Advances in Cognitive Systems 9 (2020) 51–70 Submitted 5/2019; published 12/2020

Goal-based Team Crisis Intervention

Matthew Molineaux MATTHEW.MOLINEAUX@PARALLAXRESEARCH.ORG
Michael T. Cox MICHAEL.COX@PARALLAXRESEARCH.ORG
Parallax Advanced Research, Beavercreek, OH 45431 USA

Abstract

The team crisis intervention problem concerns how to monitor and intervene in a team's goal achievement process. In this paper, we give a formal representation of this problem and describe how it can be solved via breakdown into three subproblems: state estimation, crisis recognition, and intervention generation. With respect to intervention generation, we describe goal assignment structure refinement, a new mechanism that hierarchically decomposes goals without creating a full plan. This results in a novel representation called *goal assignment structures*. These improve on plans for representing teamwork by simplifying recommendations and limiting unnecessary interventions. We then report empirical evidence that goal assignment structure refinement improves team responsiveness to crises, increases team effectiveness, and scales better than full hierarchical planning for team crisis intervention.

1. Introduction

Team goal achievement involves the cooperation of team members toward individual and shared goals necessary for success. This involves creating and executing plans that achieve the team's goals and, as such, team goal achievement includes both planning and acting. Although this topic is much studied in the multi-agent systems and planning literatures (Dix, Muñoz-Avila, Nau, & Zhang, 2003; Cohen & Levesque, 1991; Weiss, 2013; Sims, Corkill, & Lesser, 2013), we examine team goal achievement from an external perspective. That is, we consider how a system (or agent) can reason about the team goal achievement process, recognize when the team is in a crisis, and intervene in the process to assure group success. We view team goal achievement as a specialized form of goal reasoning (Aha, 2018; Aha, Cox, & Muñoz-Avila, 2013; Cox, 2017; Hawes, 2011; Klenk, Molineaux, & Aha, 2013) that constitutes selection of a dynamic set of goals that can shift in response to changing circumstances. When teams do not respond effectively, team performance degrades. We refer to changing circumstances that may degrade team performance as *complications* and to the team's failure to respond to such complications as a *team crisis*. In this paper, we present a novel model of strategic, team goal achievement, and we use this model to describe an approach for recognizing crises and intervening by changing the team's goals.

High profile examples of team crises include the failure of a flight crew to address simultaneous problems, resulting in the plane crash of UA Flight 173 in 1978 (Whipple, 2015) and the failure of a black-ops team to retake initiative after a helicopter crash, as depicted in the film *Black Hawk Down* (Bowden, 1999). In general, team crises occur when a team fails to adapt shared plans and goals to a changing situation, resulting in failure. We expect that the team crisis intervention method introduced here could help such teams.

To promote reusability, this paper describes a system in such a way that the structure and components of the solution can be replicated and reused independently. To do so, we provide formal definitions for both a large, high-level problem (i.e., team crisis intervention), and three subproblems that can be solved by simpler mechanisms. To connect them, we use the novel concept of a "cognitive skeleton", which is related to a cognitive architecture in that it supports a combination of multiple capabilities and replaceable knowledge. A skeleton differs in that it is lightweight and problem specific. The formal problem definitions and implementation of the cognitive skeleton are important contributions of this work.

Other contributions include (1) a novel data structure for describing team collaborations, called a *goal assignment structure*; (2) a formal definition of goal assignability, which describes what goals a team should delegate to subteams and how to do so; and (3) a novel procedure for refining goal assignment structures. We report results from crisis response experiments that use estimates of probable human reactions; these experiments provide evidence for the efficacy of these novel formulations and processes.

2. Team Goal Achievement Notation and Theory

Our research draws on definitions from automated planning, particularly hierarchical goal networks. In this section, we review foundational terminology from those fields and use them to advance a definition of goal assignment structures and assignability for use in team crisis interventions. These support the body of our theory, which describes:

- Goal assignment structures, which use shared mental models of team/subteam responsibilities;
- Assignability, which is a characteristic of goals that can be delegated to a subteam.

According to our theory, goal assignment structures are sufficient to describe a team's mental model of a problem solution, and they can be created by a simple planning process.

2.1 Terminology

We base our definitions for team goal achievement on standard definitions from the literature on automated planning and acting (Ghallab, Nau, & Traverso, 2016). As such, an environment Σ is defined as a tuple (S, A, γ), where S is the space of environment states, A the space of actions, and γ the transition function $S \times A \rightarrow 2^S$. We also define a goal space $G \subset 2^S$, and a set of individual performers I. A team $t \in T$ is then a tuple (M, St), where $M \subset I$ is a set of team members, and St is a possibly empty set of subteams of t whose members are a subset of M.

An action model α is defined as a four-tuple $(head(\alpha), performer(\alpha), pre(\alpha), eff(\alpha))$. Here, $head(\alpha)$, $pre(\alpha)$, and $eff(\alpha)$ are defined as usual: respectively, they are the head (name and variable parameters), preconditions (a set of literal statements about state variables and relations) and effects (a set of assignments to state variables) of the action model. The item $performer(\alpha)$ is a term of type individual that denotes who is responsible for executing the named action. Typically, this will be a variable that is constrained by the preconditions to be an individual capable of performing the action. In a ground action a, the value of performer $performer(a)$ must be an individual in I. A plan π is a finite sequence of actions $\langle a_1, a_2, \ldots a_n \rangle$.

2.2 Goal Decomposition Rules

We also define search knowledge in the form of goal decomposition rules (Langley & Choi, 2006; Shivashankar, Alford, & Aha, 2017), which provide advice to a reasoner on how to decompose a goal. Semantically, a goal decomposition rule breaks a parent goal into subgoals that are more easily accomplished. However, accomplishing the subgoals is not always sufficient to ensure the parent goal is accomplished. We extend prior definitions of goal decomposition rules with (1) a notion of active constraints that must hold while accomplishing the parent goal and (2) (team-subgoal) assignment pairs. The active constraints let a rule prevent selection of certain subgoals. For example, a decomposition rule for clearing a building could state the subgoals of clearing each room and include a constraint that the entryway must not be left unguarded. Without such constraints, search can be guided toward useful solutions but not away from poor ones.

A *goal decomposition rule gdr* is represented by the following six-tuple:

$$\big(name(gdr), goal(gdr), pre(gdr), assignedSubgoals(gdr), preced(gdr), constr(gdr)\big) .$$

Here we diverge significantly from the original definition. The name $name(gdr)$ of a goal decomposition rule is merely a unique identifier. The decomposed goal $goal(gdr)$ is a literal describing the goal about which the rule gives advice. The preconditions $pre(gdr)$ are defined as for an action model. The assigned subgoals $assignedSubgoals(gdr)$ is a list of tuples (g, t), where g is a literal describing a subgoal to be accomplished before $goal(gdr)$ and t is a term identifying a team or the value $\perp$, which indicates that g is not assigned. A precedence relation $preced(gdr)$ gives a list of tuples (ge, gl) that indicate the order in which subgoals are to be accomplished. Finally, the constraints $constr(gdr)$ describes a set of states that are not to be entered before achieving the goal $goal(gdr)$.

A goal decomposition rule is well formed if and only if the variables in the subgoals, precedences, and constraints are a subset of the variables in the goal and in the preconditions. Given a function *variables* that returns all variables within a structured statement, the following expression holds:

$$variables(assignedSubgoals(gdr)) \cup variables(preced(gdr)) \cup variables(constr(gdr)) \subset variables(goal(gdr)) \cup variables(pre(gdr))$$

The example goal decomposition rule in Table 1 named find-assault-team-for describes knowledge for clearing a building of enemies in an urban combat scenario. In the Lisp syntax shown, variables are indicated by a leading question mark. The preconditions require that the closest entrance to the best available team is not threatened. If the preconditions are met, then the original goal (cleared ?target-building) is decomposed into two subgoals, which we discuss further in Section 2.4.

2.3 Goal Assignment Structures

The solution to a team problem is described by a *goal assignment structure*. This structure recursively decomposes a shared team goal into subgoals until reaching ones for which subteams can plan autonomously. This structure ensures that subteam actions support the group goals without requiring that the entire team knows and agrees on a complete plan. As a concept, goal assignment structures are an extension of the concept of *goal networks* defined by Shivashankar (2015). Goal networks incorporate goal nodes and relationships; goal assignment structures add team assign-

Table 1. Example goal decomposition rule for the goal (cleared ?target-building).

```
(:gdr (find-assault-team-for ?subteam ?target-building)
   ;goal
   (cleared ?target-building)
   ;preconditions
   ((best-available-team ?subteam ?target-building assault)
    (closest-entrance ?target-building ?entrance ?subteam)
    (not (threatened-at ?team ?entrance ?threat-building)))
   ;subgoals
   (((assigned ?subteam (cleared ?target-building)))
    ((delegate (cleared ?target-building) ?subteam))))
```

ments and *justifications* in the form of a goal-subgoal graph. Formally, a goal assignment structure gas is stated as a tuple $(Gn, P, C, As, t = (M, St))$, where Gn denotes a set of goal nodes $gn = (id(gn), goal(gn))$, each containing a unique identifier and a goal literal. Here P is a set of temporal precedences over Gn, given as a list of tuples (ge, gl) that contain an earlier and later goal, and C is a set of goal-subgoal relationships over Gn, given as a list of tuples (gp, gc) that contain parent and child goals. The team t is responsible for accomplishing the goals in the goal structure described earlier as the tuple (M, St). As is an assignment function $Gn \rightarrow T \cup \{\perp\}$ that assigns every goal node in Gn to a subteam in St or to no subteam (i.e., $\perp$).

Figure 1 depicts a goal assignment structure from the urban combat domain that represents a solution to a hostage rescue problem. The team assigned to solve this problem (not shown) has three subteams (i.e., Subteam1, Subteam2, and Subteam3) that must handle different facets of the task in a coordinated manner.

2.4 Assignability

Because goal assignment structures assign some goals and not others to subteams, it is important to reason about what goals can be assigned. We refer to a goal as *assignable* if responsibility can be safely delegated to a subteam without further considering subgoals. For example, the goal cleared(K3) in Figure 1 can be assigned to Subteam2 without having to also assign them lower level goals, such as clearing individual rooms in K3. While the team is capable of reasoning about the actions involved in achieving the goal, doing so would slow them down, so they treat the goal as an operation at the level of goal assignment structure that need not be subdivided. We consider three conditions for determining whether a subgoal is assignable:

1. Subteam st of t can accomplish subgoal g without assistance;
2. Subteam st of t recognizes when previous goals are accomplished; and
3. Subteam st of t recognizes when subgoal g is accomplished.

More formally, a subgoal g with precedence relations P is assignable to subteam $st = (M, St)$, i.e., $assignable(g, P, st)$, iff three conditions hold:

1. $\exists \pi = < a_1 \ldots a_n >, s_0, s_1, \ldots, s_n : \forall i, 0 \leq i \leq n$:
   $s_i \in \gamma(s_{i-1}, a_i) \wedge (performer(a_i) = st \vee (performer(a_i) = (M', St') \wedge M' \subset M))$
2. $\forall s, g' : (g', g) \in P \rightarrow believes(st, g', s) \equiv s \vDash g'$
3. $\forall s : believes(st, g, s) \equiv s \vDash g$

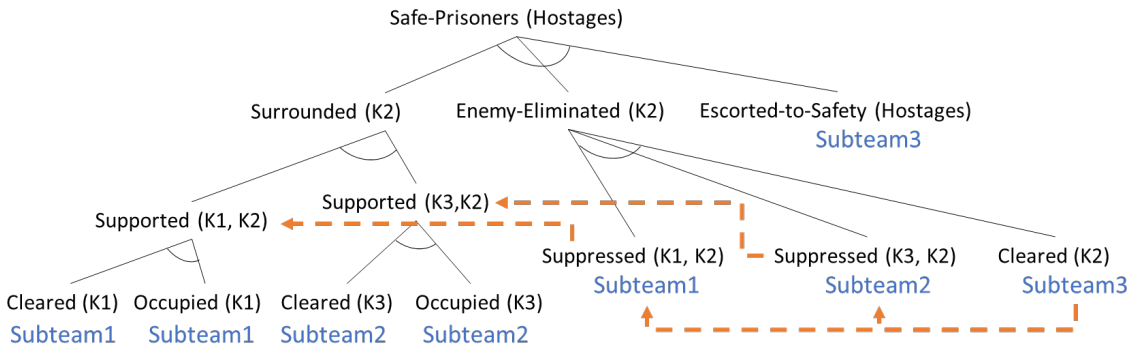

Figure 1. Example goal assignment structure. Nodes depict goal literals, unbroken black lines indicate goal-subgoal relationships, dashed orange lines depict temporal precedences, and labels in blue refer to the entities assigned to achieve the goal immediately above them.

Implementations in this work assume that assignability knowledge is compiled into goal decomposition rules and accompanying derived concepts; that is to say, a rule *gdr* only assigns a subgoal g to a subteam st if it first guarantees that it is assignable. Formally, all goal decomposition rules *gdr* must satisfy the expression

$$(g,t) \in assignedSubgoals(gdr) \wedge g \neq \perp \rightarrow assignable(g, preced(gdr), t)\,.$$

The find-assault-team-for rule in Table 1 assigns and then delegates the subgoal (cleared ?target-building). Its preconditions ensure that the first assignability condition is satisfied by requiring that the selected subteam is the best available for assault, a role that incorporates the notion of ability to accomplish cleared goals. The condition that the entrance is not threatened ensures that there are no previous goals that must be accomplished first, ensuring condition 2. Finally, condition 3 is met through a definition of the cleared concept that relies on the fully observable base concepts (see Table 2) cleared-room and room-building. As these conditions are met, find-assault-team-for delegates the cleared goal to the team bound by the preconditions and the planner will treat it as satisfied. The assigned subgoal is accomplished by a single action and serves as a "mental note" of which team was responsible.

3. Team Crisis Intervention

We can now describe our approach to team crisis intervention. In this section, we briefly specify the main intervention problem and its subproblem decomposition. We then provide a skeleton that combines the solutions to individual subproblems and specify formal definitions for each one. The implementations reported in this paper address these subproblems. We realize that this approach is unusual, but our hope is that these choices will make our work reusable by others.

3.1 Main Problem Definition

The objective of *team crisis intervention* is to assist a team when it starts to falter so that its goals are achieved and failure avoided. From a teaming agent's perspective, the problem can be stated:

Table 2. Definition of the cleared-building concept. A building is cleared if all its rooms are clear.

```
(:- (cleared-building ?building)
   (and
      ;;building has at least 1 room
      (eq (room-building ?some-room) ?building)
      ;; all rooms in building are determined cleared
      (forall (?r – room)
         ;such that
         (eq (room-building ?r) ?building)
         ;ensure
         (cleared-room ?r) ) ))
```

Given: (1) an environment Σ; (2) an initial state $s_0 \in S$; (3) a top-level goal $g^* \in G$; (4) a team t; (5) an environment interface EI; and (6) a team interface TI;

Find: A sequence of messages, delivered over time to TI, that result in the team t reaching a final state $s_f \vDash g^*$.

We reduce the team crisis intervention problem to three subproblems: *state estimation* (detailed in Section 3.3); crisis recognition (Section 3.4); and intervention generation (Section 3.5). A cognitive skeleton provides the framework to combine procedures that solve these problems into a unified solution. Figure 2 depicts the relationships between these subproblems within the cognitive skeleton. Our team crisis intervention system is composed of this skeleton and components that address each subproblem. Experiments in Section 4 compare this system with alternative solutions that use the same skeleton and different components.

3.2 Cognitive Skeleton

The team crisis intervention skeleton is responsible for intervening in the event of a crisis to improve team performance. The theory behind it is simple: when a crisis occurs, the skeleton suggests an alternative problem solution for the team to consider in the form of a goal assignment structure. The skeleton continually monitors the environment, creating a new estimate of the current state every time an observation is received. Crisis recognition is then rerun and, if a crisis is signaled, the goal reasoner is called to provide a new goal assignment structure. The skeleton then intervenes by suggesting the new goal assignment structure to the team (see Table 3). Observation of the environment and suggestions to the agents is implemented via message passing, which also synchronizes the skeleton with both environment and agents.

The cognitive skeleton defines a family of solutions to the team crisis intervention problem. In our urban combat scenario, the skeleton is responsible for recognizing signs of an ambush (using state estimation), recognizing that the team will fail to anticipate the ambush (using crisis recognition), and suggesting an alternative problem solution in the form of a goal assignment structure that avoids the ambush (using intervention generation). Alternate solutions to the problem may adopt the skeleton and solve the subproblems in different ways or use a different cognitive skeleton altogether (or none). The skeleton is intended to facilitate reusability and comparability.

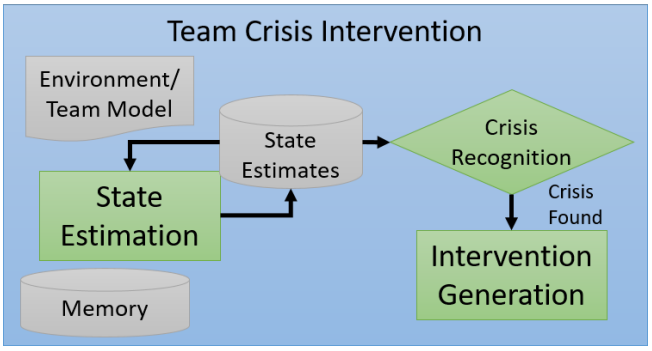

Figure 2. The cognitive skeleton for the team crisis intervention problem integrates the three subproblems of state estimation, crisis recognition, and intervention generation.

3.3 State Estimation

Because environments are only partially observed, an agent needs to estimate the true state of the world. This is an inference problem, which we can state as:

Given: (1) an environment Σ; (2) a history of observations Oh; (3) a history of actions Ah; and (4) a prior estimate of the state es_{c-1};
Find: An estimate of the state es_c at the time of the most recent observation in Oh.

Here we consider two approaches to state estimation, neither of which is novel to this work: the DiscoverHistory algorithm (Molineaux, 2017) and a trivial approach that assumes the current state is the same as the most recent observation. DiscoverHistory maintains a hypothesis about the initial state and the set of actions that have taken place. After each observation, it revises this hypothesis to be consistent with all observations and all known actions. DiscoverHistory then estimates the state by projecting the transition function from the hypothesized initial state over the set of hypothesized actions.

In the urban combat example, the state-estimation process infers unobservable information, such as an enemy ambush plan, in an attempt to identify developing crises before they cause disaster. DiscoverHistory does this by inferring the causes of an observable state, such as blocked entrances. Other suitable approaches might include Bayesian inference or plan recognition.

3.4 Crisis Recognition

As a team executes actions to accomplish their goals, crises may arise. The task of crisis recognition is to determine when a crisis has begun. We can state this problem as:

Given: (1) an environment Σ; (2) a (possibly estimated) state s; (3) a top-level goal to accomplish g^*; and (4) a team t;
Determine: The presence or absence of a team crisis.

We investigated two methods for crisis recognition. The "goal change" method creates a new goal assignment structure each time it attempts to recognize a crisis. This method signals a crisis if and

Table 3. Team crisis intervention cognitive skeleton. $\boldsymbol{\Sigma}$ represents the environment, $\boldsymbol{es_0}$ is the estimated initial state, $\boldsymbol{g^*}$ the top level goal, $\boldsymbol{t}$ is the team, $\boldsymbol{EI}$ the environment interface, and $\boldsymbol{TI}$ the team interface.

Procedure TEAMCRISISINTERVENTIONCOGNITIVESKELETON($\Sigma, es_0, g^*, t, EI, TI$)
begin
  $c \leftarrow 0$ // c = State counter
  while $\neg(es_c \models g^*)$ **do** // es_c = current state
    $c \leftarrow c + 1$
    waitOnQueue(EI, "observation") // Wait for observation from Σ
    $o \leftarrow$receive(EI, "observation") // Receive observation from Σ
    $Oh \leftarrow$addToEnd(Oh, o) // Add observation to history (list)
    $es_c \leftarrow$ estimate($\Sigma, Oh, \emptyset, es_{c-1}$) // Estimate state
    if RECOGNIZECRISIS(Σ, es_c, g^*, t) **then**
      $int \leftarrow$GENERATEINTERVENTION(Σ, es_c, g^*, t)
      send(TI, "intervention", int) // Intervene with team t
    send(EI, "processed", $\top$) // Let Σ continue.

only if the structure has changed, which makes sense if we expect the team to choose the best solution available and continue to pursue that solution even after it is no longer the best, thus causing a crisis. Alternatively, the "plan feasibility" method creates an initial plan for the team's goal that incorporates all the team members. It monitors that plan, removing actions from its head as they are accomplished and signals a crisis when the remaining plan is infeasible. This matches the intuition that the team will choose the best plan available and will want assistance as soon as that plan is invalidated.

Both methods constitute uninformed estimators of the times when crises occur, incorporating no observations of the team itself. Larue et al. (2019) discuss the problems involved in recognizing crises in observations of a team. In the urban combat example, crisis recognition is responsible for recognizing that the team's problem solution is insufficient to respond to the ambush and that the team will not change that solution without help. If crisis recognition recommends an altered solution when the team was ready, it is a false alarm that can waste time and decrease the team's confidence. Failure to anticipate that the team will fail to respond to an ambush may result in loss of life. General classification methods are applicable and should incorporate features informative about the team's shared mental model of the problem.

3.5 Intervention Generation

Intervention for a team that is faltering in the execution of its plans consists of a set of changes to the goal structures associated with the team and the subteams of which it is composed. Here we define the *intervention generation* subproblem as:

Given: (1) an environment Σ; (2) a state of that environment s; (3) a top-level goal to accomplish g^*; and (4) a team t;
Find: An intervention that improves the likelihood that team t will accomplish g^*.

To investigate the utility of providing goal assignment structures as interventions, we created the *Goal Assignment Structure Refinement (GASR)* procedure, a modification of Shivashankar's (2015)

Goal Decomposition Planner (GDP). Whereas GDP (Section 3.5.2) searches a space of goal network-plan tuples (gn, π) for terminal nodes in which gn is empty, GASR searches a space of goal assignment structures *gas* for terminal nodes in which all leaves of *gas* are assigned.

The base case for GASR in Table 4 occurs when the goal node stack is empty (line 3), which means that the goal assignment structure is complete and returned successfully. Otherwise, GASR takes the top node of the goal node stack (line 4) and attempts to handle it. If the goal node is assigned (line 5), there is no need to decompose. The projected state is then updated to reflect that the goal has been completed (line 6) and GASR recurses (line 7). Otherwise, GASR checks that the current goal $goal(gn)$ is not already in the set of pending goals Gp (line 8) and fails if it is; this would mean that the structure *gas* has a pathological cycle in its subgoal relationships C. All constraints (line 9) are then checked; if the projected state satisfies one, the current branch fails (line 10); otherwise, constraints linked to the current goal node are removed (line 11). If the current goal $goal(gn)$ is already met, GASR recurses, removing it from the pending goal set Gp, which is also removed from the goal assignment structure entirely if it has no subgoals (lines 12–16).

On lines 17–20, GASR finds the set U of all groundings of all goal decomposition rules that address the current goal and that have satisfied preconditions. For each element of *U*, GASR creates a search branch (line 22), pushes the current goal node back onto the goal stack (line 23), updates the goal assignment structure with subgoals (lines 24–31), and recurses (line 32).

GASR keeps track of the goal nodes that still need to be decomposed by adding them to a stack; at each iteration, it removes the top goal node from the stack. GASR addresses that goal by either assigning it (lines 5–7), removing it if already met (lines 12–15), decomposing it (lines 17–31), or backtracking due to search failure (lines 8, 10, 21). If the goal must be decomposed, it is put back on the stack (line 23). If the goal is met but has no children, it is removed from the structure (lines 13–15). Therefore, at the time the goal stack is empty, all goal assignment structure nodes will either be assigned or have subgoals.

As a conceptually simpler alternative, an intervention can simply be a plan to achieve the top-level goal. This is essentially the same as the turn-by-turn directions provided by a typical global positioning system that provides a complete new set of directions each time a user becomes "lost". Specifically, we consider an intervention generation method that calls GDP to generate plans. This is useful because it is fast and can use the same knowledge as GASR. In the urban combat scenario, team crisis intervention should propose a solution that includes drone reconnaissance of likely ambush sites and counterattacks on ambush sites prior to addressing the main objective. Alternative solutions should demonstrate that feedback improves a team performance and averts team crises.

4. Experimental Methodology

We designed a set of experiments in two distinct domains to provide evidence for three claims:

1. *Goal change-based crisis recognition improves team responsiveness to complications:* We consider time-critical situations in which the time taken to respond to a complication is expected to affect performance, as often occurs in the real world. We measure responsiveness as the difference between the time when a complication occurs and the resolution, which is necessarily domain dependent, as described below.
2. *Goal change-based crisis recognition improves team effectiveness:* We measure team effectiveness as a combination of the accomplishment of team goals and the resources expended by the teams.

Table 4. The intervention generation component and GASR. The parameter Σ represents the environment, s the state, Gns the goal node stack, gas the goal assignment structure, Gp the pending goal set, $cons$ the state constraints, and GDR the set of goal decomposition rules.

Procedure GENERATEINTERVENTION[GASR](Σ, s, g^*, t)
begin
  $gn \leftarrow (\mathsf{gensym}(), g^*)$
  return GASR$(\Sigma, s, \{gn\}, (\{gn\}, \emptyset, \emptyset, \emptyset, t), \emptyset, \emptyset, GDR)$

Procedure GASR$(\Sigma, s, Gns, gas = (Gn, P, C, As, t = (M, St)), Gp, cons, GDR)$
begin
  if $Gns = \emptyset$ **then return** gas; // Gns = Next goal node stack
  $gn \leftarrow \mathsf{pop}(Gns)$ // gn = Next goal node
  if $As(gn) \neq \perp$ **then** // Goal node assigned to subteam
    $s \leftarrow$ UPDATE$(s, goal(gn))$ // s = Projected state after gn
    return GASR$(\Sigma, s, Gns, gas, Gp \setminus \{goal(gn)\}, cons, GDR)$
  if $gn \in Gp$ **then fail** // Cycle check; Gp = Pending goals
  for $(gn', c) \in cons$ **do** // $cons$ = Current state constraints
    if $s \vDash c$ **then fail** // Constraint violation
    if $gn' = gn$ **then** $cons \leftarrow cons \setminus \{(gn', c)\}$
  if $s \vDash goal(gn)$ **then** // Goal already accomplished
    if $\forall (gnp, gnc) \in C : gnp \neq gn$ **then** // Goal has no subgoals
      $P' \leftarrow \{(gne, gnl) \in P \mid gne \neq gn \wedge gnl \neq gn\}$
      return GASR$(\Sigma, s, Gns, (Gn \setminus gn, P', C, As, t), Gp \setminus \{goal(gn)\}, cons, GDR)$
    else return GASR$(\Sigma, s, Gns, gas, Gp \setminus \{goal(gn)\}, cons, GDR)$
  $U \leftarrow \{(Sgn, Sp, Sc) \mid \exists gdr \in GDR, \exists \theta :$
    $\mathsf{subst}(\theta, goal(gdr)) = goal(gn) \wedge s \vDash \mathsf{subst}(\theta, pre(gdr))$
    $\wedge\ Sgn = \{\mathsf{subst}(\theta, sga) \mid sga \in assignedSubgoals(gdr)\}$
    $\wedge\ Sp = \mathsf{subst}(\theta, preced(gdr)) \wedge Sc = \mathsf{subst}(\theta, constr(gdr))\}$
  if $U = \emptyset$ **then fail**
  nondeterministically choose $(Sgn, Sp, Sc) \in U$ // Rest is one search branch
  $\mathsf{push}(gn, Gns)$
  $(Gn', P', C', As', t) \leftarrow gas$
  $P' \leftarrow P' \cup Sp$ // Sp = Subgoal node precedences
  for $(gnc, st) \in \mathsf{reverse}(\mathsf{sort}(Sgn, P))$ **do** // gnc = Subgoal; st = Subteam
    $Gn' \leftarrow Gn' \cup \{gnc\}$
    $C' \leftarrow C' \cup \{(gn, gnc)\}$
    $As'(gnc) \leftarrow st$
    $\mathsf{push}(gnc, Gns)$
  $cons \leftarrow cons \cup Sc$ // Sc = New state constraints
  return GASR$(\Sigma, s, Gns, (Gn', P', C', As', t), Gp \cup \{goal(gn)\}, cons, GDR)$

Table 5. The environment loop. The parameter s_0 represents the initial state, g^* the top-level goal, *CI* the controller interface, and *T* the universe of teams.

```
1  Procedure ENVIRONMENTSIMULATION(s_0, g*, CI, T)
2  begin
3      c ← 0                                              // c = State counter
4      while ¬(s_c ⊨ g*) ∧ ¬ unreachable(s_c, g*) do      // s_c = current state
5          c ← c + 1
6          TI ←nextTeamToAct(s_c)                         // TI = Team interface
7          o ←formObservation(TI)                         // o = Observation available to team
8          send(CI,"observation", o)                      // Notify controller of observation
9          waitOnQueue(CI,"processed", o)                 // Wait on controller processing
10         receive(CI,"processed")                        // Finish wait on controller processing
11         send(TI,"observation", o)                      // Notify individual of observation
12         waitOnQueue(TI,"action", o)                    // Wait on next action from team
13         a ←receive(TI,"action")                        // Receive team action
14         s_c ← γ(s_{c−1}, a)                            // Change state based on team action
```

3. *GASR is more scalable than hierarchical planning for team crisis intervention:* A significant concern for knowledge-rich methods is their ability to scale in complicated domains. Hierarchical task networks perform well in this regard and GASR attempts to decrease hierarchical planning effort by delegating lower-level planning activities to humans. Thus, we expect GASR will require fewer computational resources than would creation of a complete hierarchical plan. We measure computational effort in terms of nodes expanded during search.

We have sought to gather evidence relevant to these claims by conducting simulated exercises with estimated human reactions. This is a preliminary step prior to costly user studies.

We used two types of environments to interrogate these claims, which we will refer to as the *perimeter security* and *urban combat*. In each domain, multiple agents with estimated human behaviors control subteams that interact with an environment to achieve high-level goals. Three steps characterize the main loop within these environments:

1. Form and send an observation to the next agent to act;
2. Receive an action from that agent;
3. Transition to the next state according to the received action and model Σ.

This loop continues until either the top-level goal is accomplished or the top-level goal is rendered unreachable. Table 5 provides a formal description of the environment loop as the procedure EnvironmentSimulation. Message passing supports communication and synchronizes the simulation of all individuals, including opposing forces.

Agents that operate in these simulations execute a six-step loop to interact with the environment:

1. Prior to activity in the environment, the team performs an initial goal assignment step; this is simulated via the GASR system. The problem solution is represented as a shared goal assignment structure *gas* that the team tries to accomplish.

2. Each subteam st that has goals assigned to it then performs an initial planning activity, simulated using the GDP procedure. This creates a new plan π that accomplishes the goals assigned to that subteam by the agreed upon goal assignment structure *gas*.
3. Once a plan is created, each subteam executes its plans by sending the next action from its plan to the environment in response to each observation.
4. During execution, whenever a subteam's existing plans become infeasible (recognized by projection of the remaining plan steps from the current estimated state), that subteam replans to achieve its assigned goals. This is simulated using the GDP procedure. The return value from GDP replaces π for that subteam agent and execution continues.
5. During execution, whenever an intervention system suggests a new goal assignment structure *gas'*, the team updates its existing shared structure to *gas'*. This is again simulated using GDP and replaces the existing set of subteam-plan tuples, after which execution continues.
6. During execution, when an intervention system suggests a new plan π', the receiving subteam agent replaces its existing plan π. This is again simulated using GDP and execution continues.

Table 6 formally describes the agent reasoning cycle that controls one subteam. Actions and observations are communicated with the environment via message passing, as are interventions. Note that the initial goal assignment step (line 4) is shared among subteams of the same team; also, agents on different teams receive different top-level goals g^*.

Perimeter Security Domain. The perimeter security domain incorporates a base, a blue force team responsible for defending the base, and a red force team. The blue team consists of a patrol subteam and a defense subteam. The blue team's goal is to maintain the base's protected status; the red force team aims to either (a) bomb the base; (b) if aggressive and challenged, fight to the death against blue forces; or (c) if nonaggressive and challenged, escape alive. The simulation incorporates simple models of transiting via a grid, fighting a nearby enemy, and challenging a nearby subteam. The number of possible states in the perimeter security domain is approximately 4.4 × 108.

Urban Combat Domain. The urban combat domain incorporates a set of buildings with entrances, rooms, and windows; equipment, including grenades, guns, and ammunition, and units with variable posture and speed. Buildings are located on a two-dimensional grid; building walls block movement between some pairs of adjacent grid sectors, and each window and entrance faces onto a particular sector. Three blue fire teams each consist of four units. One of these units has control of a UAV that can be used for reconnaissance. One of the fire teams has an assault role and the other two have support roles. A red team consists of six to 15 units located in various rooms and buildings. One prisoner is on his own team. The goal of the blue team is to rescue the prisoner, whereas the goal of red team members is to either (a) kill blue team members without being detected, (b) if aggressive and detected, fight blue team members to the death, (b) if aggressive and detected, fight blue team members to the death, and (c) if nonaggressive and detected, to hide from blue team members. The urban combat domain is estimated to have around 10^{891} possible states.

5. Experimental Results

We conducted an experiment in perimeter security that compared an intervention system using the goal change crisis recognizer and GASR intervention generator to another that uses plan infeasibility and the GDP-based intervention generator. We then compared GASR intervention results in this domain along with those from the urban combat domain to determine amount of effort and scalability compared to a typical replanning approach.

Table 6. Team implementation. The parameter Σ represents the environment model, es_0 the estimated initial state, g^* the top-level goal, EI the environment interface, CI the controller interface, and t the team.

```
Procedure TEAMSIMULATION(Σ, es_0, g*, EI, CI, t)
begin
    c ← 0                                                    // c = State counter
    gn ← (gensym(), g*)
    (Gn, P, C, As, t) ← GASR(Σ, es_0, {gn}, ({gn}, ∅, ∅, ∅, t), ∅, ∅, GDR)  // Goal assign
    G ← {g ∈ Gn | As(g) = t}                                 // G = Leaf goals
    π ← GDP(Σ, es_0, (G, P))                                 // Subteam internal planning
    while π ≠ ∅ do
        c ← c + 1
        waitOnQueue(EI, "observation")
        o ← receive(EI, "observation")                       // Wait for observation
        Oh ←addToEnd(Oh, o)                                  // Add observation to history (list)
        es_c ← estimate(Σ, Oh, Ah, es_{c−1})                 // Estimate state
        if messageOnQueue(CI, "intervention") then
            int ← receive(CI, "intervention")
            a ← ((Consult), t, ∅, ∅)                         // Team consults intervention system
            if int is a goal assignment structure then
                (Gn, P, C, As, t) ← int
                G ← {g ∈ Gn | As(g) = t}
                π ← GDP(Σ, es_c, (G, P))                     // GASR intervention accepted
            else if int is a plan then
                π ← int                                      // Planner intervention accepted
        else
            if infeasible(π, es_{c−1}) ∨ π = ∅ then
                π ← GDP(Σ, es_c, (G, P))                     // Subteam replans
            a ← head(π)
            π ← tail(π)
        send(EI, "action", a)
        Ah ←addToEnd(Ah, a)                                  // Add action to history (list)
```

5.1 Perimeter Security Experiment

For purposes of determining responsiveness and effectiveness under different interventions in the perimeter security experiment, we estimated that human reaction time to consult the intervention system after a crisis is recognized would take 40 seconds. This includes an estimated 10 seconds to consult the intervention system and 30 seconds to communicate among the team members. For comparison, movement to an adjacent grid cell in perimeter security takes 60 seconds, planting a bomb takes 60 seconds, and a firefight typically lasts at least 5 minutes.

In this experiment, we randomly varied the arrival time and position of the enemy force at four different strength levels. We ran 20 random trials at each enemy strength level. We expected that teams using the goal assignment structures would be more responsive as the structures change less

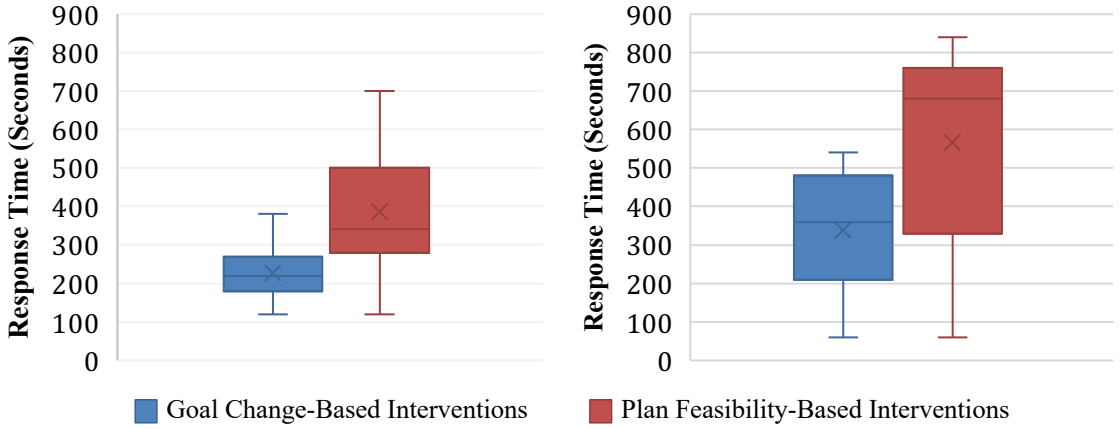

Figure 3. Team responsiveness distributions to (left) first and (right) second complication (lower is better).

frequently than plans. We also hypothesized that this would indirectly improve team performance, as faster responses are likely to be more effective in the domain.

Claim: Goal change-based crisis recognition improves team responsiveness to complications.

In each trial of the perimeter security experiment, the blue forces (eventually) respond to an initial sighting of a red force team by challenging them and the red force team would either leave or fight. We considered each of these events (red force sighting and red force response to challenge) as a complication and measured responsiveness from the start to resolution of the crisis. The first complication, appearance of an enemy, is resolved when the enemy is challenged. The second complication, enemy aggression against blue forces, is resolved when the enemy dies. In some cases, one system variant or the other failed to address a complication by failing to catch up to the red force, dying to red force attacks, or letting the red force escape after bombing the base. Figure 3 shows the distribution of team responsiveness under the goal change intervention condition and the plan feasibility intervention condition for each complication; only trials for which both conditions successfully addressed each complication are included. These are typical box plots, with lines indicating quartiles, minima, and maxima, and a box drawn between the first and third quartile, with the mean value indicated by an "X". These distributions include ten samples of performance at each of the four strength levels.

Under the goal change-based intervention condition, the team averaged a 226 second response time for successful responses to the first crisis and 338 seconds for the second. The plan infeasibility-based intervention condition was measured at 386 seconds (first complication) and 566 seconds (second complication). We compared each pair of distributions using a two-tailed paired sample t-test and found that greater responsiveness in the goal change condition was significant at the .01 level in both cases.

With respect to unsuccessful responses, we analyzed the average rate at which each complication was successfully addressed. Under the goal change-based interventions, response to both complications was successful 95 percent of the time. Plan feasibility-based interventions were successful 85 percent and 35 percent of the time at responding to the first and second complication,

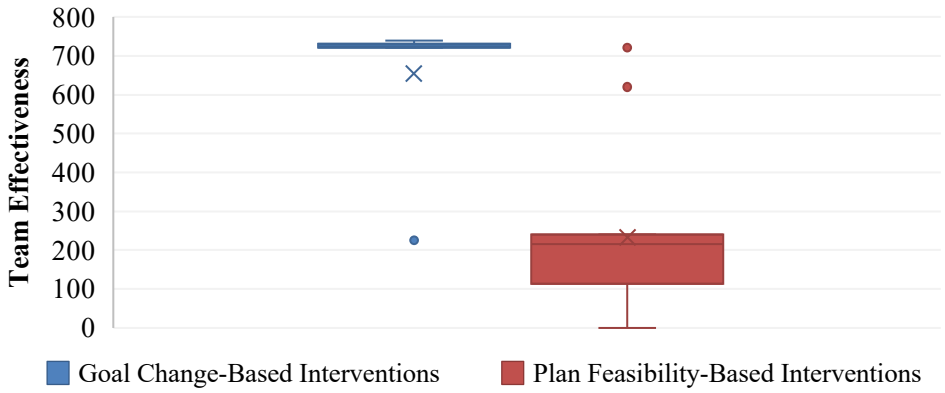

Figure 4. Team effectiveness in perimeter security (higher is better).

respectively. For the second complication, responses under the goal change-based condition were successful more frequently according to a two-tailed paired sample t-test at the .01 level. However, for the first complication the difference between success rates was not statistically significant.

Claim: Goal change-based crisis recognition improves team effectiveness.

States of both domains were stored in relational databases. We successfully demonstrated the goal-change-based interventions and showed that they improved performance relative to plan infeasibility-based interventions in both domains.

For the perimeter security domain, we measured team effectiveness based on three elements, in order of their performance. First, was the base destroyed? Second, how many blue subteams were incapacitated? Third, how much damage was sustained by the blue forces? We employed a scoring metric that provided three rewards:

- 500 points for keeping the base safe;
- 100 points for each blue force subteam still standing;
- 1 point for each unit of damage sustained by blue forces.

Figure 4 shows the distribution of effectiveness values measured during the perimeter security experiment. The results were very different; goal-based interventions resulted in an average effectiveness score of 654.9 and plan feasibility-based interventions an average effectiveness of 233.0. Again, horizontal lines indicate the minimum, maximum, and quartiles; a box shows the interquartile range. The mean of each distribution is shown with an "X". In this plot, outliers are data points that fall outside the interquartile range by more than a factor of 150 percent and are shown as dots; we did not include these outliers when calculating minima and maxima. The difference between the effectiveness scores occurred because the team's slower response time when using the baseline intervention system frequently allowed the base to be destroyed. The differences are statistically significant at the .01 level. Figure 5 breaks down the results by enemy strength. It is clear from the figure that goal change-based interventions performed well even versus higher levels of enemy strength, but plan feasibility interventions were negatively affected.

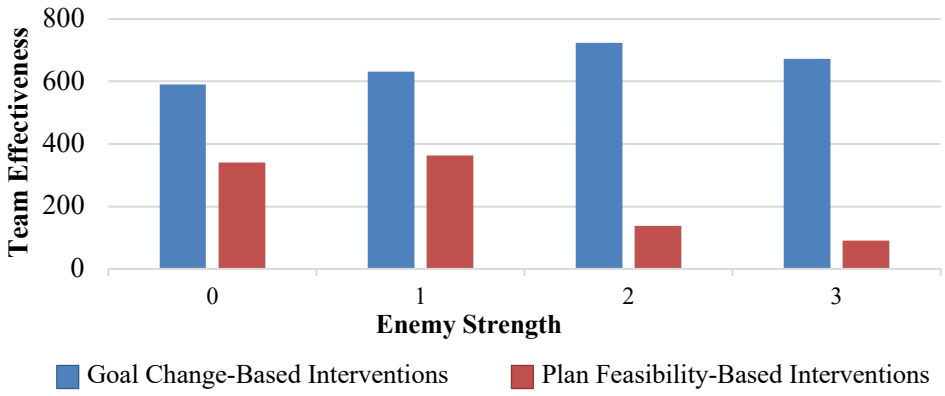

Figure 5. Team effectiveness vs. enemy strength in perimeter security (higher is better).

5.2 Urban Combat Results

In the urban combat domain, we have not carried out a controlled experiment, but we have demonstrated an intervention system using the team crisis intervention skeleton with goal change-based crisis recognition, GASR intervention generation, and a trivial state estimator. We examined performance of this intervention system in a hostage situation, both with and without an ambush. We compared the effectiveness of the team with and without intervention. In this hostage situation (Figure 6 left), three fire teams (subunits of the main team) are sent out to rescue hostages. The simulated team decides to send two support fire teams into nearby buildings K1 and K3 to generate supporting fire against K2, where hostages are held. This lets the third team assault K2 at lower risk. When an ambush occurs (Figure 6 right), blocked doors on nearby buildings force the soldiers further north until they are in a firing line from an ambush point in building A5.

Assisted by goal change-based interventions, our simulated team recognizes this crisis, performs reconnaissance, and succeeds in its mission with only one casualty (to an enemy in building L1). Without intervention, the team is ambushed and three soldiers die before the team manages to retreat, abandoning its mission. We view this performance improvement in the face of surprise as evidence that goal change-based interventions improve team effectiveness.

Claim: GASR is more scalable than hierarchical planning for team crisis intervention.

To assess this claim, we modelled teams using the GASR intervention generator and the GDP intervention generator to make decisions. Teams were modeled as faithfully and accurately updating their goals and plans based on generated interventions. We conducted a quantitative comparison of the number of search nodes used by the GASR and GDP systems across the length of a trial in each domain: perimeter security and urban combat. Figure 7 shows the results.

In both domains, the effort to generate a goal structure varies only slightly based on the difference between initial state and goal state. The effort for GDP was proportional to the number of actions, whereas GASR's effort was proportional only to the complexity of team interactions. This is because the latter relies on humans to perform heuristic planning and reactive policy execution;

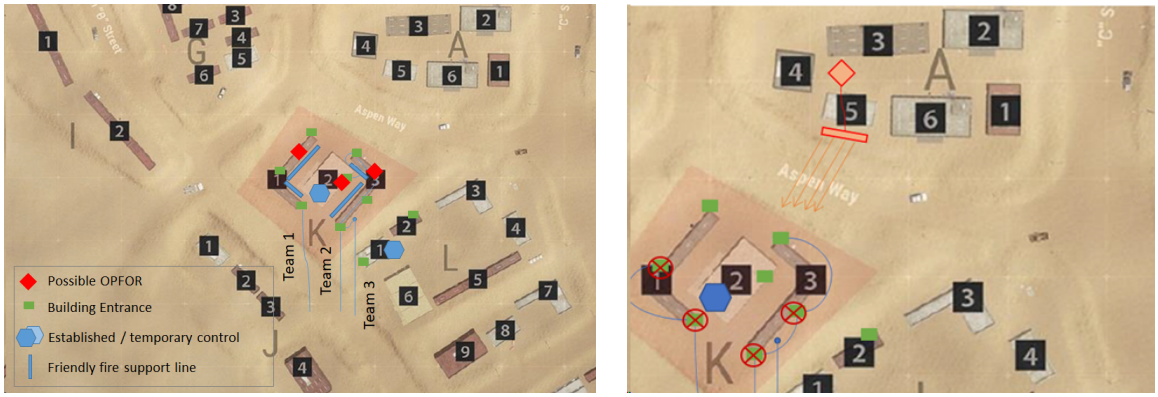

Figure 6. A hostage rescue plan in urban combat (left) and an ambush complication (right).

both of these are tasks that include long sequences of repetitive action and are inefficient to plan. With these demands removed, GASR is required to perform less repetitive high-level reasoning.

In summary, the evidence supports our claims, so the simple goal change-based crisis recognizer presented appears to be a reasonable baseline against which to compare future work on team crisis recognition. This also indicates that goal assignment structures offer a practical alternative to hierarchical plans for team goal achievement. However, we made various uncertain assumptions about human behavior, so additional work should evaluate these claims with humans in the loop.

6. Related Research

Our approach is most closely tied to research on goal reasoning, in that it uses goals to recognize team crises and to intervene. Our state estimation, crisis recognition, intervention generation loop is heavily inspired by the goal-driven autonomy loop (Molineaux et al., 2010) of detecting discrepancies, explaining them, formulating goals, and managing goals. However, we place first the state estimation step, which is analogous to explanation in goal-driven autonomy. This reflects our view that an agent's explanation of the world must be updated even when no problem exists. Prior work on goal formulation focuses primarily on control of simulated actors and autonomous vehicles (Aha, 2018), rather than the goals of teams and team members, as we do here. In the goal reasoning literature, the GRIM system (Johnson, Roberts, Apker, & Aha, 2016) uses the goal life-cycle (Roberts et al., 2014) to control a set of goal reasoning agents engaged in disaster relief. In this work, goal decomposition is a part of the goal reasoning process, but goal expansion results in plans rather than a hierarchical structure. GRIM also does not address goal assignment, precedence, or formal coordination of team members.

The distributed and multi-agent planning community also focuses on teams or organizations, but mostly consider the selection of plans rather than goals. For example, the DOMAPS system and formalism (Cardoso, 2018) is intended for team planning, but it assumes that organizational goals can be directly allocated to team members from the start of the planning process. Coordination protocols then allow for goal assignment, but these are less flexible and more monolithic than the goal assignment structure. In multi-agent planning, issues of goal selection are sometimes handled as part of organizational design. For example, the organization design system KB-ORG (Sims,

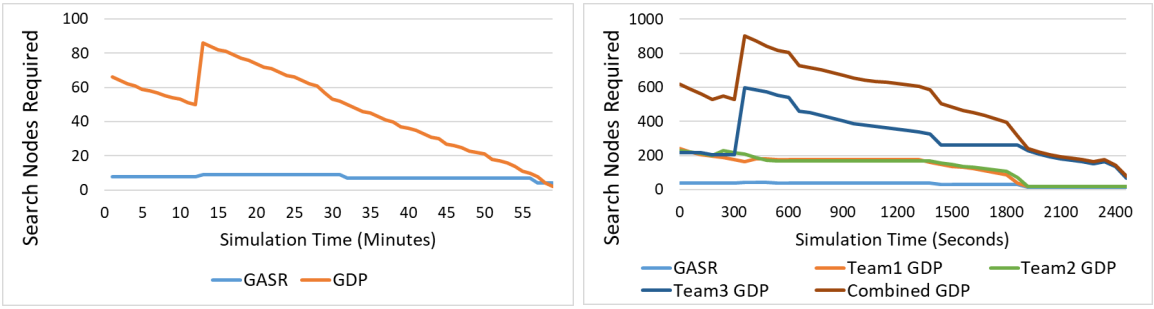

Figure 7. Effort required to intervene in the (left) perimeter security and (right) urban combat domain.

Corkill, & Lesser, 2008) is related to GASR in that it uses hierarchical methods to construct a goal network and to assign goals and roles to individual agents. In contrast, GASR assumes that roles exist already and it selects and assigns goals dynamically in response to crises, rather than at team development time. Many other systems explicitly consider distributed decision making within a team, which is not a feature of this work.

In the multi-agent systems community, *joint intentions* (Cohen & Levesque, 1991) are often used to describe how teams coordinate. This involves agents having the same intention and the mutual belief that they share the intention. Agents generally arrive at such joint intentions through prespecified negotiation protocols or dialogues. In general, the relationship between joint intentions and a higher level is not addressed or is unclear. However, the SharedPlans formalization (Grosz & Kraus, 2006), much like goal assignment structures, describes partial activities to be performed as a group that are refined over time. It requires a representation of the mental states of team members that includes individual and group intentions, beliefs, mutual beliefs, and plans. The goal assignment structure representation is simpler because it assumes that all team goals are shared (although assignments and responsibility can be held individually), and our approach does not create complete plans. A SharedPlans representation might provide increased understanding of and fidelity to agents' decisions, perhaps at a greater computational and knowledge engineering cost.

Previous work in the cognitive architectures community on representing teams include the TacAir-Soar effort (Jones et al., 1999), the related STEAM (Tambe, 1997) architecture, and Companions (Forbus, Klenk, & Hinrichs, 2009). These systems may be suited to team crisis intervention, due to their generality and demonstrated capabilities for both teamwork and response to developments in their environments. However, we cannot compare their capabilities directly due to differing representations and structures. TacAir-Soar and Companions do not provide purpose-specific representations for teams, but their existing formalisms have been shown to be sufficient. The lack of such representations may increase the burden on a knowledge engineer to provide sufficient team information. In contrast, STEAM uses a team goal achievement representation based on joint intentions and SharedPlans. The system creates plans via an operator hierarchy, which is similar to a goal assignment structure in that operators are assigned to subteams or individuals; however, it differs in that operators correspond to tasks rather than goals, which limits failure recovery, and the operator hierarchy does not include precedences. As STEAM does not support failure detection and recovery, it is not appropriate for team crisis intervention.

7. Conclusion and Future Work

In this paper, we addressed the task of monitoring and intervening in a team goal achievement process, which we referred to as the team crisis intervention problem. We defined this task formally, presented a skeleton solution, and compared several systems produced by integrating component solutions with that skeleton. As part of the crisis recognition and intervention generation components, we introduced GASR, a novel method for goal assignment structure refinement. We provided evidence that interventions using GASR improve team responsiveness and team effectiveness. We also showed evidence that team crisis intervention using goal assignment structures is more scalable than that using plans. However, this evidence is still preliminary because our studies use estimates of human reactions that have not been verified empirically. Still, these evaluations point the way to further examination with additional domains.

Future work should focus on more comprehensive methods for intervention generation to improve team effectiveness, crisis recognition research to generate higher rates of accuracy through team modelling, and incorporation of plan understanding in crisis recognition to better recognize team activities. Finally, it should explore dynamic run-time determinations of goal assignability, which should simplify knowledge requirements for the intervention process and increase the flexibility of solutions it produces.

Acknowledgements

The work was supported by AFOSR contract FA2386-17-1-4063, by ONR grant N00014-18-1-2009, and by DARPA contract N6600118C4039. We thank Danielle Brown for her comments.

References

Aha, D. W. (2018). Goal reasoning: Foundations, emerging applications, and prospects. *AI Magazine*, *39*, 3–24.

Aha, D. W., Cox, M. T., & Muñoz-Avila, H. (Eds.) (2013). *Goal reasoning: Papers from the ACS workshop* (Tech. Rep. CS-TR-5029). Department of Computer Science, University of Maryland, College Park, MD.

Bowden, M. (1999). *Black Hawk down: A story of modern war*. Berkeley, CA: Atlantic Monthly.

Cardoso, R. C. (2018). *A decentralised online multi-agent planning framework for multi-agent systems*. Doctoral dissertation, Postgraduate Computer Science Program, Pontifícia Universidade Católica Do Rio Grande Do Sul Escola Politécnica, Porto Alegre, Brazil.

Cohen, P. R., & Levesque, H. J. (1991). Teamwork. *Nous*, *25*, 487–512.

Cox, M. T. (2017). A model of planning, action, and interpretation with goal reasoning. *Advances in Cognitive Systems, 5*, 57–76.

Dix, J., Muñoz-Avila, H., Nau, D. S., & Zhang, L. (2003). IMPACTing SHOP: Putting an AI planner into a multi-agent environment. *Annals of Mathematics and AI*, *37*, 381–407.

Forbus, K.D., Klenk, M., & Hinrichs, T. (2009). Companion cognitive systems: Design goals and lessons learned so far. *IEEE Intelligent Systems*, *24*, 36–46.

Ghallab, M., Nau, D., & Traverso, P. (2016). *Automated planning and acting*. Cambridge, UK: Cambridge University Press.

Grosz, B. J., & Kraus, S. (1996). Collaborative plans for complex group action. *Artificial Intelligence*, 86, 269–357.

Hawes, N. (2011). A survey of motivation frameworks for intelligent systems. *Artificial Intelligence, 175*, 1020–1036.

Johnson, B., Roberts, M., Apker, T., & Aha, D. W. (2016). Goal reasoning with information measures. *Proceedings of the Fourth Conference on Advances in Cognitive Systems.* Evanston, IL: Cognitive Systems Foundation.

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine*, *20*, 27–41

Klenk, M., Molineaux, M., & Aha, D. W. (2013). Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence*, *29*, 187–206.

Larue, O., Juvina, I., Molineaux, M., Howard, B., Nichols, E., Minnery B., & Cox, M. (2019). Team coordination in homogeneous and heterogeneous teams. *Proceedings of the International Conference on Social Computing, Behavioral-Cultural Modeling, & Prediction and Behavior Representation in Modeling and Simulation.* Washington, DC.

Molineaux, M. (2017). *Understanding what may have happened in dynamic, partially observable environments*. Doctoral dissertation, Department of Computer Science, George Mason University, Fairfax, VA.

Roberts, M., Vattam, S., Alford, R., Auslander, B., Karneeb, J., Molineaux, M., Apker, T., Wilson, M., McMahon, J., & Aha, D.W. (2014). Iterative goal refinement for robotics. *Working Notes of the Planning and Robotics Workshop at ICAPS*. Portsmouth, NH: AAAI Press.

Shivashankar, V. (2015). *Hierarchical goal networks: Formalisms and algorithms for planning and acting*. Doctoral dissertation, Department of Computer Science, University of Maryland, College Park, MD.

Shivashankar, V., Alford, R., & Aha, D. W. (2017). Incorporating domain-independent planning heuristics in hierarchical planning. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (pp. 3658–3664). Palo Alto, CA: AAAI Press.

Tambe, M. (1997). Towards flexible teamwork. *Journal of AI Research*, *7*, 83–124.

Sims, M., Corkill, D., & Lesser, V. (2008). Automated organization design for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, *16*, 151–185.

Weiss, G. (2013). *Multiagent systems* (2nd Ed.). Cambridge, MA: MIT Press.

Whipple, J. D. (2015). *Crash course: The decisions that brought down United Flight 173*. Master's Thesis, Department of English, Portland State University, Portland, Oregon.