ゲーム情報学9－3
（2003．3．24）

レスキューエージェントの協調行動に対するグループ形成アプローチ

浅井 義樹 †，伊藤 暢浩†，江崎 哲也‡，犬塚 信博†，和田 幸一†

あらまし
大規模災害を想定したロボカップレスキューにおけるマルチエージェントシステムは，単独では解決できない問題が同時に複数発生するなどの特徴を持つ環境において協調行動を実現することが求められている．そこで本研究では，グループの形成によって，エージェントの協調行動を実現する協調モデルを提案する．ここでグループとは共通の目標達成を目指すエージェントの集合であり，提案するグループ形成アルゴリズムによって，エージェントは必要に応じたグループ形成が可能となる．また提案手法をロボカップレスキューのエージェントに実装して，有効性を検証した．

A Group Forming Algorithm for Cooperative Behavior of Rescue Agents

Yoshiki Asai †, Nobuhiro Itoh†, Tetsuya Esaki‡, Nobuhiro Inuzuka† and Koichi Wada†

abstract A multi-agent system (MAS) is used as a simulation architecture in RoboCupRescue simulation project, which is a project for large scale disaster. It simultaneously occurs several large problems which can't be solved by one rescue agent (robot) in a disaster-stricken district. So, agents must solve such a complicated problem cooperatively. In this paper, we propose a group forming algorithm for cooperative behavior of rescue agents. We define group as a set of agents that solve a same problem cooperatively. By using this algorithm, agent can form a group whenever it is required to solve the problem. We implemented our rescue agent and confirmed efficiency of our algorithm through several simulations.

1 はじめに

近年，世界各地で発生している大規模な自然災害に対する取り組みに注目が集まっている．そのような取り組みの一つに RoboCupRescue[1, 2, 3] がある．中でも RoboCupRescue シミュレーションプロジェクトでは，仮想的に地震を発生させ，それにともなって発生するいくつかの災害をシミュレーションする災害シミュレータについて研究をおこなっている．このような大規模災害の取り組みに対して，マルチエージェントシステムによるアプローチが注目されている．マルチエージェントシステムは，複数のエージェントが協調して活動をおこなうことで，効率的に問題解決をおこなうことができる．これまでおこなわれてきたマルチエージェントの協調行動の実験環境として RoboCup Soccer シミュレーション [4] などがある．ここでは複数のエージェントによる協調行動に関する研究が，サッカーにおける組織的な攻撃や守備を実現することによっておこなわれている．

しかし，本研究が対象とする RoboCupRescue のような大規模災害の現場は，これまでのマルチエージェントシステムが扱っていない，以下に示す問題を満たす協調モデルが求められている．

- 問題の難しさが変化し続ける
- 問題の解決に緊急性が求められる
- 同時に複数の問題が発生する
- 単独では解決できない問題が発生する
- 問題解決に必要なエージェント数が固定ではない

そこで本研究では，グループを導入して，上記の問題を含む環境においても効率的に協調行動をおこなうことができる協調モデルを提案する．ここでグループを形成することで，エージェント単体では解決できない問題が解決でき，また問題のサイズに応じたグループを作成できる利点がある．本研究におけるグループとは，共通の目標達成を目指す複数の

†名古屋工業大学 / Nagoya Institute of Technology
‡松下電器産業株式会社 / Matsushita Electric Industrial Co.,Ltd.

エージェントの集合である．グループは，グループを形成する権限を持つリーダエージェントとそれ以外のメンバエージェントにより構成される．これらのエージェントが協力しあって，共通の目標達成を目指して活動していく．

これまでのグループ形成による協調活動の実現に関する研究には Joint Responsbility[5] と STEAM[6] がある．しかし，これらの手法を大規模災害環境に適用するにはいくつかの問題点がある．Joint Responsibility では，時間による制約を強く与えてしまうために，問題が変化し続ける環境では正しいグループ活動がおこなえない．また行動手順について約束を交わすには多くの時間とメッセージ通信が必要となり，緊急性を要する問題に迅速に対応できない．また STEAM では，問題ごとにリーダとなるエージェントをあらかじめ決めておく必要があり，協調が必要な問題に対して自由にグループを形成できない．さらにメンバーにグループへの参加拒否権がないので，問題が同時に複数発生する環境に柔軟に対応できない．

そこで本研究では大規模災害における救助活動において，エージェントが他のエージェントと適切に協調して活動をおこなうための協調モデルを提案する．本研究が提案する協調モデルの特徴は，グループによる協調活動が必要と判断したエージェントがリーダとしてグループ形成要求ができることである．それにより，複数のエージェントが同じ目標に対して同時にグループ形成要求をおこなう可能性があるが，そのような場合でも適切にリーダ選択をして，グループ形成をおこなうことができる．また，各メンバーの参加応答に基づくグループ形成アルゴリズムであり，メンバーは参加拒否することが可能である．したがって，各エージェントが最も優先的に解決すべきと判断した目標に常に取り組むことができる．また，グループのリーダやメンバが故障しても，災害救助エージェントとしての活動を継続していけるように設計している．

本研究では RoboCupRescue シミュレーションシステムにおける消防隊エージェントに提案手法を適用して実験をおこなった．その結果，提案手法が与えたグループ化アルゴリズムが必要なサイズのグループを形成し，そのグループが正しく機能していることを示すことができた．また，Joint Responsibility や，STEAM の特徴をもつエージェントを構築して，協調活動の能力について比較実験をおこなった．その結果，道路閉塞によってエージェントの動きが大幅に制限される条件を除いては，提案手法が最も協調行動ができていることがわかった．

2 諸定義

2.1 タスクの定義

タスクとは，エージェントがおかれている環境で発生している解決すべき問題である．タスクはサイズをもち，例えばそのタスク解決に必要なエージェント数などで定義される．またタスクには状態が存在しており，エージェントの活動によってタスクの状態は変化する．エージェントは，タスクの状態をできるだけ短時間で終結状態にすることが目標である．さらにタスクには，存在する空間内の位置に関する情報が存在する．エージェントはタスク情報としてこの位置情報を交換することによって，タスクの特定をおこなうことができるものとする．なお存在するタスクの数やサイズ，状態は時々刻々と変化する．

2.2 対象とするエージェント モデル

本研究が対象とするエージェントモデルは，RoboCupRescue における災害救助エージェントのエージェントモデルに準じている．以下にそのモデルを示す．

- エージェントは固有の id を持っており，他のエージェントを区別できる．
- エージェントは同期して動く．
- エージェントは 1 ステップに問題解決か，移動のどちらかをおこなうことができる．
- 移動速度は周囲の環境により変化する (例：渋滞など)．
- エージェントは故障することがある．
- 視界には制限がある．
- エージェント間通信においてメッセージの送受信回数，メッセージ長に制限がある．
- 複数のエージェントが同時に無線通信をおこなっても，受信制限メッセージ数以内なら正しく受信できる．
- エージェントはタスクのサイズを正しく観測できる．
- エージェントのタスク解決能力には制限があり，その能力よりも大きな問題が存在することがある．

2.3 グループの定義

本研究では，エージェントが協調行動するための手段としてグループを形成する．ここでグループとは「共通の目標達成を目指すエージェントの集合」とする．エージェントは必要に応じてグループを形成し，共通の目標を達成するか，その目標が達成不能となるまでグループによる活動をおこなう．

グループはリーダとメンバによって構成される．リーダはグループを形成したり，解散する権限を持つエージェントである．メンバはグループに属するリーダ以外のエージェントである．本研究におけるグループを以下に定義する．

定義 1 (グループの定義)

$$Group = < cooperative\_target, leader, members >$$

ここで $cooperative\_target$ は，このグループが協調して解決を目指しているタスクの情報を表している．そして $leader$ はグループのリーダを表し，$members$ はこのグループの活動に参加しているメンバの集合を表している．グループを形成している $leader$ と $members$ は，$cooperative\_target$ に含まれているタスクの位置情報などから共通の目標であるタスクを特定して，そのタスクの解決を目指して協調して活動をおこなう．その活動の間，リーダとメンバは定義 1 の情報を保持し続けなければならない．なお，本研究におけるグループの形成・維持などは，全てエージェント間の通信によっておこなわれる．よって，リーダと全てのメンバは相互に通信できることが必要である．

3 協調行動の設計

3.1 エージェントの行動

本研究におけるエージェントの活動状態の遷移を図 1 に示す．エージェントは，目標を選択し，その目標達成のために問題解決をおこなうという一連の行動を繰り返す．ここでエージェントのタスク解決能力には制限があるので，単独の能力では解決できないタスクが存在する．そこで目標達成のために協調を必要とする場合には，グループの形成をおこなって他のエージェントとの協調作業で問題解決を目指す．以降，それぞれについて詳しく説明する．

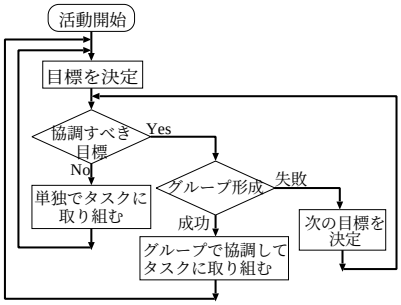

図 1: エージェントの活動状態の遷移

3.1.1 目標の決定

エージェントは複数発生しているタスクの中から，目標とするタスクを 1 つ選ばなくてはならない．そのための判断基準として，タスクの優先度 (Priority) を導入する．各エージェントは，タスクに優先度を与えるための優先度関数を持つものとし，エージェントは優先度の高いタスクから問題を解決する．

本研究が対象とする環境ではタスクの数やサイズが時間とともに変化するので，タスクの優先度も適切に変化するものと仮定する．また，タスクの優先度は目標の決定時のみ使用するものとする．

3.1.2 協調すべき目標かの判断

エージェントが決定した目標が単独で解決できない場合，その目標は「協調すべき目標である」という．エージェントは，次の定義 2 にしたがって，目標とするタスクが協調すべき目標かを判断する．

定義 2 (目標 T が協調すべき目標かの判定)

$$Need\_Cooperation(T) = \begin{cases} true & Ability(T) < Requirement(T) \\ false & Ability(T) \geq Requirement(T) \end{cases}$$

ここで $Ability(T)$ は，エージェントが目標とするタスク T に対する自分自身のタスク解決能力を表す．また $Requirement(T)$ は，エージェントが観測によって認識したタスク T のサイズを表す．

3.1.3 単独で解決可能と判断した場合

目標が単独で解決可能なタスクだと判断したエージェントは，その目標を達成するための活動を単独でおこなう．目標としているタスクの状態が終結状態になっておらず，かつ単独で解決可能である間はその活動を継続する．そして目標タスクの状態が終結状態になった場合や，単独で解決できなくなった場合は新たに目標の決定をおこなう．ここで協調すべき目標でない T への単独活動の継続を判定する関数を次の定義 3 に示す．

定義 3 (目標 T に対する単独活動の継続判定)

$$Keep\_Activity(T) = \begin{cases} true & State(T) \neq Goal\_State(T) \text{ かつ} \\ & Need\_Cooperation(T) = false \\ false & State(T) = Goal\_State(T) \text{ または} \\ & Need\_Cooperation(T) = true \end{cases}$$

ここで $State(T)$ は，エージェントが目標とするタスク T の現在の状態を参照する関数である．また $Goal\_State(T)$ は，タスク T の終結状態を参照する関数である．

3.2 協調すべき目標と判断した場合

エージェントが選択した目標が「協調すべき目標」である場合は，他のエージェントと協調活動をおこなうためにグループの形成をおこなう．本研究が提案するグループ形成手法は，次に示す特徴を持つ．

- 協調活動が必要と判断したエージェントがリーダとしてグループ形成できる．
- 同じ目標に対して、同時にグループ形成した場合、適切に競合を回避できる．
- メンバはグループへの参加を拒否でき，リーダとメンバの同意に基づいたグループ形成をおこなう．
- リーダエージェントが故障しても，メンバは適切な行動を維持できる．

本研究が提案するアルゴリズムでグループ形成するときの流れを以下に示す．このグループ形成アルゴリズムはグループ形成要求の送信，参加応答，グループ生成通知，グループ活動の 4 つのステップによって構成されている．以降，それぞれについて説明する．

Step 1. 周囲のエージェントにグループ形成要求 (目標に選んだときの優先度を含む) を送信．以降このエージェントをリーダと呼ぶ．
Step 2. グループ形成要求を受信した各エージェントは，提案のあった目標を協調して達成すべきかをリーダの優先度や id によって判断し，参加する場合は参加メッセージを応答する．以降これらのエージェントをメンバと呼ぶ．なお同一の目標に対してグループ形成要求したリーダは，受信したグループ化要求の方がタスクの優先度が高い場合や，タスクの優先度が同じで受信したリーダの id が自分より小さい場合，そのリーダに参加応答をおこなう．
Step 3. リーダは，参加応答のあった複数のエージェントと自分の能力によって，協調すべき目標が達成可能の場合は，グループ生成を全てのメンバに送信する．達成不可能の場合はグループ形成が失敗に終ったことを通知する．
Step 4. グループ生成メッセージを受信したメンバは，提案のあった目標を自分のものとして，リーダと他のメンバと共にグループによる行動をおこなう．

3.2.1 グループ形成要求の送信 (Step 1.)

グループ形成要求を送信するエージェント (リーダ) は，その目標を選択した理由をメッセージに付加しなければならない．なぜなら，グループ形成要求を受信したエージェントは，そのグループに参加するかどうかを決めなければならず，そのための判断基準が必要となるからである．本研究のエージェントは，優先度に基づいて目標タスクの選択をおこなっているので，そのタスクを目標として判断したときの優先度を送信する．また，グループ形成要求を受信したエージェントがタスクを特定できるように，タスクの位置などのタスク情報も送信する．

リーダはグループ形成要求を提案したのち，他のエージェントからの参加応答を一定期間受信する必要がある．協調すべき目標を CT としたとき，リーダが参加応答の受信を継続する条件を次に定義する．

定義 4 (参加応答の受信を継続する条件)

$$Keep\_Receive\_Response(CT) = \begin{cases} true & limit\_time > 0 \text{ かつ} \\ & State(CT) \neq Goal\_State(CT) \\ false & limit\_time = 0 \text{ または} \\ & State(CT) = Goal\_State(CT) \end{cases}$$

ここで $limit\_time$ は参加応答を受付ける制限時間を表している．この値はグループ形成要求を送信したときに設定し，ステップ毎に 1 づつ減少させる．もし参加応答の受信継続中に目標が達成された場合は，それをグループ形成要求メッセージを送ったメンバ全員に通知するものとする．

3.2.2 グループ形成への参加 (Step 2.)

グループ形成要求を受信したエージェントは，そのグループへの参加を検討する．ここでエージェントは，同時に複数のグループ形成要求を受信する場合がある．このような場合，以下のステップによってグループ形成要求に対して適切な参加応答をおこなう．

(1) 受信したグループ形成要求の中で，タスクの優先度が最も高いリーダを選択する．
(2) (1) で最も優先度が高いエージェントが複数存在した場合は，id によるエージェントの優先順位をあらかじめ決めておき，その優先順位にしたがって 1 リーダを選択する．
(3) そのリーダが提案する協調すべき目標のタスクの優先度と自分が取り組んでいるタスクの優先度を比較して，リーダの提案するタスクの優先度が高い場合はそのグループに参加する．

グループ形成要求に対し参加応答するエージェントは，現在の自分の位置の情報を含めて参加応答メッセージを送信する．これはグループ形成要求に対し必要以上にエージェントが集まった場合に，リーダが必要なエージェントを選択するために利用される．

同様に，複数のリーダが同時に同一の目標に対してグループ形成要求をおこなった場合は，そのリーダの中から 1 リーダを真のリーダとしてグループ形成をおこなう．その場合は以下のステップによって適切なリーダを選択する．

(1) 受信したグループ形成要求の中で同じ目標に対するものを選択する．
(2) 選択したグループ形成要求と自分の送信したグループ形成要求の中で，タスクの優先度が最も高いリーダを選択する．
(3) (2) で最も優先度が高いエージェントが複数存在した場合は，id によるエージェントの優先順位にしたがって 1 リーダを選択する．
(4) 選択したリーダが自分でない場合は，そのリーダに対して参加応答をおこなう．

以上のように同一の目標に対して 1 リーダが選択される．また異なるタスクに対するグループ形成要求は同時におこなわれる場合があるが，先に述べたようにより優先度が高いタスクに対して参加応答がおこなわれる．したがって自分の優先度よりも協調すべき目標の優先度の高いエージェントの参加応答が 1 リーダに集まる．このようにして，次々にグループを形成していく．

本研究では，一旦決定したタスクの優先度はそれを解決するまで変化しないものとする．したがって，一定時間参加の意志を継続する間に，自分が取り組むタスクの優先度が変化することはない．また，一定時間経過するまでに他のグループ形成要求を受信した場合には，参加応答をおこなわない．

3.2.3 グループ生成の判断 (Step 3.)

リーダは定義 4 に基づく制限時間内に参加応答したメンバで協調すべき目標が達成可能かどうかの判断をおこなう．ここで，候補リストに含まれるエージェントとリーダによる仮のグループを θ' とし，提案した協調すべき目標を CT とすると，リーダは次の定義 5 によって，グループ生成の判断をおこなう．

定義 5 (グループ生成判定関数)

$$Form\_Group(\theta', CT) = \begin{cases} true & Group\_Ability(\theta', CT) \geq Requirement(CT) \\ false & Group\_Ability(\theta', CT) < Requirement(CT) \end{cases}$$

ここで $Group\_Ability(\theta', CT)$ は，仮のグループ θ' のタスク CT に対するタスク解決能力を表す．

リーダが送信するグループ生成メッセージには，メンバの id に関する情報が含まれている．協調すべき目標の達成に必要なエージェント数以上に多数のメンバが集まった場合は，参加応答に含まれる位置情報を利用して適切なエージェントを選出し，そのエージェントをメッセージに付加する．この情報によって，参加応答したメンバは自分がグループのメンバとして認められたかを判断する．

3.2.4 グループ生成の完了 (Step 4.)

参加応答したメンバは，リーダからのグループ生成メッセージの中に自分の id が含まれているかを調べる．もし含まれていたときは，提案のあった協調すべき目標を自身の目標に設定して，リーダと他のメンバと共に協調活動をおこなう．もし自分の id が含まれていない場合は，そのグループには必要とされていないと判断し，自分が保持している目標達成に向けて活動を継続していく．

3.3 グループ活動の維持

グループを形成した後，リーダとメンバは必要に応じて活動に関する情報や行動手順などをコミュニケーションし，グループとして効率的な活動をおこなっていく．

グループによる活動の結果，タスクが解決される場合もあるが，タスクがより複雑になる場合もある．このような場合，グループの解散や離脱をおこなう．ここでリーダがグループを解散する条件とメンバがグループから離脱する条件は基本的に同じとすべきである．なぜなら，リーダが故障した場合にグループが解散されず，目標を達成してもメンバがグループに残り続けるからである．このようにメンバもグループ活動の状況を判断することで，グループの耐故障性を実現している．その条件を定義 6 に示す．

定義 6 (グループを解散・離脱する条件)

$$Breakup(Release)\_Group(\theta, CT) = \begin{cases} true & State(CT) = Goal\_State(CT) \text{ または} \\ & Group\_Ability(\theta, CT) < Requirement(CT) \\ false & State(CT) \neq Goal\_State(CT) \text{ かつ} \\ & Group\_Ability(\theta, CT) \geq Requirement(CT) \end{cases}$$

定義 6 は，グループによる活動によって協調すべき目標が達成された場合にグループを解散・離脱することを示している．また，グループ活動をおこなったにも関わらずタスクがより大きくなってしまった場合や，メンバの離脱によってグループの問題解決能力が落ちた場合には，現在のグループで目標は達成不能と判断し，グループを解散・離脱する．

さらに，メンバはグループの一員になったものの，協調すべき目標に貢献できない可能性がある．例えばエージェントの移動能力の制限によって，目標のタスクを解決するために必要な移動ができない場合もある．このような目標に貢献できるかどうかの具体的な判断は扱うタスクに依存するが，貢献できないと判断した場合にはそれを全メンバに伝え，グループを離脱する．

4 実験

本研究で提案した協調モデルをロボカップレスキューシミュレータにおける消防隊エージェントに実装した．このシミュレータでは，実時間の 1 分を 1 ステップとして，地震によって発生する火災などの災害をシミュレーションする．シミュレータのバージョンは 0.40 を使用した [1]．

構築したエージェントを用いたシミュレーションにより，提案手法が実際に正しくグループを形成できるかを評価する．また Joint Responsibility や STEAM などのグループ形成アプローチに対して，グループの形成回数と形成率，グループ形成とその活動，総合的な結果，協調行動を実現するために利用した通信回数について評価をおこなう．

4.1 実験の条件

実験で設定した災害空間の条件やエージェントの能力について，以下に説明する．

地震発生直後のエージェントの活動

地震発生から一定時間，エージェントの活動を停止させた．なぜなら，出火直後の火災はいかに早く火災現場に到着するかが重要となり，協調活動は必要とされないからである．そこでエージェントの活動停止によって火災をある程度延焼させ，協調活動が必要な状況を再現した．

メッセージの受信回数の制限

グループのサイズに支障がでないようにするため，各エージェントは 1 ステップに 80 バイトのメッセージを，フィールドにいるエージェント数 (例：15 隊) 受信できることにした．

活動するエージェント

実験ではエージェントの消火活動における協調行動に焦点を当てる．よって活動するエージェントは消防隊エージェントと，道路閉塞の啓開にあたる警察隊エージェントのみとした．なお警察隊エージェントは全実験で同一のエージェント (NITRescue02) を利用した．

上記のような条件のもとで仮想都市と呼ばれるマップ (図 2) で実験をおこなった．仮想都市マップは国土技術政策総合研究所の高橋らによって作成された架空の都市である [7]．この都市は神戸市長田区の特徴に基づいており，大都市中心部の幹線道路に面した商業・住宅混合地域を想定して作られている．図 2 に示した地図は 400m 四方であり，その区域に存在する建物は 1,281 棟である．

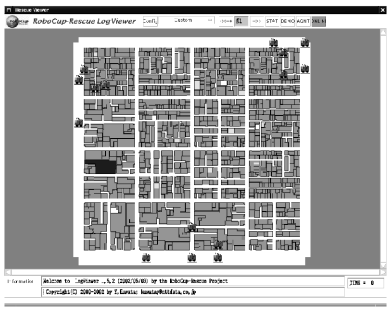

図 2: 仮想都市の概観図

実験では，消防隊エージェント 15 隊，警察隊エージェント 14 隊，初期出火建物 13 棟，初期停止ステップ 15 ステップ (実時間の 15 分)，地震の規模は RoboCup2002 競技会において優勝した Arian チームが作成したデータ (地表面加速度分布データ) を用いた．エージェントの初期位置と初期出火する建物，地震の規模を同じ条件にして 50 回のシミュレーションをおこなった．なぜなら建物の倒壊や火災の延焼は毎回異なるからである．

4.2 実験結果

4.2.1 グループの形成回数と形成率

グループ形成回数と形成率に関する実験結果を表 1 に示す．なお，表に示す値は 50 回のシミュレーションの平均値である．

表 1 から，提案手法が最も多くのグループ形成要求をしていることがわかる．これは他の手法と違い，協調活動が必要と判断した全てのエージェントがグループ形成要求できるからである．しかし，グループ形成率は低くなっている．ここで提案手法が形成するグループのサイズの平均は 6.35 隊であった．つまり最悪の場合，1 グループ形成のためにそれだけのグループ形成要求が発生する．よってグループ形成率は低くなったが，他の手法に比べて積極的にグループ形成要求をおこなうことができ，それにより多くのグループが形成できていることがわかる．

表 1: 各手法のグループ形成回数と形成率

手法	グループ形成要求回数	グループ形成回数	形成率 (%)
提案手法	92.12	24.92	27.05
Joint Responsibility	19.22	10.02	52.13
STEAM	4.34	4.18	96.31

4.2.2 グループ形成と活動

ある 1 回のシミュレーションにおいて，120 ステップから 170 ステップの間に形成したグループのサイズを図 3 に示す．

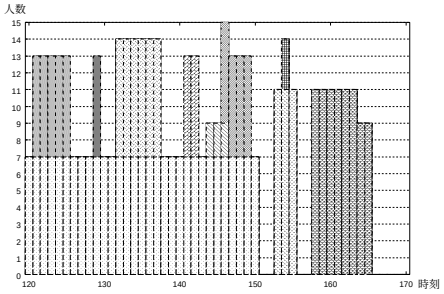

図 3: 形成されたグループのサイズ

図 3 から，ほとんどの時刻で同時に 2 チーム，多い時には 3 チームが形成されている．また 158 ステップで形成されたチームでは，エージェントが現場に到達できず，目標に貢献できないと判断して 2 隊のエージェントが離脱している．その後目標が達成したためにこのグループは解散している．このように，本研究の提案手法は，必要なグループを形成し，適切なグループ活動ができていることがわかる．

4.2.3 総合的な評価

表 2 に，各手法に基づくエージェントによって，延焼を防ぐことができた建物数を示す．これにより，提案手法が協調が必要な場面で適切に協調活動をおこなえたことがわかる．

表 2: 延焼を防いだ建物数

手法	延焼を防いだ建物数
提案手法	1013.70
Joint Responsibility	315.58
STEAM	347.16
協調なし	295.04

4.2.4 通信量に関する結果

各手法が協調行動を実現するためにおこなった通信の回数を表 3 に示す．

表 3 からわかるように，提案手法が最も通信回数が多い．これはグループ形成要求が他の手法と比べて多くなっていることと，グループを解散する際にリーダは解散メッセージを，メンバは離脱メッセー

表 3: エージェントの通信回数

手法	通信回数
提案手法	636.64
Joint Responsibility	126.2
STEAM	66.9

ジを送信することなどによる．不要なグループ形成要求を減らすことなどによって，通信回数を削減することが今後の課題である．

4.3 神戸市長田区における結果

1995 年に発生した兵庫県南部地震で最も大きな被害が発生した当時の神戸市長田区の地図を用いて，本研究の提案手法に基づくエージェントによって実験をおこなった．当時の神戸市長田区では木造家屋が密集し、狭い道路が多いために全域で道路閉塞が発生する．よってエージェントの移動に制約が起きることを考慮し，エージェントの初期位置が地域に分散している場合と集中している場合 (エージェントを 2 組に分けて配置) で実験をおこなった．それぞれの手法が延焼を防いだ建物数について表 4，5 に示す．

表 4 から，エージェントが分散している場合は，ほとんどグループを形成しない STEAM や協調なしの手法の方が結果が良い．これはグループの形成は正しくおこなわれているが，大量に発生した道路閉塞によりエージェントの移動ができず，協調活動がおこなえていないのが原因である．表 5 では，あらかじめエージェントが集まっているために協調活動に移動が必要とされないために良い結果となった．

表 4: 消火した建物数 (初期位置：分散)

タイプ	延焼しなかった建物数
提案手法	206.10
Joint Responsibility	288.62
STEAM	546.32
協調なし	535.96

表 5: 消火した建物数 (初期位置：集中)

タイプ	延焼しなかった建物数
提案手法	440.78
Joint Responsibility	97.46
STEAM	94.20
協調なし	53.92

ここで表 4 の条件においてエージェントの移動が少しでも可能であれば，STEAM や協調なしよりも多くの建物を火災から防ぐことができたと考えられる．このことから，木造家屋が密集した地域での都市計画の見直しが必要であるといえる．これは防災の観点から見れば同然のことであるが，このような考察ができることがロボカップレスキューのもう 1 つの特徴である．今後はエージェントの研究を通して，実際の防災に対する提案もおこなっていきたいと考えている．また，エージェントが移動できない場合の救助戦略の提案を考える必要がある．

5 まとめ

本研究では，これまでのマルチエージェントシステムが扱ったことのない特殊な問題が発生する大規模災害環境において，災害救助エージェントが他のエージェントと協調して活動をおこなうための協調モデルについて提案をおこなった．また，その提案手法を RoboCupRescue シミュレータの消防隊エージェントに実装して，正しくグループを形成できるて機能していることや，他のグループ形成アプローチに対して，有効であることを示した．

今後の課題としてグループの階層構造を導入することや通信回数の削減，道路閉塞が多く発生した場合にも対応できるエージェントの協調モデルの設計などがあげられる．

参考文献

[1] RoboCupRescue Official Web Page, http://www.r.cs.kobe-u.ac.jp/robocup-rescue/

[2] Hiroaki Kitano, "RoboCup Rescue: Search and Rescue in Large-Scale Disasters as a Domain for Autonomous Agents Research", SMC99,1999.

[3] "ロボカップレスキュー", 田所 諭 北野宏明監修,RoboCup-Rescue 技術委員会 The RoboCup Federation ロボカップ日本委員会編, 共立出版,2000.

[4] The RoboCup Soccer Simulator, http://sserver.sourceforge.net/index.html

[5] N.R.Jennings, "Controlling cooperative problem solving in industrial multiagent systems using joint intention", Artificial Intelligence,75,1995.

[6] M.Tambe, "Toward flexible teamwork", Journal of Artificial Intelligence Research,7,83-124,1997.

[7] 高橋 宏直 畑山 満則 松野 文俊, "仮想都市の構築と地震災害シミュレータ次期バージョンへの展開", SI2001,235-236,2001.

6　付録．擬似コード

本研究で提案した協調モデルの擬似コードを次に示す.

```
Message-Order(M) is a descending order of
messages in M(M is Messages Queue).The order is
decided with the priority included in a message.
If M includes a few messages with the same
priority,the order is decided by using agentids
included in the messages.

mystatus := free; mytask := 'none';
repeat
  receive M; /* messages queue*/
  observe tasks T;
  observe mytask if mytask is not 'none';

  case mystatus of
  leader:
    if Breakup_Group(mygroup,mytask) then
      send (cancel, myid);
      mystatus := free; mytask := 'none';
    else
      foreach mesage m=(leave, agentid) in M do
        if agentid in mygroup then
          /* メンバの離脱 */
          mygroup := mygroup - {agentid};
      act for mytask as a leader;

  member:
    if there is a message (cancel, agentid) in M
    then
      if (agentid = myleaderid) then
        mystatus := free; mytask := 'none';
    else if Release_Group(mygroup,mytask) then
      send (leave, myid);
      mystatus := free; mytask := 'none';
    else if cannot_contribute to a task then
      /* タスクに貢献できないと判断 -> 離脱 */
      send (leave, myid);
      mystatus := free; mytask := 'none';
    else
      act for mytask with myleader;

  free:
    if there is a message
      (request, agentid, task_info, priority)
    in M then
      let m := the message with the maximum
                order in the Message-Order(M)
        (request, agentid, task_info, priority);
      if (m's priority > priority(mytask)) or
        when mytask = 'none' then
        /* 要請のあったグループに参加する */
        leaderid := m's agentid;
        send(ack, leaderid, myid, mypos);
        mystatus := wait_as_member;
        mygrouptask := task(m's task_info refers);
        leader_priority := m's priority;
        timeleft := a constant time limit;
      else if ((mytask is 'none') or
            (status(mytask) = final))
            and (T is not empty)
        then
          mytask := one of tasks
                  with maximum priority in T;
          if Need_Cooperation(mytask) then
            /* 協調すべき目標であるとき */
            send (reuquest, myid,
                   mytask_info,priority(mytask));
            mystatus := wait_as_leader;
            timeleft := a constant time limit;
            group := emptyset;
          else
            /* 単独で解決可能なとき */
            act for mytask alone;

  wait_as_leader:
    /* グループ形成要求をしたリーダ */
    if there is a message
        (request, agentid, task_info, priority)
      where task_info refers a task mytask in M
    then
      let M' := M and my group forming request;
      let m := the message with the maximum
                 order in the Message-Order(M')
        (request, agentid, task_info, priority);
      /* リーダの競合が発生 */
      if (m's agentid is not myid) then
        /* 優先順位に従う */
        send (cancel, myid);
        leaderid := m's agentid;
        send (ack, leaderid, myid);
        mystatus := wait_as_member;
    else if (timeleft > 0) and there is a message
        (ack, leaderid, agentid, pos) in M then
      if (leaderid = myid) then
        /* 参加応答したメンバをリストに追加 */
        group := group U {agentid};
    else if (timeleft <= 0) and
            (Form_Group(group, mytask))
      then
        /* 集まったメンバで解決可能 */
        mystatus := leader;
        make mygroup by choosing members
                                from group by pos;
        send (confirm, myid, mygroup);
      else
        send (cancel, myid);
        mystatus := free;  mytask := 'none';
    act for mytask;

  wait_as_member:
    /* グループ参加応答したメンバ */
    if (timeleft <= 0) then
      mystatus := free;
    else
      if there is a message
         (confirm, agentid, group) in M then
        if (agentid = myleaderid) then
          if myid in group then
            mystatus := member; mygroup := group;
            mytask := mygrouptask;
          else
            mystatus := free;
    act for mytask;

  timeleft := timeleft - 1;
endrepeat
```