\}$ is the set of tasks, and
- $C = \langle Cap, C_{\Phi}, C_V\rangle$ is the capability vector, coalition capability mapping, and the task capability mapping.

The hybrid state space, S, includes boolean, discrete, and continuous variables. A state, s, is an assignment of each state space variable to a value in its associated domain. The initial state, I, is the environment state at the beginning of the mission.

The grand coalition, Φ, is the set of all available agents. A coalition, $\Phi_i \subseteq \Phi$, is any non-empty set of agents. The coalition-action set mapping, A, maps a possible coalition, 2^{Φ}, to a set of actions the coalition can execute, 2^{Act}, where Act is the set of all possible actions. An action is modeled as a tuple, $\langle \Phi_{exec}, \mathit{eff}, cond, dur\rangle$, where:

- Φ_{exec} is the executor coalition,
- $cond = \langle cond_{\vdash}, cond_{\leftrightarrow}, cond_{\dashv}\rangle$ is the action state constraints that must be satisfied at the beginning, during, and at the end of action execution, respectively, and
- $\mathit{eff} = \langle \mathit{eff}_{\vdash}, \mathit{eff}_{\leftrightarrow}, \mathit{eff}_{\dashv}\rangle$ is the action effects for atomic fact transitions applied to the state at the beginning of, during, and at the end of action execution, respectively,
- dur is a constraint on the length of the time interval required to execute the action.

The executor coalition, Φ_{exec}, for an action, a, is the set of agents that execute a. If Φ_{exec} is a singleton coalition consisting of a single agent, then a is a single-agent action. If Φ_{exec} includes more than one agent, then a is a joint action between multiple agents. A state constraint can be applied to boolean or continuous state variables. Constraints on boolean variables specify the truth value the variable must take, while constraints on continuous variables specify the interval to which the variable's value must belong. Action state constraints can be specified as applying at the beginning, during, or end of action execution,

$cond_{\vdash}$, $cond_{\leftrightarrow}$, and $cond_{\dashv}$, respectively. Action effects at the beginning of action execution, $eff_{\vdash}$, can apply to boolean state variables (as setting the value to true or false) or to continuous state variables (as an instantaneous change in value). Action effects throughout action execution, $eff_{\leftrightarrow}$, must apply to continuous state variables and represent a continuous change in the value of the variable during action execution. Action effects at the end of action execution, $eff_{\dashv}$, can apply to boolean state variables or to continuous state variables. The action duration constraint, dur, is the interval to which action duration must belong. Action duration must be non-negative. Similar actions, such as navigating between waypoints, are considered different if they are executed by different agents. For example, ϕ_i navigating from w_r to w_s is different than ϕ_j navigating from w_r to w_s.

The task set, V, is a set of tasks to be satisfied. Each task, $v \in V$, is modeled as a set of goal state constraints. A task, v, is satisfied in a state, s, if and only if all of v's goal state constraints are satisfied in s.

The capability vector, $Cap = [Cap_1, Cap_2, \ldots]$, is the vector of coalition formation capabilities used in the problem. The coalition capability mapping, C_{Φ}, is a mapping of each agent to a capability available vector. The elements of a capabilities available vector are non-negative values, with at least one non-zero element. Each agent, ϕ, has a capabilities available vector, Cap^{ϕ}. For example, if $|Cap| = 5$ and ϕ has two of Cap_3 and three of Cap_5, then $Cap^{\phi} = [Cap_1^{\phi} = 0, Cap_2^{\phi} = 0, Cap_3^{\phi} = 2, Cap_4^{\phi} = 0, Cap_5^{\phi} = 3]$, where Cap_j^i is the amount of Cap_j that entity i (agent or coalition) has at its disposal. Each coalition, Φ, has a capabilities available vector, Cap^{Φ}, equal to the sum of the capability available vectors of Φ's constituent agents, $Cap^{\Phi} = \sum_{\phi \in \Phi} Cap^{\phi}$. The task capability mapping, C_V, is a mapping of each task to a capability required vector. The elements of a capability required vector are non-negative reals, with at least one non-zero element. For example, if $|Cap| = 5$ and v requires one of Cap_2 and two of Cap_3, then $Cap^v = [Cap_1^v = 0, Cap_2^v = 1, Cap_3^v = 2, Cap_4^v = 0, Cap_5^v = 0]$, where Cap_j^i is the amount of Cap_j required to satisfy i.

A *plan*, π, is a set of action steps. An action step consists of an action, a start time to begin executing the associated action, and the duration of the action. An *executable plan* is a plan for which the action steps are executed validly. An action step is executed validly if the associated action's state constraints are satisfied. Executing the action steps in a executable plan transitions the environment from the initial state, I, to an end state, s_{end}, achieved after all action steps have finished. A solution to the problem is a *satisficing plan*, an executable plan in which s_{end} satisfies the goal state constraints of each task, $v \in V$. A utility function, such as makespan or number of action steps, can be used to compare satisficing plans. An *optimal plan* is a satisficing plan that maximizes the selected utility function. A coalition is an *executable coalition* if a satisficing plan has been derived for the coalition to complete its task. A *nonexecutable coalition* is a coalition for which a satisficing plan has not been derived for the coalition to complete its task.

4 Example domains

The goal is to solve complex real-world domain problems with multiple heterogeneous agents, durative actions, and complex state spaces. Existing planning problems were modified as a first step towards achieving this goal and evaluating the presented tools. Most existing planning domains lack at least one of the aspects representative of the desired domains and to properly evaluate the presented planning tools. A modified Blocks World domain will be used to illustrate the formal problem definition. Two additional planning domains, Rovers

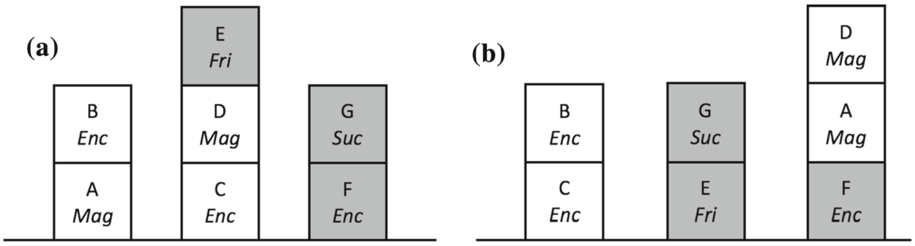


Fig. 1 Example states with the double-weight blocks *shaded* and required end effector for each block in italics. **a** Initial state, **b** goal state constraints

and a modified Zenotravel, are presented and used to experimentally validate the tools. Each domain, and the modifications to each, are presented and implemented in the Planning Domain Definition Language (PDDL) [19].

4.1 Blocks World

The modified Blocks World domain requires that heterogeneous robotic arms manipulate stacks of heterogeneous blocks on a table of finite size. Each arm has a subset of end effectors available to it, while each block requires a specific end effector to be manipulated. A block can be manipulated by an arm if and only if the arm has the block's required end effector. While blocks have the same dimensions, blocks can be either single- or double-weight. Single-weight blocks can be manipulated by a single arm with the required end effector, while double-weight blocks require two arms, each with the required end effector, in order to be manipulated. The block stacks rest on a table with only enough space for a finite number of block stacks. The goal state is a rearrangement of the blocks from the initial state into a specified set of block stacks. The modified domain has been made freely available.[1]

The state space, S, includes both boolean and continuous variables. The boolean variables describe the block stacks, each block's required end effector type, which block each arm is holding, and each arm's available end effectors. The continuous variables describe the height of each arm and block, the number of blocks on the table, and the table capacity. The domain of the continuous variables is non-negative integers, which is not continuous; however, modeling the variables as continuous simplifies the state model by not requiring all possible values to be enumerated and ordered. The initial state, I, is an assignment of a value to each variable in the state space. As a partial example, the middle stack in the example initial state in Fig. 1a is expressed by assigning the value true to the following variables: $(onTable\ C)$, $(onBlock\ D\ C)$, $(onBlock\ E\ D)$, $(requires\ C\ encompass)$, $(requires\ D\ magnetic)$, and $(requires\ E\ friction)$.

The grand coalition, Φ, is the set of arms executing actions. The actions are the up and down arm movement and block manipulation. The duration of each action is a linear function of the number of arms executing the action, i.e., a single arm picking up a single-weight block is a shorter action than two arms picking up a double-weight block due to fewer arms executing the action. An example PDDL implementation of arm a picking up a single-weight block b_1 off of block b_2 is presented in Fig. 2. The action has a duration of 1. Executing the action requires that a be empty, that b_1 be clear, and that b_1 be on b_2 at the start of action execution, while over the entire action execution a must be at the same height as b_1, that b_1 require the specified end effector, and that a have the specified end effector. The action has three start

[1] https://gist.github.com/aldukeman/879b32f602282f729770c5ddac25fbaa.

```
(:durative-action pick-single-block-on-block
  :parameters (?a - arm ?b_1 - single_block ?b_2 - block ?e - effector)
  :duration (= ?duration 1)
  :condition
    (and
      (at start (empty ?a))
      (at start (clear ?b_1))
      (at start (on_block ?b_1 ?b_2))
      (over all (= (arm_height ?a) (block_height ?b_1)))
      (over all (requires ?b_1 ?e))
      (over all (has_effector ?a ?e))
    )
  :effect
    (and
      (at start (not (empty ?a)))
      (at start (not (clear ?b_1)))
      (at start (not (on_block ?b_1 ?b_2)))
      (at end (clear ?b_2))
      (at end (holding_single ?a ?b_1))
    )
)
```

Fig. 2 PDDL implementation for an arm to pick up a block off another block

of action effects, a is no longer empty, b_1 is no longer clear, and b_1 is no longer on b_2. The two end of action effects are that b_2 is clear, and that a is holding b_1. The combination of effects at the beginning and end of action execution ensures logical consistency throughout action execution. For example, the combination of effects ensures that a third block cannot be placed on b_2 while b_2 is being removed from on top of b_1.

Each stack of blocks in the goal state corresponds to a task. The example goal state in Fig. 1b is divided into three tasks: v_C, v_E, and v_F. v_C is the stack with C on the bottom and the goal state constraints for v_C are satisfied when C is on the table and B is on C, i.e., when $(onBlock BC)$ and $(onTableC)$ are both true.

The capability vector for the Blocks World domain corresponds to the end effector types: $[suction, friction, magnetic, encompass]$. The capabilities offered vector for each arm is a function of the end effectors available to the arm. For example, an arm with a friction end effector and an encompass end effector has the capabilities available vector [0, 1, 0, 1]. Double-weight blocks require twice the capabilities of single-weight blocks, because manipulating double-weight blocks requires two robotic arms. The capabilities for each stack are a function of two sets of blocks, the blocks in the goal stack and the blocks that must be manipulated to access the blocks in the goal stack. For example, the capabilities required vector for v_E is a function of E and G, because they are the blocks in the goal stack and there are no other blocks above E and G in the initial state. E requires two suction capabilities and G requires the two friction capabilities; therefore, the capabilities required vector for v_E is [2, 2, 0, 0]. The capabilities required vector for v_C is a function of B and C, as they are in the goal stack, and of D and E, because they are above C in the initial state. The capabilities required vector will be constructed iteratively as an example. E requires two friction capabilities, thus, [0, 2, 0, 0]. D adds a requirement for a single magnetic capability, [0, 2, 1, 0]. C adds a single encompass end effector, [0, 2, 1, 1]. B requires a single encompass end effector, but an encompass end effector is already part of the capabilities required vector; therefore, the capabilities required vector is not modified. The final capabilities required vector for v_C is [0, 2, 1, 1].

4.2 Rovers

The Rovers domain has been used for several iterations of the International Planning Competition (IPC) [28]. The domain models rovers navigating between waypoints, collecting different classes of scientific data at a subset of waypoints, and communicating the data back to the central lander. The five classes of scientific data are soil analysis, rock analysis, high-resolution imagery, low-resolution imagery, and color imagery. Each rover can independently navigate a subsection of the environment and collect a subset of the classes of scientific data, but only one rover at a time can communicate data to the central lander. Rock analysis is required at a subset of waypoints and soil analysis is required at a subset of waypoints. Rovers must be at a waypoint to perform rock or soil analysis on waypoint and must be equipped for the analysis. Up to three types of imagery data can be collected at each waypoint. A rover must have the correct camera type and the target waypoint must be visible in order for the rover to collect imagery data for the target waypoint. The PDDL implementation of the domain is identical to the simple time version of the domain used in the 2002 International Planning Competition,[2] with the exception of modified action durations.

The state space contains only boolean variables and describes waypoint connectivity, waypoint visibility, rover scientific tools, data collection types and location, central lander location, and communication channel capacity. Each action has a fixed duration. The domain's capability model corresponds to the classes of scientific data being collected. Each rover's capabilities offered vector is a function of the tools available to the rover. The goal is subdivided into a task for each class of scientific data, e.g., all the state constraints concerning rock analysis are grouped into a single task. The capabilities required vector for each task corresponds to the types of scientific data collected for the task.

4.3 Zenotravel

The Zenotravel domain was originally created for testing the Zeno planner [31] and was modified to include hub and spoke airports, passengers and cargo, and short-range and long-range planes. Spoke airports are airports in smaller cities, with each spoke connected to a single hub airport. Hub airports are located in larger cities and are connected to a set of spoke airports. Short-range planes fly only between a hub and its connected spokes. The set of spoke airports for each pair of hubs is disjoint. All hubs are connected and only long-range planes can fly between them. Each plane has limited passenger and cargo capacity. The goal is satisfied when all passengers and cargo are at their destinations. The modified domain has been made freely available.[3]

The state space includes both boolean and continuous variables. The boolean variables describe the location of each passenger, cargo, and plane. The continuous variables include the amount of passengers and cargo on each plane, each plane's passenger and cargo capacity, each plane's fuel level and capacity, and the distance between connected cities. The number of passengers, amount of cargo and their respective capacities for each plane are not continuous variables; however, similar to Blocks World, modeling the values as continuous variables in PDDL facilitates the experiments and expressing the models by not requiring all possible values to be enumerated. The actions to load and unload passengers and cargo from a plane have fixed duration. Fuel use and the action duration for a plane to fly between two cities is a linear function of the distance traveled. The time required to refuel a plane is a linear

2 http://ipc02.icaps-conference.org/CompoDomains/RoversSimpleTime.pddl.

3 https://gist.github.com/aldukeman/1103214e83d47a01b4414699228f9d7d.

function of the fuel level at the start of action execution and the fuel capacity. The capability model includes passenger and cargo capacity and the hub cities. For example, a short-range plane based out of the hub airport of ATL in Atlanta, Georgia has a capabilities offered vector corresponding to its passenger and cargo capacity and its ability to travel between ATL and ATL's spoke airports. A long-range plane has a capabilities offered vector corresponding to its passenger and cargo capacity and its ability to travel between any two hub airports, such as ATL and LAX in Los Angeles, California. The goal state is divided into tasks based on the origin and destination airports of the passengers and cargo. All passengers and cargo originating in a city and traveling to the same city are grouped into a single task. The capabilities required vector of each task is a function of the number of passengers and cargo included in the task, the origin, and the destination.

5 Experimental design

This section describes the experimental design for each tool when solving the hybrid mission planning and coalition formation problem.

5.1 Random problem generation

Grand coalitions and missions were generated for each domain. A grand coalition consists of a set of agents and their associated capabilities. A Mission consists of an initial state and a goal state description. Each grand coalition in each domain was paired with each Mission in the same domain to create a *problem* to be solved. Ten grand coalitions and ten missions were generated for each domain, for a total of 100 generated problems for each domain. The specific experimental details for each domain are presented.

5.1.1 Blocks World

The grand coalitions in the Blocks World domain were a randomly generated set of robotic arms. Four types of end effectors were used: friction, suction, magnetic, and encompass. Each grand coalition had between four and eight arms, with each arm averaging two end effectors. The grand coalitions required at least two arms with each end effector to guarantee the ability to execute each mission. The generated grand coalitions were manually validated as possessing the required end effectors. If a grand coalition was deficient, then the least capable arm in the grand coalition was augmented with the missing end effector(s). The grand coalitions ranged from 4 to 8 arms, with an average of 6.5 arms. Each arm averaged 2.6 end effectors. The mission initial states included between three and five block stacks, with each stack having three blocks, for a total of nine to fifteen blocks. The missions averaged 4.1 stacks of blocks in the initial state. Each mission's goal state description required a random rearrangement of the blocks from the initial block stacks into an equal number of block stacks. The problems generated from the same mission differ in the number of arms and the number of and types of end effectors on the arms. The problems generated from the same grand coalition differ in the number of blocks and the goal state description.

5.1.2 Rovers

The grand coalitions included ten randomly generated rovers. Each rover was allocated tools allowing it to collect an average of two of the five classes of scientific data, defined in Sect. 4.2.

Table 1 Dependent variables

Dependent variable	Units
Makespan	Time
Action execution steps	Scalar
Memory usage	gigabytes
Planning tool time	Seconds

The mission initial states included the connections between the waypoints, the waypoints each rover was able to traverse, each rover's starting location, scientific data source locations, and the central lander's location. The mission goal state description requires all the scientific data to be communicated to the central lander. The missions averaged 103 waypoints, with an average of 4.7 waypoints traversable from each waypoint. Each mission required collecting an average of 116.1 pieces of scientific data. Each grand coalition averaged 4.2 rovers capable of collecting a given class of scientific data, with a minimum of two rovers in each grand coalition capable of collecting each class of scientific data.

5.1.3 Zenotravel

The grand coalitions in the Zenotravel domain were a randomly generated set of long-range planes and short-range planes. Each hub city had between one and three short-range planes and five to ten long-range planes were randomly distributed across the hubs in each mission's initial state. The generated missions use the same set of hubs and spokes, based on real airports in the US and the distances between each. Seven hub airports and forty-two spoke airports were selected, with each hub having between five and seven associated spokes. The missions consisted of an average of 60.1 passengers and 59.7 units of cargo were spread over 17 tasks. The short-range planes had a capacity of four passengers and four cargo units and the long-range planes had a capacity of eight passengers and eight cargo units. The grand coalitions averaged 8.1 long-range planes and 14.5 short-range planes.

5.2 Metrics

A test case is a single problem attempted by a single planning tool. The dependent variables, as presented in Table 1, were recorded during each test case. *Makespan* is the amount of time required to execute a plan:

$$makespan(\pi) = \max_{s \in \pi} start(s) + dur(s),$$

where π is a plan and s is an action execution step in π. The *number of action execution steps* in the generated plan was recorded. *Memory usage* was recorded using the Linux `getrusage` function. The `getrusage` function returns resource usage measures of the current process, including the maximum resident set size, which is an indicator of the amount of memory required by the planning tool. Reported memory values are in gigabytes. *Planning tool time* is the time for the planning tool to produce a solution in seconds.

Three potential outcomes exist for each test case. First, a satisficing plan for the grand coalition to achieve the goal is produced, in which case the metrics are reported. Second, no plan is produced due to a grand coalition being nonexecutable for a mission, which happens when an allocated coalition is confirmed by the planning tool as unable to complete its assigned task. If a coalition is unable to complete its task, then the grand coalition is nonexecutable for the mission. Finally, the planning tool can exceed either memory or computation time limits. All planning problems were limited to 48 GB of memory. Computation time limits were varied and are given with the results. VAL, the plan validator for PDDL [22], was used to confirm that the produced plans were satisficing. The experiments were run under Xubuntu 16.04 using an Intel Core i7-5820K CPU with 64 GB RAM. All source code is written in C++ and compiled with g++ 5.2.1.

5.3 Coalition formation and planning algorithms

Three planning algorithms and three coalition formation algorithms were used with the presented tools. ITSAT [35], a SAT-based planner, was selected for planning in the Rovers domain and a service model approximation algorithm [40] was selected for coalition formation. ITSAT was selected due to being open source after the 2014 International Planning Competition and a desire to test multiple classes of planning algorithms. The service model approximation algorithm provides solutions quickly and supports the service model of coalition formation used in the Rovers domain. TFD [17], a state space search planner using the context-enhanced additive heuristic modified for continuous state variables and temporal planning, was selected for the Blocks World domain and a dynamic programming coalition formation algorithm [40] was selected for coalition formation. TFD was selected due to being open source, performing well in the temporal track of the 2014 International Planning Competition, and supporting the continuous state variables. The dynamic programming algorithm finds solutions for the Blocks World problems quickly and works with either service or resource models. Coalition formation in the Blocks World domain uses the service model. The enforced hill climbing (EHC) version of the COLIN planner [8], a state space search planner using a relaxed plan graph heuristic, supports continuous linear effects and was selected for planning in the Zenotravel domain. COLIN was selected due to being open source and supporting the continuous effects required in the Zenotravel domain. A greedy algorithm [42] was selected for coalition formation in the Zenotravel domain. The greedy algorithm works quickly by limiting potential coalition size.

All three coalition formation algorithms can be applied to the Rovers and Blocks World domain, but the service model approximation algorithm cannot be applied in the Zenotravel domain due to not supporting the resource model of coalition formation. COLIN supports all the necessary features to plan problems in all three domains. TFD can be used to solve problems in the Rovers and Blocks World domains, but does not support the continuous effects present in the Zenotravel domain. ITSAT does not support continuous variables, so it can only be used with the Rovers domain.

6 Planning tool motivation and analysis

This section presents the tools used to solve the randomly generated problems in each domain. Each tool is presented and described using the experimental domains, followed by the experimental results, and a summary of the results and how they motivate the next tool. A summary is located at the end of this section to provide an overview of the results.

6.1 Planning alone

A solution for the generated problems can be derived using existing planning algorithms. The problem is solved as a single planning problem. All agents in the grand coalition are

Algorithm 1: Planning Alone

```
Input  : $S$ - state space, $A$ - coalition action mapping, $\Phi$ - grand coalition, $I$ - initial state, $V$ - tasks
Output: $\pi$ - plan to satisfy all tasks
1 $Actions = A(\Phi)$;
2 $G = \wedge_{v \in V} cond(v)$;
3 $\pi = Plan(S, I, Actions, G)$;
```

available for planning and all task state constraints must be satisfied simultaneously in the solution's end state. Planning for all tasks with all agents simultaneously can consider all possible interactions between the tasks and agents, but planning as a single problem becomes computationally prohibitive as the expressive features of the state model, the number of agents, and the number of tasks increase. As the number of agents increases, so too does the number of actions that can be executed in any given state, the size of the state space, and the number of executable plans that can be considered by the planning algorithm. As the number of tasks increases, more constraints are placed on the goal state and the size of the set of goal states decreases; thus, fewer plans qualify as satisficing. These two effects combine to increase the problem's difficulty.

The problem formalization is translated to a single planning problem by Algorithm 1. The set of available actions for planning, $Actions$, is a function of the agents in the grand coalition, $A(\Phi)$, shown in Line 1. The goals for the planning problem, G, are combined in Line 2. G is satisfied if and only if all the tasks, $v \in V$, are satisfied. $Plan$ in Line 3 represents a planning algorithm capable of reasoning over the action model, the state space (S), and the goals. Each presented tool is agnostic of the underlying planning algorithm. Different planning algorithms are used for each domain, as discussed in Sect. 5.3.

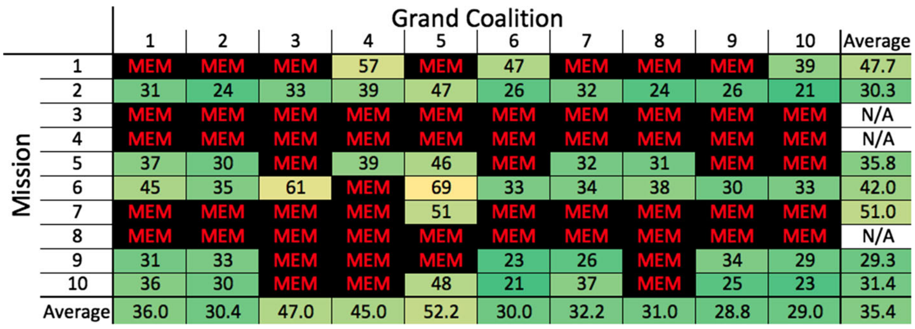
6.1.1 Blocks World

Forty-two of the one hundred generated problems for the Blocks World domain were solved by planning alone. Table 2 presents the five number summary for plan makespan, number of action execution steps, time to derive the plan, and memory required to derive the plan for the solved problems. The temporal makespan of the solved problems is presented in Table 3. A value of "MEM" indicates the memory limit was exceeded while trying to solving the problem. The makespan of the derived plans ranged from 21 to 69, with a median of 33. Plans were not produced for any of the Grand Coalitions in three of the Missions (3, 4, and 8) and a plan was derived for only one of the Grand Coalitions in Mission 7. Planning alone derived plans for all the Grand Coalitions in Mission 2. The number of action execution steps, presented in Table 4, ranged from 46 steps to 139 steps and had a median of 68 steps. The time to derived the satisficing plan for each problem is presented in Table 5 and ranged from 6.1 to 6489.1 s, with a median of 72.2 s. Table 6 presents the memory required during plan derivation for each problem, ranging from 0.05 to 43.91 GB, with a median of 0.49 GB. Planning alone failed to derive plans for 58 of the 100 problems due to exceeding the memory limit and required more than 10 GB of memory when solving nine problems; thus, there is much room for improvement. The heatmap ranges applied for the presented Blocks World results are also applied to the next sets of Blocks World results for comparison of results for the same problems across tools.

Table 2 The five number summary for metrics collected while solving the Blocks World problems using Planning Alone

Metric	Minimum	Lower Quartile	Median	Upper Quartile	Maximum
Makespan	21	29	33	39	69
Steps	46	58	68	82	139
Time (s)	6.1	25.5	72.2	454.4	6489.1
Memory (GB)	0.05	0.20	0.49	6.52	43.91

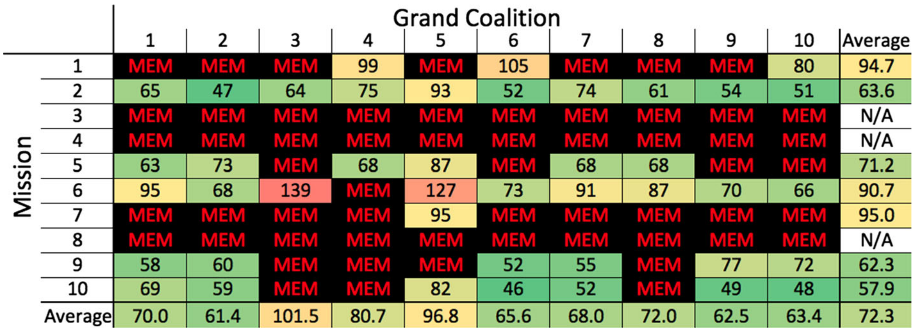
Table 3 Temporal makespan using planning alone for the Blocks World problems (Color table online)



		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	MEM	MEM	MEM	57	MEM	47	MEM	MEM	MEM	39	47.7
	2	31	24	33	39	47	26	32	24	26	21	30.3
	3	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	N/A
	4	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	N/A
	5	37	30	MEM	39	46	MEM	32	31	MEM	MEM	35.8
	6	45	35	61	MEM	69	33	34	38	30	33	42.0
	7	MEM	MEM	MEM	MEM	51	MEM	MEM	MEM	MEM	MEM	51.0
	8	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	N/A
	9	31	33	MEM	MEM	MEM	23	26	MEM	34	29	29.3
	10	36	30	MEM	MEM	48	21	37	MEM	25	23	31.4
	Average	36.0	30.4	47.0	45.0	52.2	30.0	32.2	31.0	28.8	29.0	35.4

The heatmap overlay ranges from 20 (green) to 120 (red). A value of "MEM" indicates the memory limit was exceeded while attempting the problems

Table 4 Number of action execution steps in the derived plans using planning alone for the Blocks World problems (Color table online)



		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	MEM	MEM	MEM	99	MEM	105	MEM	MEM	MEM	80	94.7
	2	65	47	64	75	93	52	74	61	54	51	63.6
	3	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	N/A
	4	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	N/A
	5	63	73	MEM	68	87	MEM	68	68	MEM	MEM	71.2
	6	95	68	139	MEM	127	73	91	87	70	66	90.7
	7	MEM	MEM	MEM	MEM	95	MEM	MEM	MEM	MEM	MEM	95.0
	8	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	N/A
	9	58	60	MEM	MEM	MEM	52	55	MEM	77	72	62.3
	10	69	59	MEM	MEM	82	46	52	MEM	49	48	57.9
	Average	70.0	61.4	101.5	80.7	96.8	65.6	68.0	72.0	62.5	63.4	72.3

The heatmap overlay ranges from 40 (green) to 150 (red). A value of "MEM" indicates the memory limit was exceeded while attempting the problems

6.1.2 Rovers

All one hundred Rovers problems were solved by Planning Alone. Table 7 presents the five number summary for plan makespan, number of actions, time to derive the plans, and memory required to derive the plans. The makespan of the derived solutions, presented in Table 8, ranged from 876 to 3463, with a median of 1585. Grand Coalition 10 had the shortest average

Table 5 Planning time in seconds using planning alone for the Blocks World problems (Color table online)

		Grand Coalition									
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	MEM	MEM	MEM	2373	MEM	6489	MEM	MEM	MEM	5043	4634.8
	2	383	138	106	569	1451	82	1780	45	250	34	483.8
	3	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	N/A
	4	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	N/A
	5	15	723	MEM	63	153	MEM	1667	85	MEM	MEM	450.8
	6	29	7	1850	MEM	235	36	25	114	24	33	261.4
	7	MEM	MEM	MEM	MEM	478	MEM	MEM	MEM	MEM	MEM	478.3
	8	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	N/A
	9	10	54	MEM	MEM	MEM	50	18	MEM	36	175	57.3
	10	8	7	MEM	MEM	551	6	14	MEM	10	32	89.6
	Average	88.9	186.0	978.0	1001.5	573.7	1332.7	700.6	81.0	79.9	1063.4	601.2

The heatmap overlay ranges from 0 (green) to 1800 s (red). A value of "MEM" indicates the memory limit was exceeded while attempting the problems

Table 6 Required memory in gigabytes using planning alone for the Blocks World problems (Color table online)

		Grand Coalition									
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	MEM	MEM	MEM	27.06	MEM	25.80	MEM	MEM	MEM	14.17	22.34
	2	7.04	0.58	1.82	11.21	43.91	0.39	26.39	0.35	2.41	0.21	9.43
	3	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	N/A
	4	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	N/A
	5	0.21	4.17	MEM	0.83	4.95	MEM	23.33	0.73	MEM	MEM	5.70
	6	0.27	0.06	28.17	MEM	4.18	0.20	0.16	0.85	0.16	0.22	3.81
	7	MEM	MEM	MEM	MEM	9.49	MEM	MEM	MEM	MEM	MEM	9.49
	8	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	N/A
	9	0.09	0.29	MEM	MEM	MEM	0.23	0.11	MEM	0.22	0.73	0.28
	10	0.12	0.07	MEM	MEM	13.15	0.05	0.12	MEM	0.07	0.20	1.97
	Average	1.55	1.03	15.00	13.03	15.14	5.33	10.02	0.64	0.72	3.11	6.07

The heatmap overlay ranges from 0 (green) to 40 GB (red). A value of "MEM" indicates the memory limit was exceeded while attempting the problems

Table 7 The five number summary for metrics collected while solving the Rovers problems using planning alone

Metric	Minimum	Lower Quartile	Median	Upper Quartile	Maximum
Makespan	876	1349	1585	1927	3463
Steps	481	668	759	902	1403
Time (s)	1029.0	1505.1	1790.4	2227.0	13033.2
Memory (GB)	12.41	15.65	17.29	20.08	40.75

makespan, while Grand Coalition 1 had the longest. Mission 10 had the longest average makespan, while Mission 3 had the shortest. The number of action execution steps for each problem is presented in Table 9 and ranged from 481 steps to 1403 steps, with a median of 759 steps. Mission 10 required the most steps (998), while Mission 3 with 556 steps required the least. Grand Coalition 1 required the most steps to complete a Mission (998), while Grand Coalition 10 with 674 steps required the least. Table 10 presents the time to derive

Table 8 Temporal makespan using planning alone for the Rovers problems (Color table online)

		Grand Coalition									
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	1519	1280	2661	1357	1515	1867	1580	1850	1452	1256	1633.7
	2	2239	1457	1751	1178	1247	1790	1528	1796	1044	1086	1511.6
	3	1400	922	1366	1098	1079	1294	1006	1264	905	876	1121
	4	3150	1840	2494	1814	1589	2235	1968	2393	1795	1786	2106.4
	5	2992	1488	1506	1412	1344	1712	2011	1926	1278	1196	1686.5
	6	3262	1541	2057	1594	1375	1816	1930	2042	1230	1247	1809.4
	7	1564	1364	2619	1214	1173	1568	1710	1860	1350	1133	1555.5
	8	1888	1351	2139	1271	1357	1842	1626	1744	1329	1459	1600.6
	9	2369	1506	3184	1341	1489	2062	2006	2311	1399	1362	1902.9
	10	3463	1865	2411	1849	1739	2074	2198	2612	1780	1744	2173.5
	Average	2384.6	1461.4	2218.8	1412.8	1390.7	1826	1756.3	1979.8	1356.2	1314.5	1710.1

The heatmap overlay ranges from a value of 800 (green) up to a value of 3500 (red)

Table 9 Number of action execution steps in the derived plans using planning alone for Rovers problems (Color table online)

		Grand Coalition									
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	792	657	1187	668	734	861	762	800	725	655	784.1
	2	889	675	805	621	643	785	663	737	581	582	698.1
	3	673	481	620	531	534	597	530	625	490	483	556.4
	4	1403	875	1120	886	782	999	959	966	850	867	970.7
	5	1032	738	752	678	625	755	913	838	663	672	766.6
	6	1284	729	909	791	726	840	796	952	666	633	832.6
	7	723	622	1066	603	609	735	718	729	639	568	701.2
	8	900	675	865	686	720	870	783	833	715	712	775.9
	9	1089	797	1160	691	789	883	1014	942	739	740	884.4
	10	1193	937	1078	974	922	909	971	1240	923	831	997.8
	Average	997.8	718.6	956.2	712.9	708.4	823.4	810.9	866.2	699.1	674.3	796.8

The heatmap overlay ranges from 480 (green) to 1400 (red)

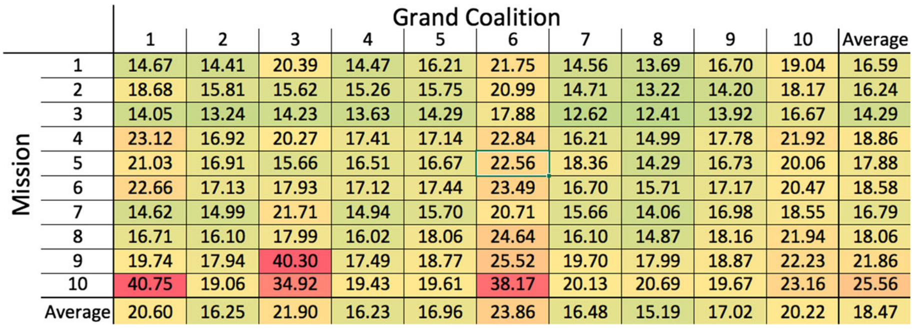
Table 10 Planning time in seconds using planning alone for the Rovers problems (Color table online)

		Grand Coalition									
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	1151	1248	2344	1242	1555	2410	1542	1757	1634	1620	1650.2
	2	1998	1507	1423	1371	1518	2396	1719	1686	1329	1564	1651.2
	3	1060	1131	1190	1232	1474	2173	1363	1326	1187	1524	1365.9
	4	2630	1666	2204	1832	1671	2713	1996	2176	1813	2207	2090.7
	5	2475	1954	1621	1832	1862	3015	2369	2029	1694	2166	2101.6
	6	2673	1740	1800	1691	1670	2938	2065	2089	1330	1889	1988.4
	7	1029	1292	2497	1238	1446	2549	1858	1781	1350	1525	1656.4
	8	1460	1282	1891	1278	1590	2587	1500	1697	1430	1850	1656.4
	9	2313	1728	13033	1580	1965	3061	2420	2459	1761	2155	3247.6
	10	3898	2015	10596	2185	2286	10782	2553	2821	2046	2473	4165.5
	Average	2068.8	1556.2	3859.9	1548.0	1703.7	3462.3	1938.4	1982.0	1557.3	1897.2	2157.4

The heatmap overlay ranges from 800 (green) to 3100 s (red)

the satisficing plan for each problem, ranging from 1118.0 to 13475.4 s, with a median of 1790.4 s. Deriving plans for Mission 10 required an average of 5123 s, the highest of all the

Table 11 Required memory in gigabytes using planning alone for the Rovers problems (Color table online)



		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	14.67	14.41	20.39	14.47	16.21	21.75	14.56	13.69	16.70	19.04	16.59
	2	18.68	15.81	15.62	15.26	15.75	20.99	14.71	13.22	14.20	18.17	16.24
	3	14.05	13.24	14.23	13.63	14.29	17.88	12.62	12.41	13.92	16.67	14.29
	4	23.12	16.92	20.27	17.41	17.14	22.84	16.21	14.99	17.78	21.92	18.86
	5	21.03	16.91	15.66	16.51	16.67	22.56	18.36	14.29	16.73	20.06	17.88
	6	22.66	17.13	17.93	17.12	17.44	23.49	16.70	15.71	17.17	20.47	18.58
	7	14.62	14.99	21.71	14.94	15.70	20.71	15.66	14.06	16.98	18.55	16.79
	8	16.71	16.10	17.99	16.02	18.06	24.64	16.10	14.87	18.16	21.94	18.06
	9	19.74	17.94	40.30	17.49	18.77	25.52	19.70	17.99	18.87	22.23	21.86
	10	40.75	19.06	34.92	19.43	19.61	38.17	20.13	20.69	19.67	23.16	25.56
	Average	20.60	16.25	21.90	16.23	16.96	23.86	16.48	15.19	17.02	20.22	18.47

The heatmap overlay ranges from 0 (green) to 40 GB (red)

Missions, while Mission 3 required the least, at 1387 s. Memory usage ranged from 12.41 to 40.75 GB, with a median of 17.29 GB, and is presented in Table 11. Mission 10 required the most memory (25.56 GB), while Mission 3 with 14.29 GB required the least. Grand Coalition 6 required the most memory (23.86 GB), while Grand Coalition 8 at 15.19 GB required the least. The large amount of memory and time required to solve these problems leaves room for improvement. The same heatmap ranges are applied to the next sets of Rovers results.

Missions 3 and 10 were the hardest and easiest Missions, respectively, in terms of time and memory required to derive a plan and in terms of derived plan makespan and number of steps. The average number of scientific data collections per Mission was 116. Mission 10 (149 data collections) was 1.64 standard deviations above the average, while Mission 3 (92 data collections) was 1.25 standard deviations below the average. The hardest and easiest Grand Coalitions for which to plan in terms of the time and memory limitations, makespan, and number of steps in the derived plan is much less clear than the equivalent analysis with the Missions. The average Grand Coalition offered 21 capabilities. Grand Coalition 10 was the most capable with a total of 28 capabilities, with at least five of the ten rovers being capable of collecting each class of scientific data. The greater capabilities translated to better plans in terms of makespan and number of steps; however, the greater capabilities did not require the most time nor memory with which to plan. Grand Coalition 8 was the least capable of the coalitions, with three rovers capable of collecting each class of scientific data (total of 15 capabilities offered), but the plans derived for Grand Coalition 8 were not the longest, in terms of makespan or number of steps. The derived plans for Grand Coalition 1 were the longest in terms of makespan and number of steps. The reason for the difference is the number of rovers capable of soil analysis. Both rock and soil analyses require more navigation throughout the environment than the imagery analyses, because the rover must be at the waypoint to perform the analysis, while the imagery analyses only require that the rover have line of sight to the waypoint. Grand Coalition 8 had three rovers capable of soil analysis, while Grand Coalition 1 had two rovers capable of soil analysis.

6.1.3 Zenotravel

None of the Zenotravel problems were solved by planning alone using the COLIN planner in EHC mode. A 2 h time limit was enforced for solving the Zenotravel problems, which resulted in no satisficing plans being generated for any of the problems. Five of the one hundred problems were selected to reevaluate without the time limit. Each of the five problems

Table 12 Planning alone summary

Domain	Problems solved	Summary
Blocks World	42	Some problems solved, but the majority of the problems exceeded the memory limit. No plans were found for three missions
Rovers	100	All problems solved, but high computational resource usage
Zenotravel	0	No problems solved within time limit. Ten problems were run without a time limit and all problems exceeded the memory limit after 16.5 h

exceeded the 48 GB memory limit after an average of 16.5 h of execution. The problems being solved in the Zenotravel domain average 60 passengers, 60 cargo, and 22 planes. The large number of planes, passengers, and cargo produces a large branching factor. Assume, as a conservative estimate, each plane can refuel, fly to three different cities, or load a single passenger or cargo. The branching factor for such a situation is over 100. Each plane is likely to be able to fly to about five cities from any particular state and load or unload cargo and passengers, thus, the true branching factor is likely to be much higher.

6.1.4 Summary

The results from each domain provide room for improvement, as summarized in Table 12. Results from each of the domains leave room for improvement. Only 42 of the 100 Blocks World problems were solved and no plans were produced for three of the Missions. All 58 failures for the Blocks World problems were due to exceeding the 48 GB memory limit. All Rovers domain problems were solved by ITSAT, but computational resource usage needs to be improved. ITSAT required, on average, 2157 s and 18.5 GB of memory to solve the problems. None of the Zenotravel domain problems were solved within the 2 h time limit and removing the time limit did not allow for any of the problems to be solved.

The primary issue when using planning alone is exponential complexity. The number of agents, the number of tasks, and the domain complexity (durative actions, continuous state variables, etc.) all contribute to the problem complexity. One option is to reduce the domain complexity through various relaxations, such as assuming all actions are instantaneous and executed sequentially. A plan is only as good as the domain and problem description from which it is created; thus, plans are more likely to succeed in real-world missions when they are created from accurate domain models with representative durative actions and continuous variables. A better option is to address the other aspects of problem complexity, i.e., reduce the number of agents and number of tasks considered at any one time. The three presented tools address problem complexity by reducing the number of goals and agents considered during planning, while maintaining high fidelity state models and plans.

6.2 Coalition formation then planning

The coalition formation then planning tool (CFP) produces several smaller planning instances focused on a subset of the goals, each using a subset of the agents to satisfy the goals. Algorithm 2 presents the CFP algorithm, which begins with an empty plan and no goals,

Algorithm 2: Coalition Formation then Planning

```
Input  : $S$ - state space, $A$ - coalition action mapping, $\Phi$ - grand coalition, $I$ - initial state, $V$ - tasks,
         $C$ - capability mappings
Output: $\pi$ - plan to satisfy all tasks
1  $\pi = \varnothing$;
2  $G = \varnothing$;
3  $\{Cap^{\phi}\}_{\phi \in \Phi} = C(\Phi)$;
4  $\{Cap^{v}\}_{v \in V} = C(V)$;
5  $\{\Phi_i, v_i\}_{i=1}^{|V|} = CF(\{Cap^{\phi}\}_{\phi \in \Phi}, \{Cap^{v}\}_{v \in V})$;
6  foreach $i \in \{1, \ldots, |V|\}$ do
7      $G = G \wedge v_i$;
8      $I_i = Simulate(I, \pi)$;
9      $Actions = A(\Phi_i)$;
10     $\pi_i = Plan(S, I_i, Actions, G)$;
11     $\pi = PlanMerge(S, I, \pi, \pi_i, G)$;
12 end
```

Lines 1 and 2, respectively. The capabilities offered vector for each agent is identified using the capability mappings in Line 3. The capabilities required vector for each task is identified using the capability mappings in Line 4. Coalition formation is applied in Line 5 to allocate coalitions to tasks. Coalition formation's result is an assignment of a candidate coalition to each task. A coalition is a candidate coalition for a task if and only if the coalition has at least as many of each type of resource as required by task, i.e., Φ_v is a candidate coalition for v if and only if $\forall i$, $Cap_i^{\Phi_v} \geq Cap_i^{v}$. The planning loop, Line 6–12, is executed after coalition formation. The goals for v_i are combined with G to form the goals to be solved in the current iteration, Line 7. Combining the previous constraints with the current iteration constraints allows the iterative plan to break previous constraints in the course of planning, as long as the constraints are satisfied at the end of the iterative planning. The initial state, I_i, for the current iteration is the end state achieved after simulating the current plan, π, from the initial state, I, using VAL, Line 8. The available actions, $Actions$, are a function of the coalition allocated to the current task, Φ_i, and available for planning, Line 9. An appropriate planning algorithm, $Plan$, finds an iterative plan, π_i, to satisfy G from the initial state, I_i, using the actions of the available coalition, $Actions$, in Line 10. CFP relies on coalition formation to produce executable coalitions. If Φ_i is a nonexecutable coalition, then CFP reports the problem as failed, else the iterative plan must be merged. π_i is merged with the current plan, π, to create a plan to satisfy G when executed from I, Line 11. The planning problem solved during each iteration in Line 10 is analogous to deriving a plan, executing the plan, and being given an additional planning goal to satisfy. The current goals, G, have been satisfied in the current state, I_i, but additional goals are given, so augmenting G with the additional goal constraints, $G = G \wedge v_i$, and a plan must be derived to transition from I_i to a state satisfying G. The plan merge step, Line 11, can be as simple as modifying the action execution steps in π_i, such that the action execution steps begin execution immediately when π ends; however, more complex scheduling can occur. A greedy scheduling approach, in which each action execution step in π_i is modified, one at a time in increasing original start time order, to occur as early as possible in the resulting plan, is applied.

Performing coalition formation affects problem complexity in two ways. First, the number of goal constraints addressed at any one time is reduced. Assume the Blocks World example from Fig. 1b. The set of goal constraints for planning alone addresses the seven blocks in the figure, whereas the goal constraints in CFP are divided into three sets of goals constraints, two of which address the locations of two blocks and one of which addresses the location of three blocks. Each of the stacks of blocks in the goal state description of the Blocks World problems correspond to a task. Planning for a single stack at a time prunes blocks from the state space that must be considered. Second, the number of agents is reduced; thus, the number of actions and the state space is reduced. Reducing the number of actions creates a lower branching factor in the search tree, but also eliminates states from the search tree. Goal states can be among the eliminated states; thus, reducing the search branching factor can force deeper searches to identify a goal state. The state space is reduced due to eliminating the variables and domain values that are no longer reachable given the reduced action set. The effects of the reduced action set and reduced state space combine to increase the number of states that can be searched per unit of time. Coalition formation in the Blocks World problems identifies a coalition of arms to allocate to each task. Each identified coalition is selected based on the end effectors available to the coalition and the end effectors required to achieve each block stack. If the grand coalition is not allocated to the task, then the branching factor in planning for the task has been reduced.

6.2.1 Blocks World

Twenty-six of the one hundred generated problems for the Blocks World domain were solved by CFP. Table 13 presents the five number summary for plan makespan, number of action execution steps, time required to derive the plan, and memory required to derive the plan for the solved problems. The temporal makespan of the derived plans are presented in Table 14. Values of "NE", "MEM", and "TIME" indicate a nonexecutable coalition, an exceeded memory limit, and an exceeded time limit, respectively. The time limit was 1 h. The makespan ranged from 30 to 114, with a median of 52. Grand Coalition 10 was the only Grand Coalition for which plans were not derived for any of the Missions and Mission 9 was the only Mission for which plans were not derived for any of the Grand Coalitions. The number of action execution steps ranged from 37 steps to 153 steps, with a median of 84 steps, and are presented in Table 15. The time to derive a plan had a median of 103.1 s, ranging from 2.6 to 3093.4 s, and are presented in Table 16. The memory required to derive a plan is presented in Table 17. The memory required ranged from 0.01 to 22.86 GB, with a median of 0.36 GB.

Seventeen problems were commonly solved by PA and CFP. A comparison of the seventeen problems is presented in Table 18. The ratio of the metric when using PA to the metric when using CFP was calculated, and the median ratio is presented. The median makespan ratio was 1.46, with 11 of the problems having higher makespan solutions (ratio $>$ 1.0) when solved using CFP compared to PA. Ten of the CFP derived plans had more actions than the counterparts derived using PA, with a median ratio of 1.06. Computation time was lower with CFP than with PA for 12 of the problems (ratio $<$ 1.0), with a median ratio of 0.71.

Table 13 The five number summary for solving the Blocks World problems using CFP

Metric	Minimum	Lower Quartile	Median	Upper Quartile	Maximum
Makespan	30	42	52	61	114
Steps	37	73	84	93	153
Time (s)	2.6	33.9	103.1	494.8	3093.4
Memory (GB)	0.01	0.15	0.36	8.35	22.86

Table 14 Temporal makespan using CFP for the Blocks World problems (Color table online)

		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	50	NE	MEM	55	56	47	65	NE	NE	NE	54.6
	2	61	NE	35	62	MEM	47	53	NE	NE	NE	51.6
	3	NE	NE	NE	NE	NE	NE	NE	72	NE	NE	72.0
	4	NE	71	NE	NE	NE	NE	NE	NE	NE	NE	71.0
	5	NE	NE	NE	57	41	NE	NE	46	57	NE	50.3
	6	NE	NE	47	NE	NE	NE	71	NE	NE	NE	59.0
	7	NE	NE	51	NE	61	NE	38	NE	NE	NE	50.0
	8	NE	NE	NE	NE	NE	NE	NE	NE	114	NE	114.0
	9	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE	N/A
	10	35	NE	MEM	MEM	30	32	NE	TIME	38	NE	33.8
	Average	48.7	71.0	44.3	58.0	47.0	42.0	56.8	59.0	69.7	N/A	53.5

The heatmap overlay ranges from 20 (green) to 120 (red). A value of "MEM", "NE", or "TIME" indicates the problem was not solved due to exceeding the memory limit, generating a nonexecutable coalition, or execeeding the time limit

Table 15 Number of action execution steps in the derived plans using CFP for the Blocks World problems (Color table online)

		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	79	NE	MEM	79	85	94	86	NE	NE	NE	84.6
	2	96	NE	64	89	MEM	76	77	NE	NE	NE	80.4
	3	NE	NE	NE	NE	NE	NE	NE	116	NE	NE	116.0
	4	NE	120	NE	NE	NE	NE	NE	NE	NE	NE	120.0
	5	NE	NE	NE	83	74	NE	NE	86	87	NE	82.5
	6	NE	NE	73	NE	NE	NE	116	NE	NE	NE	94.5
	7	NE	NE	86	NE	101	NE	69	NE	NE	NE	85.3
	8	NE	NE	NE	NE	NE	NE	NE	NE	153	NE	153.0
	9	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE	N/A
	10	54	NE	MEM	MEM	37	53	NE	TIME	55	NE	49.8
	Average	76.3	120.0	74.3	83.7	74.3	74.3	87.0	101.0	98.3	N/A	84.2

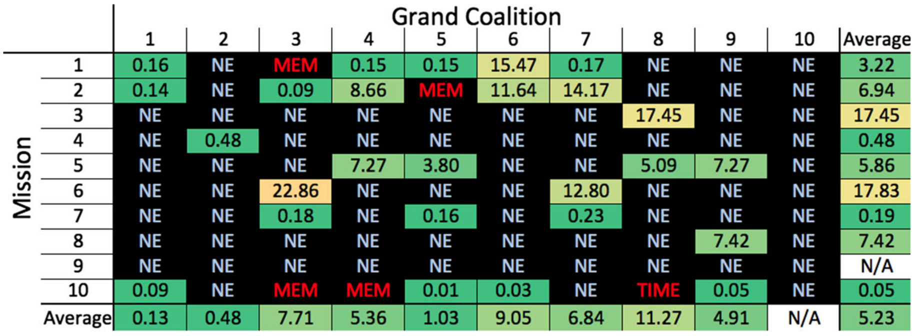
The heatmap overlay ranges from 40 (green) to 150 (red). A value of "MEM", "NE", or "TIME" indicates the problem was not solved due to exceeding the memory limit, generating a nonexecutable coalition, or execeeding the time limit

Table 16 Planning time in seconds using CFP for the Blocks World problems (Color table online)

		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	44	NE	MEM	48	41	1047	87	NE	NE	NE	253.2
	2	16	NE	12	406	MEM	525	547	NE	NE	NE	300.9
	3	NE	NE	NE	NE	NE	NE	NE	3093	NE	NE	3093.4
	4	NE	390	NE	NE	NE	NE	NE	NE	NE	NE	390.1
	5	NE	NE	NE	303	119	NE	NE	222	333	NE	244.3
	6	NE	NE	1457	NE	NE	NE	1086	NE	NE	NE	1271.4
	7	NE	NE	42	NE	32	NE	80	NE	NE	NE	51.1
	8	NE	NE	NE	NE	NE	NE	NE	NE	1739	NE	1738.8
	9	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE	N/A
	10	11	NE	MEM	MEM	3	6	NE	TIME	6	NE	6.3
	Average	23.4	390.1	503.4	252.1	48.6	525.7	450.0	1657.9	692.4	N/A	449.7

The heatmap overlay ranges from 0 (green) to 1800 s (red). A value of "MEM", "NE", or "TIME" indicates the problem was not solved due to exceeding the memory limit, generating a nonexecutable coalition, or execeeding the time limit

Table 17 Required memory in gigabytes using CFP for the Blocks World problems (Color table online)



Mission	Grand Coalition 1	2	3	4	5	6	7	8	9	10	Average
1	0.16	NE	MEM	0.15	0.15	15.47	0.17	NE	NE	NE	3.22
2	0.14	NE	0.09	8.66	MEM	11.64	14.17	NE	NE	NE	6.94
3	NE	NE	NE	NE	NE	NE	NE	17.45	NE	NE	17.45
4	NE	0.48	NE	NE	NE	NE	NE	NE	NE	NE	0.48
5	NE	NE	NE	7.27	3.80	NE	NE	5.09	7.27	NE	5.86
6	NE	NE	22.86	NE	NE	NE	12.80	NE	NE	NE	17.83
7	NE	NE	0.18	NE	0.16	NE	0.23	NE	NE	NE	0.19
8	NE	NE	NE	NE	NE	NE	NE	NE	7.42	NE	7.42
9	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE	N/A
10	0.09	NE	MEM	MEM	0.01	0.03	NE	TIME	0.05	NE	0.05
Average	0.13	0.48	7.71	5.36	1.03	9.05	6.84	11.27	4.91	N/A	5.23

The heatmap overlay ranges from 0 (green) to 40 GB (red). A value of "MEM", "NE", or "TIME" indicates the problem was not solved due to exceeding the memory limit, generating a nonexecutable coalition, or execeeding the time limit

Table 18 Median ratio of the CFP metric value to the PA metric value for Blocks World problems

Problems	Makespan ratio	Actions ratio	Time ratio	Memory ratio
17	1.46	1.06	0.71	0.71

Only problems solved by both tools are considered

Finally, thirteen of the problems required less memory to solve with CFP than with PA, with a median ratio of 0.71.

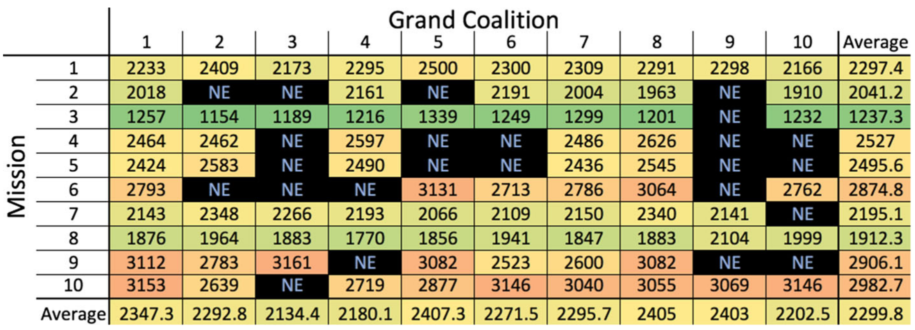
The plan merge step ensures that all actions execution steps in π_i do not occur until their conditions are satisfied in π. For example, if block A is placed on the table as a result of executing step m in π and a step n in π_i relies on A being on the table, then the plan merge step ensures that n does not start until after A has been placed on the table by m.

CFP failed to solve nine problems due to exceeding the memory limit and failed to solve 66 of the 100 problems due to allocating nonexecutable coalitions. CFP solved fewer problems than planning alone, but the median problems required 29% less memory and 29% less time than planning alone. The problems that failed due to nonexecutable coalitions can be divided into two categories, both of which are a result of the order in which tasks were planned. First, arms were not required to drop blocks at the end of their tasks. The finite table limited the number of blocks on the table and forced agents to place blocks on top of one another. If block a was placed on top of b and the task required moving b, but the coalition was incapable of moving a due to no agent in the coalition possessing the required end effector, then the coalition was nonexecutable. Second, agents were not required to release blocks at the end of a task. If arm ϕ is holding a at the end of plan execution and v_j requires a for a later plan, then Φ_j allocated to v_j will fail planning, unless ϕ releases a before the algorithm plans for v_j or $\phi \in \Phi_j$. If ϕ releases a before the algorithm plans for v_j, then a will be a part of some tower in I_j. If $\phi \in \Phi_j$, then planning will succeed due to ϕ holding a in I_j and ϕ being able to place a where it belongs. One option to avoid the second failure mode is to require agents to satisfy "cleanup goals" before planning each task. The cleanup goals will ensure that each agent is prepared to plan for the next task. Cleanup goals in the Blocks World problems will require that all agents not hold any blocks at the end of task planning. A design decision was made not to test cleanup goals for these experiments. Requiring agents

Table 19 The five number summary for solving the Rovers problems using CFP

Metric	Minimum	Lower Quartile	Median	Upper Quartile	Maximum
Makespan	1154	1990	2299	2658	3161
Steps	486	578	656	722	935
Time (s)	812.3	1036.0	1296.1	1545.3	2292.8
Memory (GB)	0.93	1.45	1.88	2.41	4.41

Table 20 Temporal makespan using CFP for the Rovers problems (Color table online)



		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	2233	2409	2173	2295	2500	2300	2309	2291	2298	2166	2297.4
	2	2018	NE	NE	2161	NE	2191	2004	1963	NE	1910	2041.2
	3	1257	1154	1189	1216	1339	1249	1299	1201	NE	1232	1237.3
	4	2464	2462	NE	2597	NE	NE	2486	2626	NE	NE	2527
	5	2424	2583	NE	2490	NE	NE	2436	2545	NE	NE	2495.6
	6	2793	NE	NE	NE	3131	2713	2786	3064	NE	2762	2874.8
	7	2143	2348	2266	2193	2066	2109	2150	2340	2141	NE	2195.1
	8	1876	1964	1883	1770	1856	1941	1847	1883	2104	1999	1912.3
	9	3112	2783	3161	NE	3082	2523	2600	3082	NE	NE	2906.1
	10	3153	2639	NE	2719	2877	3146	3040	3055	3069	3146	2982.7
Average		2347.3	2292.8	2134.4	2180.1	2407.3	2271.5	2295.7	2405	2403	2202.5	2299.8

The heatmap overlay ranges from 800 (green) to 3500 (red). A value of "NE" indicates a nonexecutable coalition failure

to drop blocks between tasks can detrimentally affect plan quality. Consider the ϕ holding a case from earlier. Assume ϕ is holding a, v_j is the next task to be planned, and Φ_j is allocated to v_j. If $\phi \in \Phi_j$ and a is a part of v_j, then requiring ϕ to drop a before planning v_j wastes time, since some agent in Φ_j will pickup a again.

The results also demonstrate that coalition formation can be detrimental. Five problems (Mission 2 with Grand Coalition 5, Mission 5 with Grand Coalitions 5 and 7, Mission 6 with Grand Coalition 5, and Mission 10 with Grand Coalition 3) were solved by planning alone, but exceeded the memory limit when attempted by CFP. More states had to be created and searched because the decrease in branching factor caused by using fewer agents did not offset the increase in the search depth required to solve the problems.

6.2.2 Rovers

Seventy-six of the one hundred generated Rovers problems were solved by CFP. The five number summary for plan makespan, number of action execution steps, time required to derive the plan, and memory required to derive the plan for the solved problems are presented in Table 19. The temporal makespan of the derived plans ranged from 1154 to 3161 and had a median of 2299. Full makespan results are presented in Table 20. A value of "NE" indicates a nonexecutable coalition was allocated. The number of action execution steps is presented in Table 21 and ranged from 486 steps to 935 steps, with a median of 656 steps. The computation time and memory usage is presented in Tables 22 and 23, respectively. Computation time ranged from 812 to 2292.8 s and had a median of 1296.1 s. Memory usage ranged from 0.93 to 4.41 GB, with a median of 1.88 GB.

Table 21 Number of action execution steps in the derived plans using CFP for Rovers problems (Color table online)

		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	669	657	665	672	658	641	675	655	638	640	657.0
	2	618	NE	NE	602	NE	598	576	578	NE	603	595.8
	3	570	503	530	486	521	503	514	529	NE	513	518.8
	4	887	795	NE	837	NE	NE	837	810	NE	NE	833.2
	5	651	646	NE	655	NE	NE	624	646	NE	NE	644.4
	6	720	NE	NE	NE	726	679	671	706	NE	699	700.2
	7	567	543	573	550	569	528	547	577	553	NE	556.3
	8	721	641	704	650	653	660	634	615	666	671	661.5
	9	772	703	727	NE	732	693	695	734	NE	NE	722.3
	10	935	808	NE	818	863	861	864	848	871	915	864.8
	Average	711.0	662.0	639.8	658.8	674.6	645.4	663.7	669.8	682.0	673.5	669.3

The heatmap overlay ranges from 480 (green) to 1400 (red). A value of "NE" indicates a nonexecutable coalition failure

Table 22 Planning time in seconds using CFP for the Rovers problems (Color table online)

		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	1257	1241	1244	1238	1265	1323	1321	1237	1184	1245	1255.5
	2	1152	NE	NE	1046	NE	1140	985	951	NE	1089	1060.3
	3	901	853	923	844	908	872	812	827	NE	922	873.5
	4	2129	1831	NE	2015	NE	NE	2048	1793	NE	NE	1963.1
	5	1162	1406	NE	1340	NE	NE	1363	1340	NE	NE	1322.0
	6	1391	NE	NE	NE	1527	1615	1411	1431	NE	1547	1487.0
	7	906	996	1044	918	999	1011	957	980	986	NE	977.4
	8	1442	1269	1401	1292	1300	1362	1208	1172	1265	1361	1307.2
	9	1639	1474	1519	NE	1660	1565	1545	1524	NE	NE	1560.9
	10	2293	1774	NE	1991	2054	2131	1997	1959	1918	2243	2039.9
	Average	1426.9	1355.4	1226.4	1335.6	1387.6	1377.4	1364.7	1321.2	1338.4	1401.0	1358.9

The heatmap overlay ranges from 800 (green) to 3100 s (red). A value of "NE" indicates a nonexecutable coalition failure

Table 23 Required memory in gigabytes using CFP for the Rovers problems (Color table online)

		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	1.40	1.17	1.37	1.87	2.09	2.25	1.63	1.92	2.37	2.02	1.81
	2	1.57	NE	NE	1.32	NE	2.46	1.85	1.11	NE	1.71	1.67
	3	1.01	1.22	1.18	1.29	1.18	1.72	1.19	1.05	NE	1.33	1.24
	4	2.19	1.80	NE	2.01	NE	NE	2.56	2.04	NE	NE	2.12
	5	1.77	2.97	NE	1.81	NE	NE	2.64	1.58	NE	NE	2.15
	6	1.90	NE	NE	NE	2.38	3.06	3.27	2.41	NE	3.31	2.72
	7	0.93	1.18	1.53	1.40	1.70	2.84	1.51	1.67	1.95	NE	1.63
	8	1.27	1.59	1.31	1.35	1.46	2.02	1.75	1.56	2.03	2.11	1.65
	9	2.14	1.88	1.76	NE	2.46	3.31	2.46	2.57	NE	NE	2.37
	10	3.03	2.82	NE	2.41	2.22	4.41	3.35	2.29	2.70	4.37	3.07
	Average	1.72	1.83	1.43	1.68	1.93	2.76	2.22	1.82	2.26	2.48	2.00

The heatmap overlay ranges from 0 (green) to 40 GB (red). A value of "NE" indicates a nonexecutable coalition failure

Table 24 Median ratio of the CFP metric value to the PA metric value for Rovers problems

Problems	Makespan ratio	Actions ratio	Time ratio	Memory ratio
76	1.34	0.85	0.73	0.11

Only problems solved by both tools are considered

A comparison of the set of 76 problems commonly solved by PA and CFP is shown in Table 24. The median makespan ratio was 1.34, with 62 of the problems having higher makespan solutions (ratio > 1.0) when solved using CFP compared to PA. Six of the CFP derived plans had more actions than the corresponding plans derived using PA, with a median ratio of 0.85. Computation time was lower with CFP than with PA for 71 of the problems (ratio < 1.0), with a median ratio of 0.73. Finally, all 76 of the problems required less memory to solve with CFP than with PA, with a median ratio of 0.11.

The plan merge step in the Rovers domain is simple as the rovers are mostly independent. The difficulty comes from the single shared communications channel with the lander. A simple merge where each plan is executed concurrently does not work because the communications channel is disrupted if multiple rovers attempt to transmit at the same time. The merged plan must ensure no two rovers are ever concurrently transmitting with the lander.

The twenty-four problems that were not solved by CFP due to a nonexecutable coalition being assigned to a task failed due to not being able to reach required waypoints to collect scientific data. Most of the instances were due to a nonexecutable coalition assigned to the rock or soil analysis task. A rover must be at waypoint w_i to collect rock or soil analysis at w_i. If no rover in the coalition can reach w_i, then the coalition is nonexecutable. The imagery analyses are less likely to be nonexecutable due to a rover not being required to be at w_i to collect imagery data for w_i. Imagery data for each waypoint requiring imagery analysis was collectable from multiple waypoints. Multiple specific waypoints must be unreachable in order for a coalition to fail this task whereas a single waypoint being unreachable can make rock or soil analysis nonexecutable.

6.2.3 Zenotravel

Eighty-five of the one hundred generated problems were solved by CFP using the resource model greedy coalition formation algorithm and the COLIN planner in EHC mode. The fifteen problems were due to using the incomplete EHC-based planning algorithm. The five number summary for plan makespan, number of steps, time to derive the plan, and memory required to derive the plan for the solved problems are presented in Table 25. Derived plan makespan is presented in Table 26. One Grand Coalition completed all Missions (Grand Coalition 9), while five Missions were completed by all Grand Coalitions (Missions 1, 2, 6, 9, and 10). The makespan of the derived plans ranged from 686 to 1891, with a median of 1055. Grand Coalition 6 had the smallest average makespan, while Grand Coalition 2 had the largest. Mission 5 had the longest average makespan, while Mission 7 had the shortest. The number of action execution steps ranged from 459 steps to 663 steps, with a median of 529 steps, and is presented in Table 27. Mission 6 required the most steps (649.0), while Mission 10, with 467.0 steps, required the least. The time to derive a satisficing plan for each problem ranged from 279.7 to 936.4 s, with a median of 410.2 s. Full results are presented in Table 28. Mission 6 required the most time, at 805 s, while Mission 10 required the least, at 316 s. The

Table 25 The five number summary for solving the Zenotravel problems using CFP

Metric	Minimum	Lower Quartile	Median	Upper Quartile	Maximum
Makespan	686	900	1055	1291	1891
Steps	459	489	529	553	663
Time (s)	279.7	354.6	410.2	473.7	936.4
Memory (GB)	0.15	0.15	0.16	0.19	0.19

Table 26 Temporal makespan using CFP for the Zenotravel problems (Color table online)

		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	854	1277	937	696	966	882	1364	729	943	1076	972.4
	2	782	1445	872	781	1016	849	1158	686	889	1157	963.5
	3	1476	EHC	719	EHC	1190	900	1291	1027	1555	1128	1160.8
	4	1513	1139	1003	EHC	EHC	1252	1124	1145	1044	1340	1195.0
	5	EHC	1240	EHC	1304	1411	EHC	EHC	1305	1845	1510	1435.8
	6	960	1845	1429	993	1891	886	1299	1202	1565	1604	1367.4
	7	EHC	EHC	1000	905	918	EHC	1080	900	735	EHC	923.0
	8	1043	1372	EHC	764	1260	EHC	1368	EHC	1320	1057	1169.1
	9	736	1316	944	1129	904	804	1012	854	1055	1118	987.2
	10	849	1246	930	887	1187	820	870	959	941	1223	991.2
	Average	1026.6	1360.0	979.3	932.4	1193.7	913.3	1174.0	978.6	1189.2	1245.9	1099.3

The heatmap overlay ranges from 700 (green) to 1900 (red). A value of "EHC" indicates a failure due to using an incomple enforced hill climbing algorithm

Table 27 Number of action execution steps in the derived satisficing plan using CFP for the Zenotravel problems (Color table online)

		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	482	497	499	498	498	489	511	493	487	487	494.1
	2	494	496	480	472	479	474	488	469	475	476	480.3
	3	504	EHC	510	EHC	491	498	504	507	501	500	501.9
	4	529	541	535	EHC	EHC	533	544	536	532	541	536.4
	5	EHC	557	EHC	546	560	EHC	EHC	552	560	557	555.3
	6	642	649	659	647	651	645	639	663	647	648	649.0
	7	EHC	EHC	570	566	578	EHC	571	569	574	EHC	571.3
	8	538	547	EHC	552	551	EHC	555	EHC	553	525	545.9
	9	525	534	531	535	521	535	531	541	530	521	530.4
	10	472	463	460	460	475	478	469	471	463	459	467.0
	Average	523.3	535.5	530.5	534.5	533.8	521.7	534.7	533.4	532.2	523.8	530.5

The heatmap overlay ranges from 460 (green) to 660 (red). A value of "EHC" indicates a failure due to using an incomple enforced hill climbing algorithm

required memory results are presented in Table 29 and ranged from 0.15 to 0.19 GB, with a median of 0.16 GB.

Coalition formation was the least likely to generate nonexecutable coalitions for the Zenotravel domain. The domain and capability model was designed such that a candidate coalition will always be executable. The only instance in which a coalition will be nonexecutable occurs if planning for an earlier task transports a passenger to an airport inaccessible to the nonex-

Table 28 Planning time in seconds using CFP for the Zenotravel problems (Color table online)

		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	307	412	396	375	398	305	423	406	374	379	377.5
	2	370	366	381	310	321	360	373	310	320	320	343.0
	3	317	EHC	384	EHC	304	343	372	339	334	358	343.9
	4	366	447	435	EHC	EHC	355	413	441	417	445	414.8
	5	EHC	481	EHC	378	410	EHC	EHC	416	439	453	429.6
	6	677	833	728	781	897	901	728	936	880	692	805.1
	7	EHC	EHC	494	474	512	EHC	496	497	577	EHC	508.2
	8	449	533	EHC	524	461	EHC	505	EHC	485	408	480.6
	9	459	432	417	448	383	462	448	498	433	395	437.6
	10	339	312	280	285	320	366	349	322	296	290	315.9
	Average	410.5	476.9	439.4	446.8	444.9	441.6	456.2	462.9	455.5	415.6	445.3

The heatmap overlay ranges from 280 (green) to 940 s (red). A value of "EHC" indicates a failure due to using an incomple enforced hill climbing algorithm

Table 29 Required memory in gigabytes using CFP for the Zenotravel problems (Color table online)

		Grand Coalition									
		1	2	3	4	5	6	7	8	9	10
Mission	1	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19
	2	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15
	3	0.15	EHC	0.15	EHC	0.15	0.15	0.15	0.15	0.15	0.15
	4	0.16	0.16	0.16	EHC	EHC	0.16	0.16	0.16	0.16	0.16
	5	EHC	0.16	EHC	0.16	0.16	EHC	EHC	0.16	0.16	0.16
	6	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19
	7	EHC	EHC	0.18	0.18	0.18	EHC	0.18	0.18	0.18	EHC
	8	0.19	0.19	EHC	0.19	0.19	EHC	0.19	EHC	0.19	0.19
	9	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16
	10	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15

The heatmap overlay ranges from 0.15 (green) to 0.19 GB (red). A value of "EHC" indicates a failure due to using an incomple enforced hill climbing algorithm

ecutable coalition. The iterative initial state will have changed enough that the candidate coalition will no longer be executable. Assume the goal for v_i is to transport passenger $pass_i$ from $city_o$ to $city_i$ and the goal for v_j is to transport $pass_j$ from $city_o$ to $city_j$. Φ_j is assigned to v_j and can only travel between $city_o$ and $city_j$. If v_i is planned before v_j and the plan for v_i transports $pass_j$ to $city_i$, then Φ_j will be nonexecutable due to being unable to pickup $pass_j$ in $city_i$.

The Zenotravel domain has the simplest plan merge due to all tasks being independent. Two plans executed by disjoint coalitions can be executed concurrently. $plane_a$ transporting $cargo_i$ from $city_m$ to $city_n$ is independent of $plane_b$ transporting $cargo_j$ from $city_m$ to $city_n$. Plan merge must consider when the agent's prior action ended execution and if the passengers or cargo are prepared for loading or unloading. For example, a plane can begin refueling as soon as the plane arrives at an airport, but it will have to wait to continue to its destination if new passengers or cargo are not yet ready to board.

The average makespan for plans derived for Grand Coalition 2 was 1360, which is 0.93 standard deviations above the average makespan, while the average makespan for plans derived for Grand Coalition 6 was 913, 0.71 standard deviations below the average makespan. Grand Coalition 2 was composed of six long-range planes and twelve short-range planes,

Table 30 Coalition formation then planning summary

Domain	Problems solved	Summary
Blocks World	26	Fewer problems solved than with planning alone due to nonexecutable coalitions, but lower computation time and memory for most problems
Rovers	76	Fewer problems solved than with planning alone due to nonexecutable coalitions, but lower computation time for most problems and lower memory usage for all problems
Zenotravel	85	Most of the problems are solved with coalition formation then planning. All failures were due to using an incomplete underlying planning algorithm

while Grand Coalition 6 was composed of nine long-range planes and fourteen short-range planes. Having more planes of both types allowed Grand Coalition 6 to complete more tasks simultaneously. Mission 6 required an average of 649 action execution steps per derived plan. Mission 6 had nineteen tasks to be planned, with fifteen tasks requiring that the passengers and cargo be unloaded from one plane and loaded onto another plane. Mission 1 also had nineteen tasks, but required a below average number of action execution steps. Eleven of the Mission 1 tasks did not require passengers and cargo to change planes; therefore, fewer actions were required to produce a satisficing plan. The memory results are an effect of using the EHC version of COLIN. Focusing on specific agents and goals decreases the size of the heuristic plateaus that require breadth-first search.

6.2.4 Summary

CFP provided improved performance in terms of computational resource usage compared to PA on some problems, while introducing a new failure mode, nonexecutable coalitions. A summary of the results is presented in Table 30. The number of Blocks World problems solved by CFP is lower than the number of problems solved by PA, with 26 problems solved with CFP versus 42 solved with PA. However, most of the 74 failures are correctable. Nonexecutable coalition failures are correctable and caused 69 failures. Only five of the failures were due to exceeding the computational resource limits, which is not correctable by any of the presented tools. Only one Mission failed to generate a plan with any of the Grand Coalitions, compared to three Missions with no plans for PA. Fewer problems were solved for the Rovers domain by CFP compared to PA. CFP solved 76 problems, while PA solved 100. However, all the failures were due to nonexecutable coalitions, which is a correctable failure mode. Most of the Rovers problems were solved faster by CFP than PA. Furthermore, CFP had a required memory usage median of only 11% that required by PA. Most (85 out of 100) of the Zenotravel problems were solved using CFP, which is much better than the zero problems solved using PA. The fifteen failures were due to using a planning algorithm based on the incomplete Enforced Hill Climbing algorithm. Planning algorithm failures are not addressed by the presented tools, but using a complete underlying planning algorithm can allow the problems to be solved.

Algorithm 3: Coalition Formation then Planning with Relaxed Plan Coalition Augmentation

```
Input  : $S$ - state space, $A$ - coalition action mapping, $\Phi$ - grand coalition, $I$ - initial state, $V$ - tasks,
         $C$ - capability mappings
Output: $\pi$ - plan to satisfy all tasks
1  $\pi = \varnothing$;
2  $G = \varnothing$;
3  $\{Cap^{\phi}\}_{\phi \in \Phi} = C(\Phi)$;
4  $\{Cap^{v}\}_{v \in V} = C(V)$;
5  $\{\Phi_i, v_i\}_{i=1}^{|V|} = CF(\{Cap^{\phi}\}_{\phi \in \Phi}, \{Cap^{v}\}_{v \in V})$;
6  foreach $i \in \{1, \ldots, |V|\}$ do
7      $G = G \wedge v_i$;
8      $I_i = Simulate(I, \pi)$;
9      $Actions = A(\Phi_i)$;
10     $\pi_i = Plan(S, I_i, Actions, G)$;
11     while $\Phi_i$ is nonexecutable do
12         $AllActions = A(\Phi)$;
13         $\pi_r = RelaxedPlan(S, I_i, AllActions, G)$;
14         $\Phi_i = \Phi_i \cup RelaxedPlanCoalitionAugmentation(\Phi, \Phi_i, \pi_r)$;
15         $Actions = A(\Phi_i)$;
16         $\pi_i = Plan(S, I_i, Actions, G)$;
17     end
18     $\pi = PlanMerge(S, I, \pi, \pi_i, G)$;
19 end
```

CFP provides an alternative method of centralized planning for multi-agent systems providing reduced computational resource usage at the cost of being an incomplete algorithm. Problems in both the Rovers and Blocks World domains failed due to nonexecutable coalitions. The coalition formation algorithms used only consider task capability requirements and capabilities available to the coalitions; thus, the algorithms are reliant on the capability models being accurate representations of each coalition's action set and each task's requirements. Research in coalition formation includes spatial and temporal constraints on tasks. The more expressive coalition formation problem is more difficult, but provides more information as to whether or not a coalition can complete a task. The Rovers domain is an example of task spatial constraints. Considering the waypoints each rover can reach allows a coalition formation algorithm to eliminate rovers which cannot reach required waypoints; however, the more complex coalition formation formalisms are still estimates of the coalitions capable of completing a task. A plan must still be produced and coalitions can still be found to be nonexecutable. The relaxed plan coalition formation modification to CFP was developed to addresses this shortcoming to CFP.

6.3 Relaxed plan coalition augmentation

The Coalition Formation then Planning with Relaxed Plan Coalition Augmentation (RPCA) tool addresses the nonexecutable coalition limitation that arises with the CFP tool. RPCA adds logic to the planning loop of the Coalition Formation then Planning tool (see lines 11–17 in Algorithm 3). Planning for relaxed domains creates plans from low fidelity models of the real-world, but the problem is easier; thus, relaxed plans are appropriate for use as a heuristic or, in this case, coalition modification. Planning is attempted as in CFP, Line 10. If planning fails, then the algorithm enters a loop (lines 11–17). A nonexecutable coalition, Φ_i, is modified in order to make it executable. Coalitions can be modified by adding agents, removing agents, or a combination thereof; however, allowing any modifications introduces an exponential number of possible coalitions. The tool is limited to adding agents in order to transform a nonexecutable coalition into an executable coalition. The set of actions available to Φ, $AllActions$, is derived, Line 12. A relaxed plan to complete the task, π_r, is generated from $AllActions$, Line 13. The grand coalition, Φ, the currently allocated coalition, Φ_i, and the generated relaxed plan, π_r, are analyzed to select additional agent(s) to allocate to the task, Line 14. Each action execution step in the relaxed plan is analyzed in execution order. If the action in the step can be executed by an agent in Φ_i or a subcoalition of Φ_i, then analysis continues to the next action execution step, otherwise, the agent or coalition assigned to the action execution step is added to Φ_i and relaxed plan analysis stops. At least one agent must be added to Φ_i before planning is attempted again, so if all steps of the relaxed plan are analyzed and no agent has been added, then the agent, ϕ, executing the most action execution steps in π_r and $\phi \notin \Phi_i$ is added to Φ_i. Planning is attempted with the new coalition, Line 16, and the loop repeats if necessary.

The relaxed plan coalition formation augmentation loop (Lines 11–17) has two potential completions. Either an executable plan is derived using the newly generated coalitions and planning attempts or the grand coalition is allocated to the task and the algorithm is unable to identify a plan. The latter case is inconclusive, since the grand coalition can be nonexecutable for multiple reasons. The grand coalition may be nonexecutable due to previous planning decisions that make the task nonexecutable. An example can be created from a finite fuel modification to Zenotravel. The version of Zenotravel used in these experiments allows unlimited refueling, but if the amount of fuel available is limited, then prior planning decisions using excessive amounts of fuel can create situations in which no planes can reach their destination. The grand coalition may also be nonexecutable if the task cannot actually be executed, in which case the problem is unsolvable by any tool. An example of a nonexecutable grand coalition in the Blocks World domain is a grand coalition for which there are fewer than two arms with access to a required end effector type. If all the initially allocated coalitions are executable, then the loop starting at Line 11 is never entered; thus, RPCA reduces to CFP when all coalitions are executable. All Zenotravel coalitions were executable, so RPCA results are not reported for zenotravel.

6.3.1 Blocks World

Seventy of the one hundred problems were solved by RPCA within the time limit of 1 h, a large improvement over the 42 and 25 problems solved by planning alone and CFP, respectively. The five number summary for the solved problems are presented in Table 31. The quality of the derived plans are presented in Table 32. At least one satisficing plan was derived for all Grand Coalitions and all Missions. Plans were derived for all Grand Coalitions for Mission 9 and for all Missions for Grand Coalition 6. The number of actions in each plan is presented

Table 31 The five number summary for solving the Blocks World problems using RPCA

Metric	Minimum	Lower Quartile	Median	Upper Quartile	Maximum
Makespan	22	42	49	62	148
Steps	37	70	79	96	168
Time (s)	2.6	13.6	76.7	437.6	3111.9
Memory (GB)	0.01	0.09	0.35	6.40	30.80

Table 32 Temporal makespan using RPCA for the Blocks World problems (Color table online)

		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	50	TIME	MEM	55	56	47	65	49	57	58	54.6
	2	61	TIME	35	62	MEM	47	53	45	40	22	45.6
	3	TIME	TIME	65	74	TIME	148	73	72	70	94	85.1
	4	TIME	71	TIME	TIME	TIME	60	TIME	TIME	TIME	TIME	65.5
	5	TIME	73	39	56	42	61	45	46	57	TIME	52.4
	6	39	63	47	42	TIME	90	67	45	69	TIME	57.8
	7	42	TIME	51	46	61	47	38	41	42	64	48.0
	8	TIME	62	TIME	TIME	TIME	63	TIME	TIME	114	TIME	79.7
	9	45	34	47	41	59	38	58	47	56	45	47.0
	10	35	33	MEM	MEM	30	32	45	TIME	38	29	34.6
	Average	45.3	56.0	47.3	53.7	49.6	63.3	55.5	49.3	60.3	52.0	54.2

The heatmap overlay ranges from 20 (green) to 120 (red). A value of "MEM" or "TIME" indicates the memory or time limit, respectively, was exceeded while attempting to solve

Table 33 Number of action execution steps in the derived plans using RPCA for the Blocks World problems (Color table online)

		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	79	TIME	MEM	79	85	94	86	77	92	109	87.6
	2	96	TIME	64	89	MEM	76	77	79	62	43	73.3
	3	TIME	TIME	104	99	TIME	168	106	116	96	117	115.1
	4	TIME	120	TIME	TIME	TIME	76	TIME	TIME	TIME	TIME	98.0
	5	TIME	112	76	79	78	96	83	73	72	TIME	83.6
	6	79	95	83	63	TIME	136	108	84	95	TIME	92.9
	7	91	TIME	80	85	62	66	67	68	74	98	76.8
	8	TIME	99	TIME	TIME	TIME	102	TIME	TIME	149	TIME	116.7
	9	77	61	72	72	82	64	88	69	77	80	74.2
	10	52	50	MEM	MEM	37	53	53	TIME	59	46	50.0
	Average	79.0	89.5	79.8	80.9	68.8	93.1	83.5	80.9	86.2	82.2	83.3

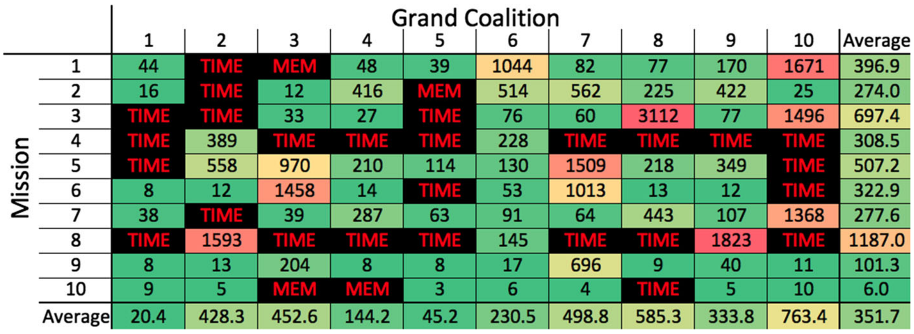
The heatmap overlay ranges from 40 (green) to 150 (red). A value of "MEM" or "TIME" indicates the memory or time limit, respectively, was exceeded while attempting to solve

in Table 33. The derived plans had a median of 70 action execution steps and ranged from 37 to 168 action execution steps. Time and memory requirements are presented in Tables 34 and 35, respectively. Time requirements ranged from 2.6 to 3111.9 s, with a median of 76.7 s. Memory usage had a median of 0.35 GB and ranged from 0.01 to 30.80 GB.

A set of 37 Blocks World problems were commonly solved by both PA and RPCA. The median metric ratios for the commonly solved problems are presented in Table 36. The solution quality metric ratios were 1.45 and 1.11 for makespan and steps, respectively. A ratio greater than 1 indicates that the plans derived by PA were better than the plans derived by RPCA. The makespan and action ratios were greater than 1 for 29 and 24 problems, respectively. However, the computational resource ratios were 0.73 and 0.70 for time and memory, respectively, indicating that RPCA derived plans while using less memory than PA. RPCA required less time and memory than PA for 25 and 24 problems, respectively.

The coalition formation failures in CFP are a result of the plans derived for earlier tasks and can be divided into two different cases. Assume a problem with the start state in Fig. 1a and the goal state description illustrated in Fig. 1b. Achieving stack C in the goal state requires

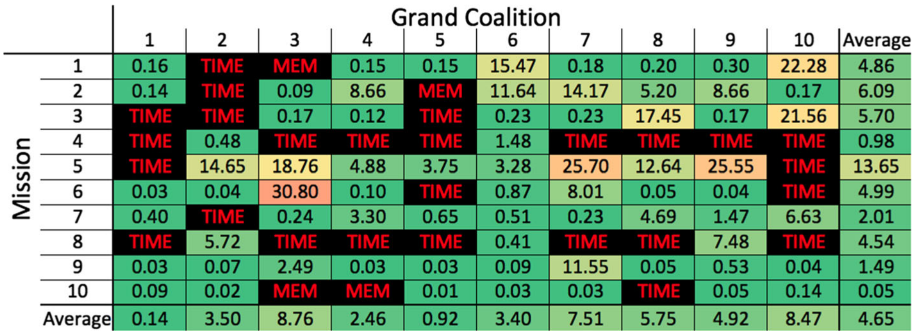
Table 34 Planning time in seconds using RPCA for the Blocks World problems (Color table online)



		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	44	TIME	MEM	48	39	1044	82	77	170	1671	396.9
	2	16	TIME	12	416	MEM	514	562	225	422	25	274.0
	3	TIME	TIME	33	27	TIME	76	60	3112	77	1496	697.4
	4	TIME	389	TIME	TIME	TIME	228	TIME	TIME	TIME	TIME	308.5
	5	TIME	558	970	210	114	130	1509	218	349	TIME	507.2
	6	8	12	1458	14	TIME	53	1013	13	12	TIME	322.9
	7	38	TIME	39	287	63	91	64	443	107	1368	277.6
	8	TIME	1593	TIME	TIME	TIME	145	TIME	TIME	1823	TIME	1187.0
	9	8	13	204	8	8	17	696	9	40	11	101.3
	10	9	5	MEM	MEM	3	6	4	TIME	5	10	6.0
	Average	20.4	428.3	452.6	144.2	45.2	230.5	498.8	585.3	333.8	763.4	351.7

The heatmap overlay ranges from 0 (green) to 1800 s (red). A value of "MEM" or "TIME" indicates the memory or time limit, respectively, was exceeded while attempting to solve

Table 35 Required memory in gigabytes using RPCA for the Blocks World problems (Color table online)



		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	0.16	TIME	MEM	0.15	0.15	15.47	0.18	0.20	0.30	22.28	4.86
	2	0.14	TIME	0.09	8.66	MEM	11.64	14.17	5.20	8.66	0.17	6.09
	3	TIME	TIME	0.17	0.12	TIME	0.23	0.23	17.45	0.17	21.56	5.70
	4	TIME	0.48	TIME	TIME	TIME	1.48	TIME	TIME	TIME	TIME	0.98
	5	TIME	14.65	18.76	4.88	3.75	3.28	25.70	12.64	25.55	TIME	13.65
	6	0.03	0.04	30.80	0.10	TIME	0.87	8.01	0.05	0.04	TIME	4.99
	7	0.40	TIME	0.24	3.30	0.65	0.51	0.23	4.69	1.47	6.63	2.01
	8	TIME	5.72	TIME	TIME	TIME	0.41	TIME	TIME	7.48	TIME	4.54
	9	0.03	0.07	2.49	0.03	0.03	0.09	11.55	0.05	0.53	0.04	1.49
	10	0.09	0.02	MEM	MEM	0.01	0.03	0.03	TIME	0.05	0.14	0.05
	Average	0.14	3.50	8.76	2.46	0.92	3.40	7.51	5.75	4.92	8.47	4.65

The heatmap overlay ranges from 0 (green) to 40 GB (red). A value of "MEM" or "TIME" indicates the memory or time limit, respectively, was exceeded while attempting to solve

Table 36 Median ratio of the RPCA metric value to the PA metric value for Blocks World problems

Problems	Makespan ratio	Steps ratio	Time ratio	Memory ratio
37	1.45	1.11	0.73	0.70

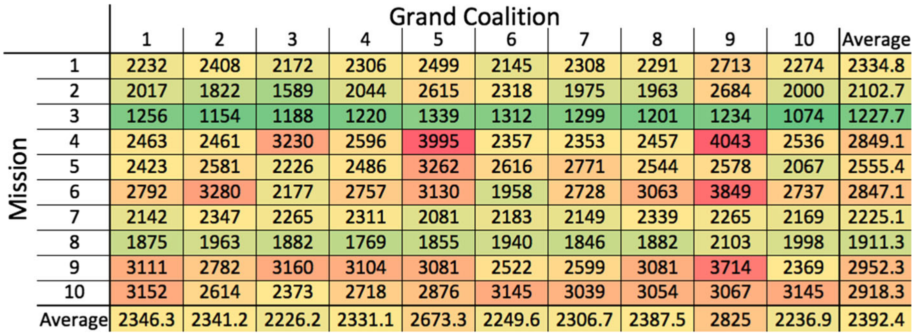
Only problems solved by both tools are considered

moving D and E in order to access C. The goal state to plan to achieve stack C does not place any constraints on D and E. The first coalition formation failure case occurs when an arm is still holding a block at the end of a task plan. Blocks D or E in this example can remain held at the end of the plan due to not being part of the goal description. The second coalition formation failure case occurs when a block is placed such that the blocks in the goal description are inaccessible. Block D placed on G is an example of the second failure case. The end effector required to manipulate D is not considered in the capabilities required vector for the stack E task. If the coalition for stack E cannot manipulate D, then the coalition will be nonexecutable.

Table 37 The five number summary for solving the Rovers domain problems using RPCA

Metric	Minimum	Lower Quartile	Median	Upper Quartile	Maximum
Makespan	1074	2013	2355	2742	4043
Steps	486	596	658	727	935
Time (s)	777.0	1032.3	1280.6	1526.3	2289.3
Memory (GB)	0.93	1.57	2.01	2.53	7.99

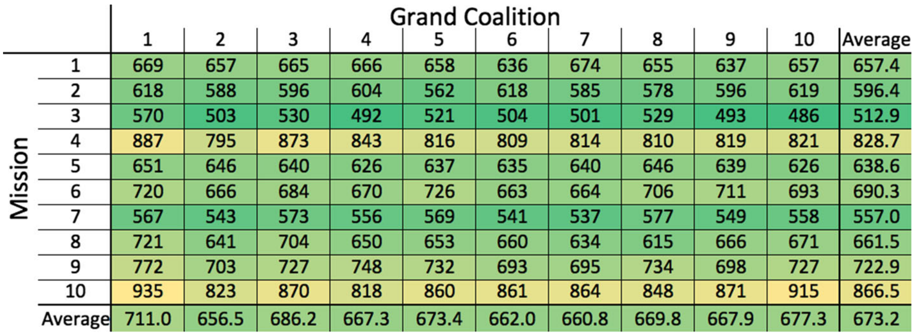
Table 38 Temporal makespan using RPCA for the Rovers problems (Color table online)



		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	2232	2408	2172	2306	2499	2145	2308	2291	2713	2274	2334.8
	2	2017	1822	1589	2044	2615	2318	1975	1963	2684	2000	2102.7
	3	1256	1154	1188	1220	1339	1312	1299	1201	1234	1074	1227.7
	4	2463	2461	3230	2596	3995	2357	2353	2457	4043	2536	2849.1
	5	2423	2581	2226	2486	3262	2616	2771	2544	2578	2067	2555.4
	6	2792	3280	2177	2757	3130	1958	2728	3063	3849	2737	2847.1
	7	2142	2347	2265	2311	2081	2183	2149	2339	2265	2169	2225.1
	8	1875	1963	1882	1769	1855	1940	1846	1882	2103	1998	1911.3
	9	3111	2782	3160	3104	3081	2522	2599	3081	3714	2369	2952.3
	10	3152	2614	2373	2718	2876	3145	3039	3054	3067	3145	2918.3
	Average	2346.3	2341.2	2226.2	2331.1	2673.3	2249.6	2306.7	2387.5	2825	2236.9	2392.4

The heatmap overlay ranges from 800 (green) to 3500 (red)

Table 39 Number of action execution steps in the derived plans using RPCA for Rovers problems (Color table online)



		Grand Coalition										
		1	2	3	4	5	6	7	8	9	10	Average
Mission	1	669	657	665	666	658	636	674	655	637	657	657.4
	2	618	588	596	604	562	618	585	578	596	619	596.4
	3	570	503	530	492	521	504	501	529	493	486	512.9
	4	887	795	873	843	816	809	814	810	819	821	828.7
	5	651	646	640	626	637	635	640	646	639	626	638.6
	6	720	666	684	670	726	663	664	706	711	693	690.3
	7	567	543	573	556	569	541	537	577	549	558	557.0
	8	721	641	704	650	653	660	634	615	666	671	661.5
	9	772	703	727	748	732	693	695	734	698	727	722.9
	10	935	823	870	818	860	861	864	848	871	915	866.5
	Average	711.0	656.5	686.2	667.3	673.4	662.0	660.8	669.8	667.9	677.3	673.2

The heatmap overlay ranges from 480 (green) to 1400 (red)

6.3.2 Rovers

RPCA solved all of the Rovers problems. The five number summary of the collected metrics is presented in Table 37. Each problem's solution makespan is presented in Table 38 and ranged from 1074 to 4043, with a median of 2355. The number of action execution steps in the derived plans is presented in Table 39 and had a median of 658 steps. Plan derivation time ranged from 777.0 to 2289.3 s, with a median of 1280.6 s. Full derivation time results are presented in Table 40 and the memory results are presented in Table 41. Required memory ranged from 0.93 to 7.99 GB.

Table 40 Planning time in seconds using RPCA for the Rovers problems (Color table online)

	Grand Coalition										
Mission	1	2	3	4	5	6	7	8	9	10	Average
1	1260	1242	1242	1217	1262	1254	1263	1239	1313	1283	1257.4
2	1157	852	908	1095	893	1200	1005	954	777	965	980.5
3	901	852	886	847	911	913	794	838	780	824	854.4
4	2130	1821	1770	2040	1731	1864	1878	1789	1758	2027	1880.9
5	1162	1405	1034	1302	1109	1488	1444	1342	1430	1223	1293.8
6	1399	1015	1110	1425	1526	1394	1351	1429	1265	1462	1337.5
7	903	997	1041	1028	996	1044	1004	982	1067	1023	1008.4
8	1439	1271	1401	1297	1302	1365	1207	1173	1266	1361	1308.2
9	1639	1476	1521	1441	1661	1564	1546	1524	1278	1501	1515.0
10	2289	1807	1895	1985	2011	2133	1991	1963	1911	2247	2023.2
Average	1427.9	1273.6	1280.8	1367.6	1340.2	1421.7	1348.3	1323.3	1284.3	1391.6	1345.9

The heatmap overlay ranges from 800 (green) to 3100 s (red)

Table 41 Required memory in gigabytes using RPCA for the Rovers problems (Color table online)

	Grand Coalition										
Mission	1	2	3	4	5	6	7	8	9	10	Average
1	1.40	1.17	1.37	1.87	2.09	2.49	1.63	1.92	2.37	2.02	1.83
2	1.57	2.52	1.76	1.32	1.76	2.81	1.85	1.11	1.75	1.71	1.82
3	1.01	1.22	1.18	1.29	1.18	1.67	1.19	1.05	1.19	1.33	1.23
4	2.19	1.80	1.91	2.01	2.03	2.66	2.56	2.04	2.00	3.05	2.23
5	1.77	2.97	2.16	2.01	2.35	3.08	2.64	1.58	1.89	3.42	2.39
6	1.90	3.69	2.27	1.54	2.38	3.06	3.27	2.41	2.40	3.31	2.62
7	0.93	1.18	1.53	1.40	1.70	2.84	1.51	1.67	1.95	1.72	1.64
8	1.26	1.59	1.31	1.35	1.46	2.02	1.75	1.56	2.03	2.11	1.64
9	2.14	1.88	1.76	2.65	2.46	3.31	2.46	2.57	3.09	3.07	2.54
10	3.03	2.82	7.99	2.41	2.22	4.58	3.35	2.30	2.67	4.36	3.57
Average	1.72	2.08	2.32	1.79	1.96	2.85	2.22	1.82	2.13	2.61	2.15

The heatmap overlay ranges from 0 (green) to 40 GB (red)

All 100 Rovers problems were solved by both PA and RPCA. The median metric ratios are presented in Table 42. The solution quality metric ratios were 1.38 and 0.87 for makespan and steps, respectively. PA produced lower makespan plans than RPCA for 85 of the problems, while produced plans with more actions than the corresponding RPCA plans for 86 of the problems. The computational resource ratios were 0.71 and 0.11 for time and memory, respectively, indicating that RPCA derived plans while using less memory than PA. PA required more time than RPCA for 95 of the problems, and more memory to solve all of the problems.

RPCA was able to solve the 24 problems that CFP failed to solve and in so doing solved every problem that planning alone was able to solve, but with a much lower computational resource requirement. The median memory usage and computation time for RPCA was 11% and 72% that of PA, respectively. The CFP failures were the result of rovers being unable to reach a necessary waypoint to collect scientific data. The generated relaxed plan provided information regarding which rovers were able to reach the required waypoints. A single rover was selected to augment the coalition and planning reattempted.

Table 42 Median ratio of the RPCA metric value to the PA metric value for Rovers problems

Problems	Makespan ratio	Steps ratio	Time ratio	Memory ratio
100	1.38	0.87	0.71	0.11

Only problems solved by both tools are considered

Table 43 Relaxed plan coalition augmentation summary

Domain	Problems solved	Summary
Blocks World	70	Solved more problems than PA and CFP, with lower average time and memory
Rovers	100	All problems were solved, with average computation time and memory usage well below that of PA
Zenotravel	85	Results are identical to CFP

6.3.3 Summary

The RPCA algorithm was able to correct all the nonexecutable coalition failures that occurred during the CFP evaluation. RPCA solved at least as many problems as CFP in each domain, as summarized in Table 43. The RPCA algorithm solved more Blocks World problems than PA and did so with lower average computation time and memory. RPCA solved the same Blocks World problems as CFP in addition to others, but PA solved some problems that RPCA did not. RPCA solved all the Rovers problems, requiring median lower memory and computation time compared to PA. All of the nonexecutable coalition failures from CFP were fixed by applying RPCA. No nonexecutable coalition failures occurred when CFP was applied to the the Zenotravel problems; thus, the RPCA results are identical to the CFP results, because RPCA only addresses CFP's nonexecutable coalition failure mode.

The ability to correct for nonexecutable coalitions is imperative when generating plans for real-world problems. Generating a relaxed plan is easier than the original planning problem, so the grand coalition can be used in relaxed planning. Using the grand coalition during the relaxed planning step generates additional information unavailable during coalition formation. An agent, previously unassigned to the coalition, is selected to augment a nonexecutable coalition and transform it into an executable coalition. Adding agents to a nonexecutable coalition does not guarantee that the coalition will become executable, but it does provide additional actions that can be considered during planning to produce an executable plan for the coalition's assigned task.

6.4 Task fusion

The Coalition Formation followed by Task Fusion then Planning (TF) tool addresses the limited ability of CFP to reason over task interactions. Coalition formation is applied as in CFP, Line 5 in Algorithm 4. Task fusion reasons over the tasks and their capabilities required and the coalitions and their capabilities offered to guess which tasks and coalitions are most likely to interact and benefit from joint planning, Line 6. Two coalition-task pairs, $\langle \Phi_a, v_a \rangle$ and $\langle \Phi_b, v_b \rangle$, selected to be fused create a single coalition-task pair, $\langle \Phi_a \cup \Phi_b, v_a \wedge v_b \rangle$.

Algorithm 4: Coalition Formation followed by Task Fusion then Planning with Relaxed Plan Coalition Augmentation

```
Input : $S$ - state space, $A$ - coalition action mapping, $\Phi$ - grand coalition, $I$ - initial state, $V$ - tasks,
        $C$ - capability mappings
Output: $\pi$ - plan to satisfy all tasks
1  $\pi = \emptyset$;
2  $G = \emptyset$;
3  $\{Cap^{\phi}\}_{\phi \in \Phi} = C(\Phi)$;
4  $\{Cap^{v}\}_{v \in V} = C(V)$;
5  $\{\Phi_i, v_i\}_{i=1}^{|V|} = CF(\{Cap^{\phi}\}_{\phi \in \Phi}, \{Cap^{v}\}_{v \in V})$;
6  $\{\Phi_j, v_j\}_{j=1}^{|V_f|} = TaskFusion(\{\Phi_i, v_i\}_{i=1}^{|V|})$;
7  foreach $j \in \{1, \ldots, |V_f|\}$ do
8      $G = G \wedge v_j$;
9      $I_j = Simulate(I, \pi)$;
10     $Actions = A(\Phi_j)$;
11     $\pi_j = Plan(S, I_j, Actions, G)$;
12     while $\Phi_j$ is nonexecutable do
13         $AllActions = A(\Phi)$;
14         $\pi_r = RelaxedPlan(S, I_j, AllActions, G)$;
15         $\Phi_j = \Phi_j \cup RelaxedPlanCoalitionAugmentation(\Phi, \Phi_j, \pi_r)$;
16         $Actions = A(\Phi_j)$;
17         $\pi_j = Plan(S, I_j, Actions, G)$;
18     end
19     $\pi = PlanMerge(S, I, \pi, \pi_j, G)$;
20 end
```

Fusing tasks into a single task permits more potential interactions between the tasks to be considered during the planning step. The coalition capability offerings and the task capability requirements will be used to determine which tasks to fuse. Assume two coalitions, Φ_i and Φ_j, are allocated to two tasks, v_i and v_j, respectively. If Φ_i is also a candidate coalition for v_j and Φ_j is also a candidate coalition for v_i, then the coalitions can assist each other; fusing the coalition-task allocations allows both coalitions to contribute to both tasks during the planning step. Each coalition-task allocation fusion decreases the number of subproblems by one, but the difficulty of planning for the two tasks is much lower than planning for the fused subproblem due to the exponential complexity of the planning problem. The goal of Task Fusion is to select the coalition-task pairs to fuse for which improved plan quality is worth the increase in computational resources.

TF applied to the Blocks World domain allows planning to intentionally make progress towards constructing multiple stacks. A block can be placed intentionally where it belongs in the goal state. TF can be applied to the other experimental domains as well. Fusing tasks in the Rovers domain allows rovers to work together. Assume the rock and soil tasks have been fused. If a rover is capable of both analyses and is at a waypoint requiring both analyses, then the rover can perform both rock and soil analysis. Zenotravel task fusion allows more of each plane's capacity to be used. The CFP algorithm only plans for one task at a time; thus, the current task's passengers and cargo must be transported to their destination before passengers and cargo for subsequent tasks can be considered. Fusing tasks involving transport between the same hubs will allow more of each plane's capacity to be used.

TF allows mission planners to tune computational resource usage and plan quality. Analyzing TF is most logical in mission planning problems for which plan quality is measured by plan makespan due to the tradeoff between planning time and plan quality introduced by the tool. The time to derive a solution can be as important as the quality of the derived solution. Solving problems in domains, such as planetary rovers, can take longer to plan, because the time to execute the plan is likely to dominate the planning plus execution time equation due to the large areas being explored. If the makespan unit in the example Blocks World problem is 30 s, then TF minimizes planning time plus execution time. If the makespan unit is 1 s, then CFP minimizes planning plus execution time. If the makespan unit is 1 min, then planning alone minimizes planning plus execution time. TF can be applied to decrease plan makespan at the cost of increased memory usage and planning time. However, TF must be applied intelligently, as increasing the planning time by minutes is illogical if the plan makespan is only decreased by seconds. TF represents an intermediate option between CFP and planning alone. Fusing selected coalition-task pairs can produce better plans than CFP, while still satisfying computational resource constraints.

6.5 Summary

The first tool evaluated, PA, is an existing method for solving multi-agent planning problems containing expressive state models. The other two tools evaluated in this research, CFP and RPCA, present an alternative for multi-agent planning addressing the problem's computational complexity. Three domains were used to evaluate the tools, Blocks World, Rovers, and Zenotravel. Each tool was evaluated using 100 randomly generated problems from each domain, with the number of problems for which plans were derived presented in Fig. 3. RPCA derived plans for at least as many problems as PA in each domain. Every problem for which CFP derived a plan, an identical plan was derived by RPCA. Furthermore, every nonexecutable coalition failure mode that occurred during the CFP evaluation was corrected by RPCA; thus, RPCA dominates CFP and CFP will not be further analyzed. Statistical analysis of the PA and RPCA is complicated by the computation resource limits applied during the evaluation. The limits are realistic in that real-world applications of planning research must consider the time and memory required to solve a problem, but the limits cause the collected data to be right-censored, i.e., no data is collected for problems requiring time above the limit or memory above the limit, even though problems capable of generating such data are included in the evaluation. Missing data limits the data points that can be used in statistical significance testing and when determining effect size. All the problems attempted during the

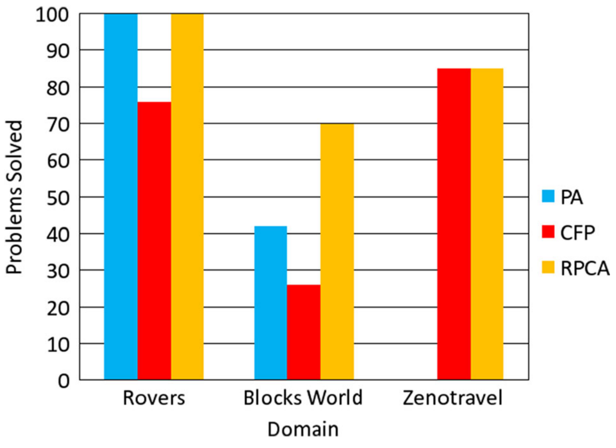


Fig. 3 Number of problems in each domain with derived plans by algorithm

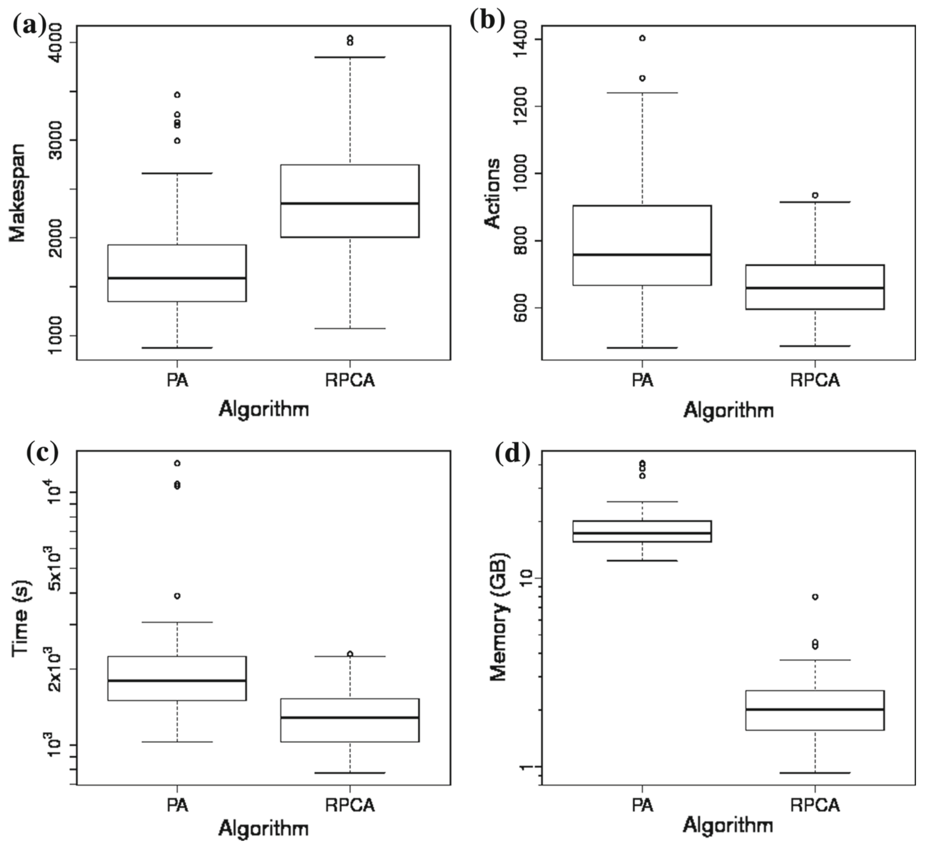


Fig. 4 Metrics *box plots* for Rovers problems solved with PA and RPCA. **a** Makespan, **b** steps, **c** computation Time and **d** required memory

evaluation are solvable, but data is lacking for many, especially for the PA algorithm. Given the limits of the data collection, statistical analysis using limited assumptions regarding the data was conducted.

Both PA and RPCA solved Blocks World problems that the other algorithm did not, with PA solving 42 problems and RPCA solving 70 problems. A McNemar's test found a significant difference in the proportion of problems for which plans were derived by each tool ($a = 37, b = 5, c = 33, d = 25, p < 0.001$). The set of 37 problems commonly solved by both PA and RPCA was used for analyzing the four metrics using the sign test. Significant differences between the PA and RPCA algorithms were found for makespan ($Z = 3.66$, $p < 0.001$) and computation time ($Z = 2.33$, $p = 0.028$), but there was not enough evidence to reject the significance hypothesis for differences in required memory ($Z = 1.81$, $p = 0.099$) or number of actions ($Z = 1.86$, $p = 0.090$). Given that all 58 of the PA failures were due to exceeding memory limits, it is predicted that increasing the memory limit will produce additional data points that will demonstrate significant differences for required memory between PA and RPCA in the Blocks World domain.

All the Rovers problems were solved by both PA and RPCA. The box plots for each metric are presented in Fig. 4, please note that the y-axis for the time and memory box plots are on a log scale. The sign test found significant differences between PA and RPCA for all four metrics ($Z = 7.00$, $p < 0.001$ for makespan, $Z = 7.20$, $p < 0.001$ for action steps, $Z = 9.00$, $p < 0.001$ for computation time, and $Z = 10.00$, $p < 0.001$ for required memory), with PA deriving plans with lower makespan, but RPCA requiring less time and memory to derive plans with fewer action executions.

PA derived no plans for the Zenotravel domain; therefore, it is infeasible to provide an analysis of the results compared to RPCA. RPCA derived plans for 85 of the problems.

7 Conclusion

Multi-agent systems will only become more complex and robots will need to plan and act autonomously alongside humans, not simply be controlled by humans. An important aspect of mission planning is the tradeoff between mission planning time and plan quality. If unlimited computational resources are available, then existing planning algorithms alone are able to solve large, complex mission planning instances for high fidelity real-world domain models. However, computational resource constraints and real-world time constraints encourage development of new tools to support real-time planning for dynamic, uncertain real-world domains. The presented tools manage the planning problem's exponential complexity. The Coalition Formation then Planning tool uses existing coalition formation algorithms to factor the problem prior to applying existing planning algorithms in order to generate executable plans. Two issues arise with the Coalition Formation then Planning tool: nonexecutable coalitions and suboptimal plans. The Relaxed Plan Coalition Formation tool addresses the nonexecutable coalition issue by using relaxed planning to select agents to insert into coalitions in order to transition nonexecutable coalitions into executable coalitions. Task Fusion addresses the suboptimal plans issue by fusing coalition-task pairs for which the increase in plan quality from planning simultaneously offsets the increase in computational resource requirements. Deciding whether or not to apply Task Fusion is mission dependent. If plan quality is judged by makespan, then makespan must be reduced by an amount of time greater than or equal to the additional time required to derive the plan. These tools work together to generate executable plans for large, complex mission planning problems with high fidelity domain models that include continuous variables and concurrently executing actions.

Future work involves integrating the Task Fusion augmentation into the existing framework for coalition formation then planning and analyzing different heuristics for selecting tasks for fusion. A test domain modeling a first response scenario and associated problems will be evaluated with each of the presented tools. Finally, the mission planning system will be integrated with the i-CiFHaR coalition formation system in order to create a complete system for real-world mission deployments.

References

1. Allouche, M. K., & Boukhtouta, A. (2010). Multi-agent coordination by temporal plan fusion: Application to combat search and rescue. *Information Fusion*, *11*(3), 220–232.
2. Barrientos, A., Colorado, J., Cerro, Jd, Martinez, A., Rossi, C., Sanz, D., et al. (2011). Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. *Journal of Field Robotics*, *28*(5), 667–689.
3. Bibaï, J., Savéant, P., Schoenauer, M., & Vidal, V. (2010). On the benefit of sub-optimality within the divide-and-evolve scheme. In P. Cowling & P. Merz (Eds.), *Evolutionary computation in combinatorial optimization* (Vol. 6022, pp. 23–34)., Lecture notes in computer science Berlin: Springer.
4. Brafman, R. I., & Domshlak, C. (2008). From one to many: Planning for loosely coupled multi-agent systems. In *Proceedings of the 18th international conference on autonomous planning and scheduling* (pp. 28–35).

5. Bryce, D., Gao, S., Musliner, D., & Goldman, R. (2015). SMT-based nonlinear PDDL+ planning. In *Proceedings of the 29th conference on artificial intelligence* (pp. 3247–3253).
6. Chaimowicz, L., Cowley, A., Sabella, V., & Taylor, C. (2003). ROCI: A distributed framework for multi-robot perception and control. In *Proceedings of the 2003 IEEE/RSJ international conference on intelligent robots and systems* (Vol. 1, pp. 266–271).
7. Chen, Y., Wah, B. W., & Hsu, C. W. (2006). Temporal planning using subgoal partitioning and resolution in SGPlan. *Journal of Artificial Intelligence Research*, *26*(1), 323–369.
8. Coles, A., Coles, A., Fox, M., & Long, D. (2012). COLIN: Planning with continuous linear numeric change. *Journal of Artificial Intelligence Research*, *44*(1), 1–96.
9. Cox, J. S., & Durfee, E. H. (2005). An efficient algorithm for multiagent plan coordination. In *Proceedings of the 4th international joint conference on autonomous agents and multiagent systems* (pp. 828–835).
10. Dias, M., Ghanem, B., & Stentz, A. (2005). Improving cost estimation in market-based coordination of a distributed sensing task. In *Proceedings of the 2005 IEEE/RSJ international conference on intelligent robots and systems* (pp. 3972–3977).
11. Dimopoulos, Y., Hashmi, M. A., & Moraitis, P. (2012). SATPLAN: Multi-agent planning as satisfiability. *Knowledge-Based Systems*, *29*, 54–62.
12. Do, M., & Kambhampati, S. (2001). Sapa: A domain-independent heuristic metric temporal planner. In *Proceedings of the 6th European conference on planning* (pp. 57–68).
13. Engesse,r T., Bolander, T., Mattmüller, R., & Nebel, B. (2015). Cooperative epistemic multi-agent planning with implicit coordination. In *Proceedings of the 3rd workshop on distributed and multi-agent planning* (pp. 68 – 76).
14. Ephrati, E., & Rosenschein, J. S. (1993). Multi-agent planning as the process of merging distributed sub-plans. In *Proceedings of the 12th international workshop on distributed artificial intelligence* (pp. 115–129).
15. Ephrati, E., & Rosenschein, J. S. (1994). Divide and conquer in multi-agent planning. In *Proceedings of the 12th national conference on artificial intelligence* (Vol. 1, pp. 375–380).
16. Erol, K., Nau, D. S., & Subrahmanian, V. S. (1992). On the complexity of domain-independent planning. In *Proceedings of the 10th national conference on artificial intelligence* (pp. 381–386).
17. Eyerich, P., Keller, T., Aldinger, J., & Dornhege, C. (2014). Preferring preferred operators in temporal fast downward. In *International planning competition* (pp. 121–126).
18. Fikes, R. E., & Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*(34), 189–208.
19. Fox, M., & Long, D. (2003). PDDL 2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, *20*(1), 61–124.
20. Gerkey, B. P., & Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, *23*(9), 939–954.
21. Hoffmann, J. (2003). The metric-ff planning system: Translating ”ignoring delete lists” to numeric state variables. *Journal of Artificial Intelligence Research*, *20*(1), 291–341.
22. Howey, R., Long, D., & Fox, M. (2004). VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL. In *IEEE international conference on tools with artificial intelligence* (pp. 294–301).
23. Hsu, C. W., Wah, B. W., Huang, R., & Chen, Y. (2007). Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In *Proceedings of the 20th international joint conference on artifical intelligence* (pp. 1924–1929).
24. Kim, M. H., Baik, H., & Lee, S. (2014). Response threshold model based uav search planning and task allocation. *Journal of Intelligent and Robotic Systems*, *75*(3–4), 625–640.
25. Koes, M., Nourbakhsh, I., & Sycara, K. (2005). Heterogeneous multirobot coordination with spatial and temporal constraints. In *Proceedings of the 20th national conference on artificial intelligence* (Vol. 3, pp. 1292–1297).
26. Laborie, P., & Ghallab, M. (1995). IxTeT: An integrated approach for plan generation and scheduling. In *Proceedings of the 1995 INRIA/IEEE symposium on emerging technologies and factory automation* (Vol. 1, pp. 485–495).
27. Liemhetcharat, S., & Veloso, M. (2014). Weighted synergy graphs for effective team formation with heterogeneous ad hoc agents. *Artificial Intelligence*, *208*, 41–65.
28. Long, D., & Fox, M. (2003). The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research*, *20*(1), 1–59.
29. Moon, S., Oh, E., & Shim, D. (2013). An integral framework of task assignment and path planning for multiple unmanned aerial vehicles in dynamic environments. *Journal of Intelligent and Robotic Systems*, *70*, 303–313.

30. Muise, C., Belle, V., Felli, P., McIlraith, S., Miller, T., Pearce, A. R., & Sonenberg, L. (2015). Planning over multi-agent epistemic states: A classical planning approach. In *Proceedings of the 29th AAAI conference on artificial intelligence* (pp. 3327–3334).
31. Penberthy, J. S., & Weld, D. S. (1994). Temporal planning with continuous change. In *Proceedings of the 1994 AAAI conference on artificial intelligence* (pp. 1010–1015).
32. Ponda, S., Johnson, L., & How, J. (2012). Distributed chance-constrained task allocation for autonomous multi-agent teams. In *American control conference* (pp. 4528–4533).
33. Rahwan, T., Ramchurn, S. D., Jennings, N. R., & Giovannucci, A. (2009). An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research*, *34*, 521–567.
34. Ramchurn, S. D., Polukarov, M., Farinelli, A., Truong, C., & Jennings, N. R.(2010). Coalition formation with spatial and temporal constraints. In *Proceedings of the 9th international conference on autonomous agents and multiagent systems* (Vol. 3, pp. 1181–1188).
35. Rankooh, M. F., & Ghassem-Sani, G. (2015). ITSAT: An efficient sat-based temporal planner. *Journal of Artificial Intelligence Research*, *53*, 541–632.
36. Ren, Z., Feng, Z., & Wang, X. (2008). An efficient ant colony optimization approach to agent coalition formation problem. In *Proceedings of the 7th world congress on intelligent control and automation* (pp. 7879–7882).
37. Sandholm, T., Larson, K., Andersson, M., Shehory, O., & Tohmé, F. (1999). Coalition structure generation with worst case guarantees. *Artificial Intelligence*, *111*(1–2), 209–238.
38. Sen, S. D. (2015). An intelligent and unified framework for multiple robot and human coalition formation. Ph.D. thesis, Vanderbilt University.
39. Sen, S. D., & Adams, J. A. (2014). An influence diagram based multi-criteria decision making framework for multirobot coalition formation. *Autonomous Agents and Multi-Agent Systems*, *29*, 1061–1090.
40. Service, T. C., & Adams, J. A. (2011). Coalition formation for task allocation: Theory and algorithms. *Journal of Autonomous Agents and Multi-Agent Systems*, *22*(2), 225–248.
41. Service, T. C., Sen, S. D., & Adams, J. A. (2014). A simultaneous descending auction for task allocation. In *Proceedings of the 2014 IEEE international conference on systems, man and cybernetics, IEEE* (pp. 379–384).
42. Shehory, O., & Kraus, S. (1998). Methods for task allocation via agent coalition formation. *Artificial Intelligence*, *101*, 165–200.
43. Shiroma, P. M., & Campos, M. F. M. (2009). Comutar: A framework for multi-robot coordination and task allocation. In *2009 IEEE/RSJ international conference on intelligent robots and systems* (pp. 4817–4824).
44. Sujit, P., Manathara, J., Ghose, D., & de Sousa, J. (2014). Decentralized multi-uav coalition formation with limited communication ranges. In K. P. Valavanis & G. J. Vachtsevanos (Eds.), *Handbook of unmanned aerial vehicles* (pp. 2021–2048). Dordrecht: Springer.
45. Tang, F., & Parker, L. E. (2005). ASyMTRe: Automated synthesis of multi-robot task solutions through software reconfiguration. In *Proceedings of IEEE international conference on robotics and automation* (pp. 1513–1520).
46. Torreño, A., Onaindía, E., & Sapena, O. (2012). An approach to multi-agent planning with incomplete information. *European Conference on Artificial Intelligence*, *242*, 762–767.
47. Turpin, M., Michael, N., & Kumar, V. (2013). Trajectory planning and assignment in multirobot systems. In E. Frazzoli, T. Lozano-Perez, N. Roy, & D. Rus (Eds.), *Algorithmic foundations of robotics X, Springer tracts in advanced robotics* (Vol. 86, pp. 175–190). Berlin: Springer.
48. Turpin, M., Michael, N., & Kumar, V. (2014). CAPT: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*, *33*(1), 98–112.
49. Vidal, V. (2004). The yahsp planning system: Forward heuristic search with lookahead plans analysis. In *4th international planning competition, Citeseer* (pp. 56–58).
50. Vig, L., & Adams, J. A. (2006a). Market-based multi-robot coalition formation. In *Proceedings of the 8th international symposium on distributed autonomous robotic systems* (pp. 227–236).
51. Vig, L., & Adams, J. A. (2006b). Multi-robot coalition formation. *IEEE Transactions on Robotics*, *22*(4), 637–649.
52. de Weerdt, M., Bos, A., Tonino, H., & Witteveen, C. (2003). A resource logic for multi-agent plan merging. *Annals of Mathematics and Artificial Intelligence*, *37*(1–2), 93–130.
53. Wicke, D., Freelan, D., & Luke, S. (2015). Bounty hunters and multiagent task allocation. In *Proceedings of the 2015 international conference on autonomous agents and multiagent systems* (pp. 387–394).
54. Zhu, D., Huang, H., & Yang, S. (2013). Dynamic task assignment and path planning of multi-auv system based on an improved self-organizing map and velocity synthesis method in three-dimensional underwater workspace. *IEEE Transactions on Cybernetics*, *43*(2), 504–514.