Greedy Approaches for Solving Task-Allocation Problems with Coalitions

Xiaoming Zheng Sven Koenig
University of Southern California
Computer Science Department
Los Angles, California, USA
{xiaominz, skoenig}@usc.edu

ABSTRACT

We study task-allocation problems where one wants to minimize the team cost subject to the constraint that each task must be executed by a given number of cooperative agents simultaneously, called the coalition for this task. We use the same principle to develop several efficient and effective (greedy) hillclimbing strategies for determining both which agents belong to the coalition for each task and when the coalition should execute the task. We analyze their complexities and compare them experimentally with each other and a previous algorithm on multi-agent routing problems.

1. INTRODUCTION

We study task-allocation problems where one wants to minimize the team cost subject to the constraint that each task must be executed by a given number of cooperative agents simultaneously, called the coalition for this task. Each agent can participate in coalitions for different tasks. Each task requires a number of different capabilities, and each coalition for the task needs to provide the required capabilities. We use the same principle to develop several efficient and effective (greedy) hillclimbing strategies for determining both which agents belong to the coalition for each task (= assigning tasks, sometimes also called coalition formation) and when the coalition should execute the task (= scheduling tasks). We analyze the complexities of our strategies and compare them experimentally with each other and a previous algorithm on multi-agent routing problems.

2. RELATED WORK

Task-allocation problems are typically studied in a competitive setting, often in the context of strategies from game theory [2, 7] or negotiation protocols for coalition formation [5]. We, on the other hand, study task-allocation problems in a cooperative setting. Lau and Zhang classified coalition formation problems for these kinds of task-allocation problems and explored the runtime complexity of each category [6]. Shehory and Kraus developed several iterative distributed greedy algorithms for coalition formation [8, 9]. Yong et al. studied a similar problem in the context of grid computing [11]. Abdallah and Lesser explored how organization structures of agents can guide the coalition formation process and used reinforcement learning to optimize the decisions of each agent [1]. Soh and Li used multilevel learning for coalition formation, mostly without considering the team cost [10]. Most of this work suffers from two limitations that we relax in this paper, namely that each agent can participate in a coalition for at most one task [1, 8, 11] and that the contribution of a given coalition to the team cost is either pre-defined [1, 6] or can be calculated quickly [8, 9, 11]. We, on the other hand, study task-allocation problems where each agent can participate in coalitions for different tasks and needs to solve a complex scheduling problem to calculate the contribution of a given coalition to the team cost. The most closely related existing work is our work on ARF (= Approach with Reaction Functions), that allocates one unallocated task per round until all tasks have been allocated [12]. The agents submit so-called reaction functions for each unallocated task, and the central planner basically allocates one unallocated task per round to one qualified coalition next so that the team cost increases the least.

AAMAS 2008 Workshop on Formal Models and Methods for Multi-Robot Systems, May 13 2008, Estoril, Portugal.

3. MULTI-AGENT ROUTING

Our motivating problems are multi-agent routing problems where agents have to visit targets in the plane. The terrain and the locations of the agent and targets are known. An example is fire fighting, where several fires need to get fought (both simultaneously and one after the other) so that the fires are extinguished as quickly as possible. Multi-agent routing problems where each target needs to get visited by only one agent are standard test domains in robotics [3]. Here, however, we extend them to the case where some targets need to get visited simultaneously by several agents. An example is again fire fighting, where large fires can be extinguished only by several fire engines simultaneously. We formalize multi-agent routing problems as follows: The finite set of targets is $X = \{x_1, \ldots, x_m\}$. The finite set of agents is $A = \{a_1, \ldots, a_n\}$. The finite set of capabilities is $B = \{b_1, \ldots, b_r\}$. Each agent $a \in A$ is characterized by a vector of capabilities $B^a = \langle b_1^a, \ldots, b_r^a \rangle$ that it provides, where each b_i^a is a non-negative integer. Each target $x \in X$ is characterized by a vector of capabilities $B^x = \langle b_1^x, \ldots, b_r^x \rangle$ that it requires, where each b_i^x is a non-negative integer. A coalition $O \subseteq A$ of agents is qualified for target $x \in X$ iff it satisfies $\sum_{a \in O} B^a \geq B^x$. All agents in the coalition need to visit the target at the same visit time t. An allocation of agent $a \in A$ consists of the pair (X_a, C_a), where $X_a \subseteq X$ is the set of targets assigned to it and C_a is the set of its visit times for the targets in X_a, in the form $x \leftarrow t$. The agent cost $c_a^{agent}(X_a, C_a)$ is the largest visit time of all targets in X_a if the agent can visit all targets in X_a at the visit times in C_a (and infinity if it cannot). We want to assign and schedule targets so that the team cost is small. We consider two different ways of defining the team cost. The team cost is $\sum_{a \in A} c_a^{agent}(X_a, C_a)$ for the MiniSum team objective (which is roughly proportional to the energy needed by all agents for waiting and moving) and $\max_{a \in A} c_a^{agent}(X_a, C_a)$ for the MiniMax team objective (which is equal to the task-completion time).

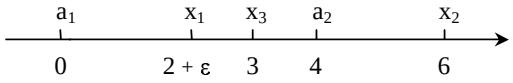

Figure 1: Multi-Agent Routing Problem on the Real Line

Function AssignTarget(*STRATEGY*, $\{X_a\}_{a\in A}$, k)
Input:
STRATEGY: the strategy for assigning targets;
$\{X_a\}_{a\in A}$: the current partial assignments;
k : the parameter for the simplified systematic strategy.
Output:
$\{(X_a, C_a)\}_{a\in A}$: the allocations;
lowestevaluation: the team cost of the allocations.
{01} **While** ($|X \setminus \cup_{a\in A} X_a| > k$)
{02} *lowestevaluation* $:= \infty$;
{03} **Foreach** $x \in X \setminus \cup_{a\in A} X_a$
{04} **Foreach** $O \subseteq A$ *qualified for* x
{05} **Foreach** $a \in A$
{06} **If** $a \in O$
{07} $X'_a := X_a \cup \{x\}$;
{08} **Else**
{09} $X'_a := X_a$;
{10} **If** *STRATEGY = GREEDY*
{11} $(\{(X_a, C_a)\}_{a\in A}$, *evaluation*$) :=$ *ScheduleTargets*$(\{X'_a\}_{a\in A})$;
{12} **Else**
{13} $(\{(X''_a, C''_a)\}_{a\in A}$, *evaluation*$) :=$
AssignTarget(*GREEDY*, $\{X'_a\}_{a\in A}, 0)$;
{14} **If** *evaluation* $<$ *lowestevaluation*
{15} *lowestevaluation* $:=$ *evaluation*;
{16} $x^* := x$;
{17} $O^* := O$;
{18} **Foreach** $a \in A$
{19} **If** $a \in O^*$
{20} $X_a := X_a \cup \{x^*\}$;
{21} **If** ($|X \setminus \cup_{a\in A} X_a| > 0$)
{22} $(\{(X_a, C_a)\}_{a\in A}$, *lowestevaluation*$) :=$
AssignTarget(*GREEDY*, $\{X_a\}_{a\in A}, 0)$;
{23} **Else**
{24} $(\{(X_a, C_a)\}_{a\in A}$, *lowestevaluation*$) :=$ *ScheduleTargets*$(\{X_a\}_{a\in A})$;
{25} **Return** $(\{(X_a, C_a)\}_{a\in A}$, *lowestevaluation*$)$;

Figure 2: Pseudo Code for Assigning Targets

4. EXAMPLE PROBLEM

Consider the multi-agent routing problem in Figure 1, which we use throughout this paper. The agents and targets are located on the real line. There exists only one capability. Agents a_1 and a_2 provide one unit of this capability each. Targets x_1 and x_2 require one unit of this capability each, while target x_3 requires two units. In other words, targets x_1 and x_2 need to be visited by one agent each, while target x_3 needs to be visited by both agents simultaneously. The agents can move one unit of distance in one unit of time. The complete assignments with the minimal team cost for the MiniMax team objective are as follows: Agent a_1 first visits target x_1 at time $2 + \epsilon$ and then target x_3 at time 5. Agent a_2 first visits target x_2 at time 2 and then target x_3 at time 5. Agent a_1 can reach target x_3 already at time 3 but then has to wait there until agent a_2 arrives at time 5. Thus, the minimal team cost is 5.

5. ASSIGNING TARGETS

Our strategies assign one unassigned target per round until all targets have been assigned, as shown in Figure 2. Assigning one unassigned target per round keeps the runtime small, yet allows them to take into account the positive and negative synergies between the targets that still need to be assigned and the targets that have already been assigned. Consider an arbitrary round. Let the partial assignments of targets at the end of the previous round be $\{X_a\}_{a\in A}$. The assignments $\{X_a\}_{a\in A}$ are called complete iff $X = \cup_{a\in A} X_a$. Otherwise, they are called partial.

5.1 Greedy Strategy

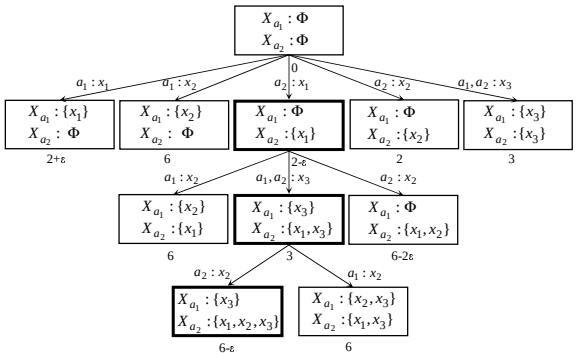

Figure 3: Greedy Strategy for Assigning Targets

The greedy strategy assigns one unassigned target per round to a qualified coalition so that the resulting partial assignments have the smallest team cost. The function call **AssignTarget**(*GREEDY*, $\{\emptyset\}_{a\in A}, 0)$ in Figure 2 executes the greedy strategy. For each unassigned target $x \in X$ and each qualified coalition $O \subseteq A$, it sets

$$X'_a := \begin{cases} X_a \cup \{x\} & \text{if } a \in O \\ X_a & \text{if } a \notin O \end{cases}$$

(Lines 05-09). The greedy strategy calls $\mathbf{ScheduleTargets}(\{X'_a\}_{a\in A})$ to schedule the assigned targets and obtain the resulting team cost (Line 11). (We define this function later.) Assume that the partial assignments resulting from assigning target x^* to coalition O^* have the smallest team cost (Lines 14-17). Then, the greedy strategy sets $X_a := X_a \cup \{x^*\}$ for all agents $a \in O^*$ (Lines 18-20), which terminates the current round.

Example: Consider again the multi-agent routing problem in Figure 1 for the MiniMax team objective. The greedy strategy assigns the targets in three rounds, as shown in Figure 3. Each box in the figure represents (partial or complete) assignments. The number below it is the corresponding team cost. There are five possible ways of assigning targets in the first round: target x_1 to agent a_1, target x_2 to agent a_1, target x_1 to agent a_2, target x_2 to agent a_2 and target x_3 to agents a_1 and a_2. Target x_1 is assigned to agent a_2 in the first round since the resulting partial assignments have the smallest team cost, namely $2 - \epsilon$, shown by the thick border of the corresponding box. Similarly, target x_3 is assigned to agents a_1 and a_2 in the second round, and target x_2 is assigned to agent a_2 in the third round. The minimal team cost of the resulting assignments is $6 - \epsilon$ and thus not the minimal team cost.

5.2 Systematic Strategy

The greedy strategy assigns one unassigned target per round to a qualified coalition so that the resulting partial assignments have the smallest team cost. The team costs of these partial assignments can be calculated quickly but often do not predict the team costs of their completions well. The systematic strategy therefore assigns one unassigned target per round to a qualified coalition so that the greedy completions of the resulting partial assignments have the smallest team cost. The function call **AssignTarget**(*SYSTEMATIC*, $\{\emptyset\}_{a\in A}, 0)$ in Figure 2 executes the systematic strategy. For each unassigned target $x \in X$ and each qualified coalition $O \subseteq A$, it sets

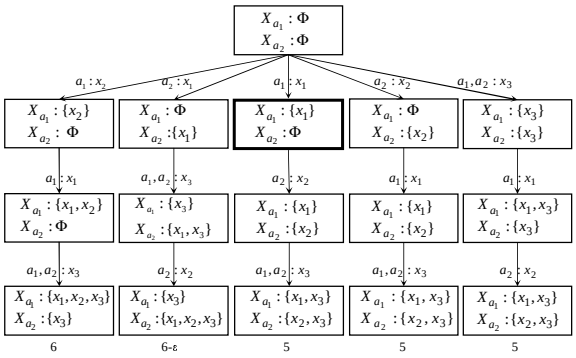

Figure 4: Simplified Systematic Strategy with $k = 2$ (= First Round of Systematic Strategy) for Assigning Targets

$$X'_a := \begin{cases} X_a \cup \{x\} & \text{if } a \in O \\ X_a & \text{if } a \notin O \end{cases}$$

and then uses the greedy strategy to complete these partial assignments to complete assignments (Line 13). It then determines the team costs of these complete assignments. Assume that the greedy completions of the partial assignments resulting from assigning target x^* to coalition O^* have the smallest team cost. Then, the systematic strategy sets $X_a := X_a \cup \{x^*\}$ for all agents $a \in O^*$, which terminates the current round.

Example: Consider again the multi-agent routing problem in Figure 1 for the MiniMax team objective. The systematic strategy assigned the targets in three rounds, as shown in Figure 4 for the first round only. Target x_1 is assigned to agent a_1 in the first round since the greedy completions of the resulting partial assignments have the smallest team cost (tied with two other ways of assigning targets), namely 5. Similarly, target x_2 is assigned to agent a_2 in the second round, and target x_3 is assigned to agents a_1 and a_2 in the third round. The minimal team cost of the resulting assignments is 5 and thus the minimal team cost.

5.3 Simplified Systematic Strategy

The systematic strategy assigns one unassigned target per round to a qualified coalition so that the greedy completions of the resulting partial assignments have the smallest team cost, which is more runtime intensive than the greedy strategy but has the potential to result in smaller team costs. In earlier rounds, the partial assignments that are evaluated by the greedy strategy are not very predictive for their greedy completions that are evaluated by the systematic strategy. The simplified systematic strategy therefore uses the systematic strategy in the first rounds but the greedy strategy in the last k rounds, where k is a parameter. Thus, it is identical to the greedy strategy if $k = \infty$ and identical to the systematic strategy if $k = 0$. The simplified systematic strategy is more runtime intensive than the greedy strategy but less runtime intensive than the systematic strategy. It has the potential to result in smaller team costs than the greedy strategy and almost as small team costs as the systematic strategy. The function call **AssignTarget**(*SIMPLIFIEDSYSTEMATIC*, $\{\emptyset\}_{a \in A}, k$) in Figure 2 executes the simplified systematic strategy.

Example: Consider again the multi-agent routing problem in Figure 1 for the MiniMax team objective. The simplified systematic strategy with $k = 2$ assigns the targets in three rounds, as shown in Figure 4. The target assigned in the first round is determined as by the systematic strategy. Thus, target x_1 is assigned to agent a_1 in the first round. The targets assigned in the next two rounds are determined as by the greedy strategy. Thus, target x_2 is assigned to agent a_2 in the second round, and target x_3 is assigned to agents a_1 and a_2 in the third round. The minimal team cost of the resulting assignments is 5 and thus the minimal team cost.

6. SCHEDULING TARGETS

The strategies for assigning targets call $\mathbf{ScheduleTargets}(\{X'_a\}_{a \in A})$ to schedule the assigned targets and obtain the resulting team cost, which is an NP-hard problem (similar to the traveling salesperson problem). We now explain how this function can be implemented. Clearly, there always exists a way of scheduling targets since the strategies for assigning targets assign targets only to qualified coalitions. Our strategies for scheduling targets first determine the order in which the agents should visit the targets assigned to them, called their schedules, and only then the visit times of the targets. Our strategies determine the schedules using the same principles as our strategies for assigning targets. They call $\mathbf{Schedule}(\{\vec{X}_a\}_{a \in A})$ in Figure 5 to determine the earliest visit times of the targets that are consistent with the schedules and obtain the resulting team cost.[1]

6.1 Naive Strategy

The naive strategy makes the agents visit the targets in the order in which they were assigned to them, which is completely determined since the strategies for assigning targets assign one target per round. In other words, each agent inserts the target at the end of its schedule whenever it is assigned a target.

Example: Consider again the multi-agent routing problem in Figure 1 for the MiniMax team objective. Assume that some strategy assigns target x_3 to agent a_1 and targets x_1, x_3 and x_2 to agent a_2 (in this order). The minimal team cost of the resulting assignments is $6 - \epsilon$. The naive strategy thus results in the schedules where agent a_1 visits target x_3 and agent a_2 visits targets x_1, x_3 and x_2 (in this order). Agent a_1 visits target x_3 at time 3. Agent a_2 first visits target x_1 at time $2 - \epsilon$, then target a_2 at time 3 and finally target x_2 at time 6. Thus, the team cost is 6 and thus not the minimal team cost of these assignments.

6.2 Greedy Strategy

Our other strategies for scheduling targets schedule one unscheduled target per round until all targets have been scheduled, as shown in Figure 5. Consider an arbitrary round. Let the schedules at the end of the previous round be $\{\vec{X}_a\}_{a \in A}$. The greedy strategy schedules one unscheduled target per round so that the resulting partial schedules have the smallest team cost. The function call **ScheduleTargets**(*GREEDY*, $\{\emptyset\}_{a \in A}, \{X_a\}_{a \in A}, 0$) in Figure 5 executes the greedy strategy and can be used in Figure 2 if one adds the remaining arguments. We do not explain its pseudo code since it as similar to the one in Figure 2.

[1]We write the schedule of agent $a \in A$ as $\vec{X}_a$. The pseudo code uses the function $\vec{X} + \{x\}$ to insert target $x \in X$ at the end of schedule $\vec{X}$ and the function $d(x, x')$ to compute the distance from target x to target x'. The pseudo code uses the variable $\bar{x}_a$ to store the last target in the current partial schedule of agent a and variable t_a to store the visit time of the target. The pseudo code repeatedly adds the agents a to the coalitions O_{x_a} of the targets x_a that they visit next (Lines 35-36). Once a coalition is qualified for a target, the pseudo code calculates the smallest visit time of the target, removes it from the schedules of the agents for its coalition (Line 37-44), and repeats the process.

Example: Consider again the multi-agent routing problem in Figure 1 for the MiniMax team objective. Assume that some strategy assigns target x_3 to agent a_1 and targets x_1, x_3 and x_2 to agent a_2. The greedy strategy then schedules the targets in three rounds. Target x_1 is scheduled in the first round since the team costs of the partial schedules that result from scheduling targets x_1, x_2 and x_3 are $2-\epsilon$, 2 and 3, respectively. Similarly, target x_3 is scheduled in the second round, and target x_2 is scheduled in the third round. The team cost of the resulting schedules is 6 and thus not the minimal team cost of these assignments.

6.3 Systematic Strategy

The systematic strategy schedules one unscheduled target per round so that the greedy completions of the resulting partial schedules have the smallest team cost. The function call **ScheduleTargets**(*SYSTEMATIC*, $\{\emptyset\}_{a\in A}, \{X_a\}_{a\in A}, 0$) in Figure 5 executes the systematic strategy and can be used in Figure 2 if one adds the remaining arguments.

Example: Consider again the multi-agent routing problem in Figure 1 for the MiniMax team objective. Assume that some strategy assigns target x_3 to agent a_1 and targets x_1, x_3 and x_2 to agent a_2. The systematic strategy then schedules the targets in three rounds. Target x_2 is scheduled in the first round since the team costs of the greedily completions of the partial schedules that result from scheduling targets x_1, x_2 and x_3 are 6, $6-\epsilon$ and $8-2\epsilon$, respectively. Similarly, target x_3 is scheduled in the second round, and target x_1 is scheduled in the third round. The team cost of the resulting schedules is $6-\epsilon$ and thus the minimal team cost of these assignments.

6.4 Simplified Systematic Strategy

The simplified systematic strategy uses the systematic strategy in the first rounds but the greedy strategy in the last k rounds, where k is a parameter. The function call **ScheduleTargets**(*SIMPLIFIEDSYSTEMATIC*, $\{\emptyset\}_{a\in A}, \{X_a\}_{a\in A}, k$) in Figure 5 executes the simplified systematic strategy and can be used in Figure 2 if one adds the remaining arguments.

Example: Consider again the multi-agent routing problem in Figure 1 for the MiniMax team objective. Assume that some strategy assigns target x_3 to agent a_1 and targets x_1, x_3 and x_2 to agent a_2. The simplified systematic strategy with $k=2$ then schedules the targets in three rounds. The target scheduled in the first round is determined as by the systematic strategy. Thus, target x_2 is scheduled in the first round. The targets scheduled in the next two rounds are determined as by the greedy strategy. Thus, target x_3 is scheduled in the second round, and target x_1 is scheduled in the third round. The team cost of the resulting schedules is $6-\epsilon$ and thus the minimal team cost of these assignments.

7. COMPLEXITY ANALYSIS

We now analyze the runtime complexity of our strategies. Remember that m is the number of targets, n is the number of agents and k is the parameter for the simplified systematic strategy. c is the largest possible coalition size for any target and thus bounded by $\max_{x\in X}\sum_{i=1}^{r} b_i^x$ from above.

We first consider the strategies for assigning targets. The strategies have $O(m)$ rounds. In each round, a strategy evaluates all combinations of $O(m)$ unassigned targets and $O(n^c)$ coalitions for the targets. Thus, they evaluate $O(m^2n^c)$ partial assignments. The greedy strategy schedules targets for each assignment once, for a total of $O(m^2n^c)$ schedule determinations. The systematic strategy and the simplified systematic strategy greedily complete each partial assignment (in the first $O(m-k)$ rounds for the simplified systematic strategy) once with the greedy strategy, for a total of $O(m^2n^c)$ applications of the greedy strategy and $O(m^4n^{2c})$ schedule determinations.

We now consider the strategies for scheduling targets, which call the function **Schedule**. The runtime of our simple implementation of this function is $O(mnc)$. The naive strategy calls it once, for a total runtime of $O(mnc)$. The other strategies have $O(m)$ rounds. In each round, a strategy evaluates $O(m)$ targets. Thus,

Function Schedule($\{\vec{X}_a\}_{a\in A}$)
Input:
  $\{\vec{X}_a\}_{a\in A}$: the current partial schedules.
Output:
  $\{(X_a, C_a)\}_{a\in A}$: the allocations;
  lowestevaluation: the team cost of the allocations.
{26} **Foreach** $a \in A$
{27} $X_a := \emptyset$;
{28} $C_a := \emptyset$;
{29} $\bar{x}_a :=$ the current location of agent a;
{30} $t_a = 0$;
{31} **Foreach** $x \in X$
{32} $O_x = \emptyset$;
{33} **While** ($\{\vec{X}_a\}_{a\in A} \neq \{\emptyset\}_{a\in A}$)
{34} **Foreach** $a \in A$ **With** $\vec{X}_a \neq \emptyset$
{35} $x_a :=$ the first target in $\vec{X}_a$;
{36} $O_{x_a} := O_{x_a} \cup \{a\}$;
{37} **If** O_{x_a} is qualified for x_a
{38} $t' := \arg\max_{a'\in O_{x_a}} (t_{a'} + d(\bar{x}_{a'}, x_a))$;
{39} **Foreach** $a' \in O_{x_a}$
{40} $t_{a'} := t'$;
{41} $X_{a'} := X_{a'} \cup \{x_a\}$;
{42} $C_{a'} := C_{a'} \cup \{x_a \leftarrow t_{a'}\}$;
{43} $\bar{x}_{a'} := x_a$;
{44} remove x_a from $\vec{X}_{a'}$;
{45} **If** the team objective is MiniMax
{46} *lowestevaluation* $:= \arg\max_{a\in A} t_a$;
{47} **Else** /* the team objective is MiniSum */
{48} *lowestevaluation* $:= \sum_{a\in A} t_a$;
{49} **Return** ($\{(X_a, C_a)\}_{a\in A}$, *lowestevaluation*)

Function ScheduleTargets(*STRATEGY*, $\{\vec{X}_a\}_{a\in A}, \{X_a\}_{a\in A}, k$)
Input:
  STRATEGY: the strategy for scheduling targets;
  $\{\vec{X}_a\}_{a\in A}$: the current partial schedules;
  $\{X_a\}_{a\in A}$: the current assignments;
  k : the parameter for the simplified systematic strategy.
Output:
  $\{(X_a, C_a)\}_{a\in A}$: the final complete allocations;
  lowestevaluation: the team cost of the final complete allocations.
{50} **While** ($|\cup_{a\in A} X_a| > k$)
{51} *lowestevaluation* $:= \infty$;
{52} **Foreach** $x \in \cup_{a\in A} X_a$
{53} **Foreach** $a \in A$
{54} **If** $x \in X_a$
{55} $\vec{X}'_a := \vec{X}_a + x$;
{56} $X'_a := X_a \setminus \{x\}$;
{57} **Else**
{58} $\vec{X}'_a := \vec{X}_a$;
{59} $X'_a := X_a$;
{60} **If** *STRATEGY = GREEDY*
{61} $(\{(X_a, C_a)\}_{a\in A}, \textit{evaluation}) :=$ *Schedule*$(\{\vec{X}'_a\}_{a\in A})$;
{62} **Else**
{63} $(\{(X_a, C_a)\}_{a\in A}, \textit{evaluation}) :=$ *ScheduleTargets*(*GREEDY*, $\{\vec{X}'_a\}_{a\in A}, \{X'_a\}_{a\in A}, 0$);
{64} **If** *evaluation* $<$ *lowestevaluation*
{65} *lowestevaluation* $:=$ *evaluation*;
{66} $x^* := x$;
{67} **Foreach** $a \in A$
{68} **If** $x^* \in X_a$
{69} $\vec{X}_a := \vec{X}_a + x^*$;
{70} $X_a := X_a \setminus \{x^*\}$;
{71} **If** ($|\cup_{a\in A} X_a| > 0$)
{72} $(\{(X_a, C_a)\}_{a\in A}, \textit{lowestevaluation}) :=$ *ScheduleTargets*(*GREEDY*, $\{\vec{X}_a\}_{a\in A}, \{X_a\}_{a\in A}, 0$);
{73} **Else**
{74} $(\{(X_a, C_a)\}_{a\in A}, \textit{lowestevaluation}) :=$ *Schedule*$(\{\vec{X}_a\}_{a\in A})$;
{75} **Return** ($\{(X_a, C_a)\}_{a\in A}$, *lowestevaluation*);

Figure 5: Pseudo Code for Scheduling Targets

		Greedy Assignment						Simplified Systematic Assignment				ARF	
		Naive Scheduling		Greedy Scheduling		Simplified Systematic Scheduling		Naive Scheduling		Greedy Scheduling			
Agents	Targets	TC	RT	TC	RT	TC	RT	TC	RT	TC	RT	TC	RT
4	20	1320.4	0.00	1311.6	0.02	1250.2	0.2	1124.8	0.08	1045.7	1.61	1298.4	1.8
4	30	1616.0	0.08	1616.0	0.09	1587.4	1.7	1460.6	0.42	1428.0	7.64	1524.7	1.8
4	40	2019.4	0.01	2019.4	0.34	2010.9	8.5	1750.6	1.22	1696.3	24.9	1995.8	1.8
6	20	1136.9	0.04	1190.4	0.08	1125.2	1.1	1063.9	1.17	1001.3	24.3	1108.5	0.5
6	30	1471.5	0.02	1471.6	0.45	1451.6	8.7	1329.9	4.92	1272.8	79.09	1451.2	0.5
6	40	1685.5	0.04	1684.4	1.57	1685.4	42.8	1586.4	15.5	1521.2	454.4	1646.7	0.5
8	20	1060.5	0.02	1053.4	0.25	1028.6	3.3	970.1	7.16	917.2	121.3	1022.8	0.2
8	30	1325.4	0.04	1343.3	1.24	1322.0	25.5	1207.2	28.8	1150.4	501.5	1330.9	0.2
8	40	1580.4	0.08	1579.8	4.45	1570.6	84.5	1483.5	93.9	—	—	1556.4	0.2
10	20	963.6	0.03	963.6	0.54	960.6	7.0	907.8	25.5	890.4	415.0	936.7	0.2
10	30	1229.6	0.08	1229.6	2.96	1193.7	56.5	1135.7	104.2	—	—	1215.6	0.2
10	40	1489.1	0.16	1485.0	9.68	1480.2	164.1	1307.5	340.2	—	—	1489.6	0.2

Table 1: Experimental Results for Multi-Agent Routing Problems with the MiniSum Team Objective

		Greedy Assignment						Simplified Systematic Assignment				ARF	
		Naive Scheduling		Greedy Scheduling		Simplified Systematic Scheduling		Naive Scheduling		Greedy Scheduling			
Agents	Targets	TC	RT	TC	RT	TC	RT	TC	RT	TC	RT	TC	RT
4	20	367.3	0.00	367.3	0.02	345.8	0.2	320.4	0.09	316.6	1.82	385.3	0.4
4	30	425.1	0.01	421.4	0.11	398.8	2.1	372.9	0.45	366.0	5.32	447.1	0.4
4	40	529.4	0.02	528.8	0.35	492.5	8.0	455.1	1.40	452.9	15.5	545.7	0.4
6	20	207.4	0.04	213.6	0.08	210.2	1.1	182.4	1.08	186.3	23.9	217.4	0.1
6	30	259.6	0.02	261.6	0.44	251.0	8.8	224.4	4.27	215.1	81.2	268.1	0.1
6	40	303.4	0.03	313.9	1.60	308.6	40.5	263.9	15.3	256.5	400.4	319.4	0.1
8	20	175.4	0.01	175.3	0.24	165.6	3.2	150.1	6.44	143.1	100.2	176.7	0.1
8	30	202.8	0.03	203.3	1.18	192.2	20.7	172.4	26.1	165.2	433.4	213.4	0.1
8	40	247.4	0.08	247.4	4.27	236.1	68.9	205.1	85.9	—	—	253.1	0.1
10	20	132.4	0.02	132.2	0.49	130.4	6.3	114.3	26.0	108.2	412.5	144.4	0.2
10	30	159.5	0.07	158.5	2.46	158.9	50.9	140.3	101.9	—	—	163.1	0.2
10	40	188.9	0.18	187.4	9.17	180.7	149.7	186.9	321.4	—	—	202.1	0.2

Table 2: Experimental Results for Multi-Agent Routing Problems with the MiniMax Team Objective

they evaluate $O(m^2)$ partial schedules. The greedy strategy calls the function **Schedule** for the partial schedules once, for a total runtime of $O(m^3nc)$. The systematic strategy and the simplified systematic strategy greedily complete each partial schedule (in the first $O(m-k)$ rounds for the simplified systematic strategy) once with the greedy strategy, for a total runtime of $O(m^5nc)$.

The runtime of combinations of strategies for assigning and scheduling targets is their product. If r is constant, then the resulting runtimes are polynomial. However, they can be large, especially if the systematic strategy or simplified systematic strategy are used for either assigning or scheduling targets.

8. EXPERIMENTAL RESULTS

We now evaluate our strategies using multi-agent routing problems on known four-neighbor planar grids of size 51×51 with square cells that are either blocked or unblocked. The grids resemble office environments with randomly closed doors [4]. We vary the number of agents from 4, 6, 8 to 10 and the number of targets from 10, 20, 30 to 40. There exists only one capability. Each agent provides one unit of this capability, and each target requires one, two or three units of this capability. For each of these scenarios, we average over 50 runs with randomly generated cells for the locations of the agents and targets and randomly chosen quantities of the capability required by the targets. We introduced three strategies for assigning targets and four strategies for scheduling targets in this paper. We pick the five combinations with the smallest runtime and let the simplified systematic strategy perform the systematic strategy only in the first round, both for assigning targets and for scheduling targets. We compare these combinations (that use optimized versions of the pseudo code presented earlier) against the most closely related existing work, namely ARF (= Approach with Reaction Functions) [12].

Tables 1 and 2 present the resulting average runtimes (RT) in seconds and the resulting average team costs (TC). We terminated each multi-agent routing algorithm after 1000 seconds if it had not terminated by then. The results show that there is a trade-off between the runtime of a combination and the resulting team cost. The runtime increases only slowly with the number of agents or targets if the simplified systematic strategy is used neither for assigning nor scheduling targets, otherwise it increases quickly. Using the

simplified systematic strategy rather than the greedy strategy for assigning targets reduces the team costs on average by about 9-13 percent for the MiniSum team objective and 13-15 percent for the MiniMax team objective (everything else being equal). Using the simplified systematic strategy rather than the naive or greedy strategy for scheduling targets reduces the team costs on average by about 1-2 percent for the MiniSum team objective and 4 percent for the MiniMax team objective (everything else being equal). Overall, using the simplified systematic strategy for assigning targets results in smaller team costs but larger runtimes than ARF for the MiniSum team objective. All of our combinations result in smaller team costs than ARF for the MiniMax team objective even those that have smaller runtimes than ARF.

9. CONCLUSIONS

We studied task-allocation problems where one wants to minimize the team cost subject to the constraint that each task must be executed by a given number of cooperative agents simultaneously. We use the same principle to develop several efficient and effective (greedy) hillclimbing strategies for both assigning and scheduling targets, namely the greedy strategy, the systematic strategy and the simplified systematic strategy. We analyzed their complexities and showed experimentally that our strategies result in smaller team costs than a previous algorithm on multi-agent routing problems with the MiniMax team objective, even those that have smaller runtimes than the previous algorithm. It is future work to make our strategies decentralized to distribute their computations among the agents and thus reduce their runtime.

Acknowledgments

This research was partly supported by NSF awards under contracts ITR/AP0113881, IIS-0098807, IIS-0350584 and IIS-0413196 as well as seed funding from NASA's Jet Propulsion Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, companies or the U.S. government.

10. REFERENCES

[1] S. Abdallah and V. Lesser. Organization-based cooperative coalition formation. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Techonology, IAT*, pages 162–168, China, September 2004.

[2] V. D. Dang and N. Jennings. Generating coalition structures with finite bound from the optimal guarantees. In *Proceedings of the Third International Joint Conference on Autonomous Agents and MultiAgent Systems*, pages 564–571, 2004.

[3] M. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[4] S. Koenig, C. Tovey, X. Zheng, and I. Sungur. Sequential bundle-bid single-sale auction algorithms for decentralized control. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1359–1365, Hyderabad, India, 2007.

[5] S. Kraus, O. Shehory, and G. Taase. Coalition formation with uncertain heterogeneous information. In *Proceedings of the Second International Joint Conference on Autonomous Agents and MultiAgent Systems*, pages 1–8, 2003.

[6] H. C. Lau and L. Zhang. Task allocation via multi-agent coalition formation: Taxonomy, algorithms and complexity. In *ICTAI '03: Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, page 346, Washington, DC, USA, 2003.

[7] T. Sandholm and V. R. Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1-2):99–137, 1997.

[8] O. Shehory and S. Kraus. Task allocation via coalition formation among autonomous agents. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 655–661, Montréal, Québec, Canada, 1995.

[9] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2):165–200, 1998.

[10] L.-K. Soh and X. Li. An integrated multilevel learning approach to multiagent coalition formation. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 619–624, Acapulco, Mexico, 2003.

[11] G. Yong, Y. Li, Z. Wei-ming, S. Ji-chang, and W. Chang-ying. Methods for resource allocation via agent coalition formation in grid computing systems. In *Proceedings of IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, pages 295–300, 2003.

[12] X. Zheng and S. Koenig. Reaction functions for task allocation to cooperative agents. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and MultiAgent Systems (in print)*, 2008.