グループ形成によるレスキューエージェントの協調モデルについて

伊藤 暢 浩*・浅井 義 樹**・江崎 哲 也***

犬塚 信 博*・和田 幸 一*

A Cooperative Model of Rescue Agents by a Group Forming Algorithm

Nobuhiro ITO*, Yoshiki ASAI**, Tetsuya ESAKI***,
Nobuhiro INUZUKA* and Koichi WADA*

Multi-agent techniques are expected to develop flexible systems for utilization in environmental situations, which change complicatedly and continuously — especially, rescue systems in large-scale disasters enabling reductions of damage and preventing many deaths. The RoboCupRescue project was started in Japan in 1999. The project is one of international research to develop rescue systems for large-scale disasters. The project uses the multi-agent approach to realize its aims. In this paper, we deal with the RoboCupRescue simulation project.

One of areas of interest in multi-agent techniques is the realization of efficient cooperative behavior design of multi-agent systems. The problems of large-scale disasters can not be solved utilizing these traditional frameworks because rescue agents have mobility and can get only restricted information (just a part of the information). So, in the traditional frameworks, agents cannot communicate with each other and form short time suitable groups. Besides, agents need to behave in a real-time environment that is changing complicatedly and continuously.

In this paper, we propose a cooperative model of rescue agents by a group forming algorithm behavior of rescue agents. We define group as a set of agents that solve a same problem cooperatively. By using this model, agent can form a group whenever it is required to solve the problem. We implemented our rescue agent and confirmed efficiency of our model and algorithm through some simulations.

Key Words: RoboCup Rescue, multi-agent simulation, agent-based approach, cooperative protocol, rescue engineering

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ロボカップレスキュープロジェクト^{1)~3)}がある。中でもロボカップレスキュー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行なっている。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における研究課題の一つに効率の良い協調行動による問題解決があり、その実験環境の一

つとしてロボカップサッカーシミュレーション⁴⁾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行なわれている⁵⁾。

しかし、本研究が対象とするロボカップレスキューのような大規模災害の現場では、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、以下に示す問題を解決できる協調モデルが求められている。

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

そこで本研究では、グループという概念を導入して、上記の問題を含む環境においても効率的に協調行動を行なうことができる協調モデルを提案する。また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もつリーダーエージェントとそれ以外のメンバーエージェントにより構成される。本モデルはつぎのような特徴をもつ。

* 名古屋工業大学 名古屋市

** ジェイアール東海情報システム (株) 名古屋市

*** 松下電器産業 (株) 門真市

* Nagoya Institute of Technology, Nagoya

** JR TOKAI Information Systems Company, Nagoya

*** Matsushita Electric Industrial Co., Ltd., Kadoma

(Received April 1, 2005)

(Revised November 6, 2005)

- (1) グループの形成により、単独のエージェントでは解決できない問題を解決できる(上述の(a)).
- (2) グループの形成により、問題の難しさに応じたグループを作成できる(上述の(b)).
- (3) 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はリーダーとしてグループ形成要求ができる。これにより、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行なう可能性があるが、そのような場合でも適切にリーダー選択をして、グループ形成を行なうことができる(上述の(c)および(d)).
- (4) 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(上述の(c)および(e)).
- (5) グループのリーダーやメンバーが故障しても、災害救助エージェントとしての活動を継続できる(上述の(b)および(c)).

これまでのグループ形成による協調活動に関する主要な研究としては、Joint Responsibility⁶⁾とSTEAM⁷⁾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、問題の変化に合わせたグループ形成・解散が行えない。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、問題ごとにリーダー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。本研究のエージェントモデルは前述の特徴(3)～(5)により、これらの問題に対応できる。

本研究ではロボカップレスキュー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行なった。その結果、提案手法により与えたグループ化アルゴリズムが火災の消火に必要なサイズのグループを迅速に形成・解散し、協調して消火活動を行なっていることを確認することができた。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行なった。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できることが確認できた。

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、たと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、タスクの特定を行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、ロボカップレスキュー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をもっており、ほか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き、同期の単位をステップと呼ぶ。
- エージェントは1ステップに問題解決か、移動のどちらかを行なうことができる。
- 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。
- エージェントの視界には制限がある。
- エージェントはすべてのエージェントと通信できる。
- 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通信を行なっても、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行なう。グループはリーダーとメンバーによって構成される。リーダーはグループを形成したり、解散する権限をもつエージェントである。メンバーはグループに属するリーダー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)

$Group = \langle cooperative_target, leader, members \rangle$

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を表わしている。そして *leader* はグループのリーダーを表わし、*members* はこのグループの活動に参加しているメンバーの集合を表わしている。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行なう。その活動の間、リーダーとメンバーは形成している

グループ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、すべてエージェント間の通信によって行なわれる。よって、リーダーとすべてのメンバは相互に通信できることが必要である。

3. 協調モデルの設計

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 Fig. 1 に示す。エージェントは観測したタスクから、1つのタスクを目標として選択し、その目標達成のために問題解決を行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、グループの形成を行なってほかのエージェントとの協調作業で問題解決を目指す。

Fig. 1 State transition diagram for an agent.

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もつものとし、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もので、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、つぎの定義 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 (タスク 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わす。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 のサイズを表わす。

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行なう。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行なう。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つぎの定義 3 に示す。

【定義 3】 (目標 (タスク 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、ほかのエージェントと協調活動を行なうためにグループの形成を行なう。本研究が提案するグループ形成手法は、つぎに示す特徴をもつ。

- 協調活動が必要と判断したエージェントがリーダー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、リーダーとメンバの同意に基づいたグループ形成を行なう。
- 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は以下の 4 つのフェーズによって構成されている。

フェーズ 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) を送信する。以降このエージェントをリーダーと呼ぶ。

フェーズ 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ーの優先度や 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ーは、受信したグループ形成要求のほう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ーの id が自分の id より優先順位が高い場合、そのリーダーに参加応答を行なう。

フェーズ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、グループ生成をすべ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わったことを通知する。

フェーズ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、リーダーとほかのメンバと共にグループによる行動を行なう。

3.2.1 グループ形成要求の送信 (フェーズ1)

グループ形成要求を送信するエージェント (リーダー) 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ない、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行なっているので、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。リーダーはグループ形成要求を提案したのち、ほか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、リーダーが参加応答の受信を継続する条件をつぎに定義する。

【定義4】 (参加応答の受信を継続する条件)

$Keep_Receive_Response(CT) =$

$$\begin{cases} \text{true} & \text{limit_time} > 0 \text{ かつ} \\ & State(CT) \neq Goal_State(CT) \\ \text{false} & \text{limit_time} = 0 \text{ または} \\ & State(CT) = Goal_State(CT) \end{cases}$$

ここで $limit_time$ は参加応答を受け付ける制限時間を表わしている。この値はグループ形成要求を送信したときに設定し、ステップごとに1つづつ減少させる。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 (フェーズ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、以下の手順でグループ形成要求に対して適切な参加応答を行なう。

- (1) 受信したグループ形成要求の中から、タスクの優先度が最も高い要求を送信したリーダーを選択する。
 - (2) (1) で選択すべき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リーダーを選択する。
 - (3) そのリーダー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、リーダー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、リーダー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、複数のリーダーが同時に同一の目標に対してグループ形成要求を行なった場合は、そのリーダーの中から1リーダーを真のリーダーとしてグループ形成を行なう。その場合は以下の手順で適切なリーダーを選択する。

- (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要求を送信したリーダーを選択する。
- (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リーダーを選択する。
- (4) 選択したリーダーが自分でない場合は、そのリーダーに対して参加応答を行なう。

以上のように同一の目標に対して1リーダーが選択される。また異なるタスクに対するグループ形成要求は同時に行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行なわれる。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リーダーに集まる。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、いったん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ほかのグループ形成要求を受信した場合には、参加応答を行なわない。

3.2.3 グループ生成の判断 (フェーズ3)

リーダーは定義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行なう。ここで、候補リストに含まれるエージェントとリーダー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、リーダーはつぎの定義5によって、グループ生成の判断を行なう。

【定義5】 (グループ生成判定関数)

$Form_Group(\theta', CT) =$

$$\begin{cases} \text{true} & Group_Ability(\theta', CT) \geq Requirement(CT) \\ \text{false} & Group_Ability(\theta', CT) < Requirement(CT) \end{cases}$$

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わす。リーダー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 (フェーズ 4)

参加応答したメンバは、リーダー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、リーダーとほかのメンバと共に協調活動を行なう。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、リーダー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行なっていく。

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、タスクのサイズが大きくなり、タスクが解決できなくなる場合もある。このような場合、グループの解散や離脱を行なう。ここでリーダー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、リーダー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)

$$\text{Breakup}(\text{Release_Group}(\theta, CT)) = \begin{cases} \text{true} & \text{State}(CT) = \text{Goal_State}(CT) \text{ または } \\ & \text{Group_Ability}(\theta, CT) < \text{Requirement}(CT) \\ \text{false} & \text{State}(CT) \neq \text{Goal_State}(CT) \text{ か } \\ & \text{Group_Ability}(\theta, CT) \geq \text{Requirement}(CT) \end{cases}$$

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、グループ活動を行なったにもかか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。たと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. 関連研究

本節ではグループ形成による協調行動の代表研究である Joint Responsibility⁶⁾ および STEAM⁷⁾ について述べる。協調行動の主な理論研究としては文献^{6), 8)~11)}、そして実装も

含めた研究として文献^{7), 12)}がある。そのほとんどの論文で利用されているのが Joint Responsibility であり、その実現を目指した代表例が STEAM である。また近年では協調行動の評価の枠組み¹³⁾や形式化¹⁴⁾も提案されている。

4.1 Joint Responsibility

Joint Responsibility⁶⁾は工業分野のマルチエージェントシステムにおけるエージェントの協調行動のために提案された手法である。Jennings による文献⁶⁾では電力輸送マネジメントを行なうエージェントとしての実装例が示されている。この手法はチームワーク理論である Joint Intention 理論⁸⁾に基づいており、GRATE*(Generic Rules and Agent model Testbed Environment) と呼ばれるエージェントアーキテクチャと、GRATE Distributed Planning Protocol によるグループ形成を提案して、変化をとまなう複雑な環境におけるエージェントの協調行動の設計を可能にしている。

この手法は特に能力の異なる異種エージェントが協調する枠組みとして有用である。また環境の複雑さを考慮しつつ、グループの作成を希望するエージェントの提案のみで協調行動を実現するのではなく、エージェント相互の同意によってグループを形成し、柔軟な協調行動を実現している。しかし、この手法は災害救助のような問題を扱うにはつぎのような問題がある。まず、協調行動が時間の制約を強く受けるため、問題の変化に柔軟に対応できず、問題の変化に合わせて迅速なグループの形成・解散ができないこと。そして行動手順を決定するのに多くのやり取りが必要となるため、緊急性を要する問題に対して迅速に対応できないことである。

4.2 STEAM

STEAM⁷⁾は軍事問題や RoboCup のサッカーエージェントなど、実用的なマルチエージェントシステムにおける協調行動のための手法である。Joint Responsibility と同様、Joint Intention 理論に基づいており、Establish Commitments Protocol⁸⁾を用いてグループ形成による協調行動を行なっている。

STEAM では、個人で行なう行動とグループで行なう行動に関する明確な知識をエージェントに与える。そしてそれぞれのエージェントは個々にグループによる協調行動の進み具合などを保持しており、ほかのエージェントとの協調行動によって問題解決を試みる。また、グループを構成するエージェントを監視しながらグループの再編成をすることもできる。無駄な通信を削減することにも取り組んでおり、これらによって対象とする問題に柔軟に対応でき、また耐故障性をもった協調行動も可能である。しかし、災害救助のような問題に対しては、問題ごとにリーダー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ことや、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という問題がある。

5. 実験

本研究で提案した手法の有効性を確認するために、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として提案手法、Joint Responsibility, STEAMの協調戦略をもたせた3種類の消防隊エージェントを構築する。

ロボカップレスキューシミュレータは、情報を束ねるカーネルに火災延焼、建物倒壊、道路閉塞などの物理シミュレータと消防隊、警察隊、救急隊などのエージェントを接続して、地震災害をシミュレートするものである³⁾。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している。

消防隊エージェントの構築はつぎのように行なった。最初に、移動、消火などの基本能力はもつが、通信能力はもたないベースとなるエージェントを構築した。そしてつぎに提案手法、Joint Responsibility, STEAMの各手法を実現する通信能力を付け加えた。Joint ResponsibilityとSTEAMの協調戦略を実現するアルゴリズムは、それぞれJenningsによる文献⁶⁾およびTambeによる文献⁷⁾に従った。

構築したエージェントを用いて、つぎのような実験を行ない、評価を行なう。

(1) 提案手法によるグループ形成のようす

エージェントの動作を観察し、3章で設計した協調モデルが1章で述べた問題(a)～(e)に対応しているかどうかを調べる。

(2) グループの形成要求回数と形成回数

広がって大きくなっていく火災に対する、3種類の手法の協調戦略によるグループ形成のようすを調べる。

(3) 延焼を防いだ建物数

各協調戦略によって形成されたグループによって、どれほどの延焼を防ぐことができたのかを調べる。

(4) 通信回数

3種類の協調戦略は、通信能力に依存しているため、その通信量について調べる。

5.1 実験の条件

1. レスキューシミュレータ

シミュレータのバージョンは0.40を使用する¹⁾。

2. 使用する地図

仮想都市(Fig. 2)および神戸市長田区のマップを利用する。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¹⁵⁾。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。Fig. 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。また、神戸市長田区マップは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に基づいてつくられている。その大きさは仮想都市マップと同じ400m四方で、建物は740棟である。

3. エージェントの行動

出火直後の火災は単独のエージェントでも容易に消火できる可能性が高い。そこで仮想都市マップでは15ステップ(実時間の15分)、神戸市長田区マップでは20ステップ延焼させた状態からエージェントの行動を開始する。

4. 利用するエージェント

実験では地震発生時の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(仮想都市マップ:15隊, 神戸市長田区マップ:12隊)、道路閉塞の啓開にあたる警察隊エージェント(仮想都市マップ:14隊, 神戸市長田区マップ:10隊)のみとする。また警察隊エージェントは全実験で同一のエージェント(NITRescue02)を利用する。

5. メッセージの受信回数の制限

よりグループの形成に注目するために、グループの形成要求および応答を1ステップで受け取れるようにする。したがって各エージェントの1ステップあたりのメッセージ最大受信回数をフィールドにいるエージェント数(例:15隊)とする。

6. 災害の条件

初期出火建物は仮想都市マップで13棟、神戸市長田区マップで6棟とする。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)を用いる。

7. シミュレーション回数

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、3種類の手法の消防エージェントを用いて、それぞれ50回のシミュレーションを行なう。

8. エージェントの初期配置

ロボカップレスキューシミュレーションでは、エージェントは分散して配置される傾向がある。しかし実際の災害では消防隊が分散していることは稀であることから、分散配置されている場合と集中配置されている場合についてシミュレーションを行なう。分散配置はできるかぎり均一に配置するものとし、集中配置は仮想都市マップで3ヶ所、神戸市長田区マップで2ヶ所にまとめて配置するものとする。なお集中配置の場所は、提案手法がすべての火事を完全消火(鎮火)しづらい場所を選択した。

ここで仮想都市マップと神戸市長田区マップで条件の変更を行なっているのは建物数などの条件が違うためである。

5.2 実験結果

5.2.1 提案手法によるグループ形成のようす

提案手法によるグループ形成のようすを確認するために、ある1回のシミュレーションにおいて、120～170ステップの間に形成したグループのサイズを調べ、Fig. 3に示した。シミュレーションは、1つの消防隊エージェントでは消火できない火事が複数箇所が発生しているものを選択した。

Fig. 3から、ほとんどの時刻で同時に2チーム、多いときには3チームが形成されていることがわかった。また158ス

Fig. 2 The map of virtual city

Fig. 3 Change in size and number of agents' group between 120 steps and 170 steps. The number in kinds of bar graph shows the number of groups and the length of each bar graph shows the number of agents who consist of a group, respectively.

ト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、164ステップで2隊のエージェントが離脱していることがシミュレーションの観察からも確認できた。そして、その後目標が達成したためにこのグループは解散された。これらのことから、本研究の提案手法は、1章で述べた問題 (a) ~ (e) に対応したグループ活動をしていることがわかる。

5.2.2 グループの形成要求回数と形成回数

グループ形成要求回数と形成回数に関する実験結果を Table 1, Table 2 に示した。なお、表に示す値は 50 回のシミュレーション結果の平均値と標準偏差である。また鎮火

回数として、50 回のシミュレーションで、すべての火災が完全に消火されてしまったシミュレーションの回数も示した。

Table 1, Table 2 から、提案手法が最も多くのグループを形成できていることがわかる。また形成要求回数も最も多い。これは協調行動を必要と判断したすべてのエージェントが積極的にグループ形成要求を行なっているからである。提案手法の標準偏差が大きくなっているのは、鎮火回数が多くなっているためである。火災が鎮火されると、それ以後の救助活動が行なわれず、グループ形成も行なわないため、このような結果となっている。

神戸市長田区マップの初期配置が分散の場合 (Table 2),

STEAMの鎮火回数が48回と多くなっている。これについては次節の実験の考察とまとめて述べる。

Table 1 The number of group-forming requests and group formations(virtual city)

methods	group-forming requests		group formations		the number of extinction
	average	SD	average	SD	
Initial Configuration: Dispersion					
Our method	92.12	60.46	24.92	11.22	38
Joint Responsibility	19.22	8.12	10.02	3.22	0
STEAM	4.34	1.33	4.18	1.24	0
Initial Configuration: Convergence					
Our method	176.30	101.58	34.66	10.68	28
Joint Responsibility	25.32	7.70	11.96	4.14	0
STEAM	5.92	1.71	5.08	1.83	0

Table 2 The numbers of group-forming requests and group formations(Nagata-ku, Kobe)

methods	group-forming requests		group formations		the number of extinction
	average	SD	average	SD	
	Initial Configuration: Dispersion				
Our method	155.46	36.85	46.20	7.78	23
Joint Responsibility	24.76	6.83	17.06	3.98	18
STEAM	3.64	1.79	3.38	1.72	48
Initial Configuration: Convergence					
Our method	119.42	56.40	37.68	11.14	41
Joint Responsibility	26.30	3.87	16.10	2.53	5
STEAM	5.68	2.06	5.34	2.04	6

5.2.3 延焼を防いだ建物数

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 **Table 3**, **Table 4** に示す。この実験から各手法が、次第に燃え広がる火災を消火するという複雑な協調問題を、どれくらい解決できるか調べることができる。

Table 3 より、仮想都市マップでは提案手法が最も多く建物を延焼から防いでいることがわかる。これは火災の大きさに合わせてエージェントがグループのサイズを調整し、迅速にグループの形成と解散を行なっているからであると考えられる。提案手法の標準偏差が大きくなっているのは、前述の実験と同様、鎮火回数が多くなっているためである。

また **Table 4** から、神戸市長田区マップの場合は、エージェントの初期配置が分散のとき、STEAMや協調なしの手法のほうが結果が良い。この場合のSTEAMは **Table 2** からわかるとおり、3~4回しかグループを形成していない。つまり、グループを形成せず消火にあたった戦略が有効であったことがわかる。これは神戸市長田区では(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し、道路閉塞がエージェントの移動を制限してしまうことが原因である。本実験で用いた消防隊エージェントでは、警察隊エージェントによる十分な道路啓開を想定しており、定義6で述べた「貢献できない」条件として「移動できない」という条件を含めていなかった。このため **Table 2** のように提案手法では46回程度のグループ形成はできているが協調行動による消火は

妨げられ、延焼を防ぐことできなかったと考えられる。逆に初期配置が集中のときには、提案手法が一番良い。これは初めからエージェントが集まっており移動する必要がないために良い結果となっていると考えられ、初期配置が分散のときの原因を裏付けている。

Table 3 The number of a building that escaped a fire (Virtual City)

methods	buildings that escaped a fire		the number of extinction
	average	SD	
Initial Configuration: Dispersion			
Our method	1013.70	249.25	38
Joint Responsibility	315.58	48.25	0
STEAM	347.16	15.01	0
No cooperation	295.04	0.60	0
Initial Configuration: Convergence			
Our method	871.02	250.52	28
Joint Responsibility	289.98	74.43	0
STEAM	365.82	52.08	0
No cooperation	320.22	1.76	0

Table 4 The number of buildings that escaped a fire (Nagata-ku, Kobe)

methods	buildings that escaped a fire		the number of extinction
	average	SD	
Initial Configuration: Dispersion			
Our method	206.10	134.60	23
Joint Responsibility	288.62	165.64	18
STEAM	546.32	48.97	48
No cooperation	535.96	51.09	47
Initial Configuration: Convergence			
Our method	440.78	162.95	41
Joint Responsibility	97.46	47.26	5
STEAM	94.20	107.26	6
No cooperation	53.92	47.77	0

5.2.4 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行なった通信の回数を **Table 5**, **Table 6** に示す。これらの表からわかるように、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ほか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ーは解散メッセージを、メンバは離脱メッセー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。標準偏差についてはこれまでの実験と同様、鎮火回数の影響がみられる。

6. ま と め

本研究では、大規模災害環境のような緊急性をもつ複雑な環境に対応できる協調モデルの提案を行なった。また、ロボカップレスキュー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した。その結果、提案手法により与えたアルゴリズムが火災の消火に必要なサイズのグループを迅速に形成・解散し、協調して消火活動を行なっていることを確認することができた。

Table 5 The number of communications between agents (Virtual City)

methods	communications		the number of extinction
	average	SD	
Initial Configuration: Dispersion			
Our method	636.64	126.2	38
Joint Responsibility	126.20	45.10	0
STEAM	66.90	61.55	0
Initial Configuration: Convergence			
Our method	977.02	369.42	28
Joint Responsibility	155.24	46.75	0
STEAM	103.58	39.88	0

Table 6 The number of communications between agents (Nagata-ku, Kobe)

methods	communications		the number of extinction
	average	SD	
Initial Configuration: Dispersion			
Our method	769.10	128.34	23
Joint Responsibility	170.96	36.97	18
STEAM	54.92	31.76	48
Initial Configuration: Convergence			
Our method	672.96	196.53	41
Joint Responsibility	165.26	29.21	5
STEAM	73.88	47.64	6

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.rescuesystem.org/robocuprescue/index.html
- 2) Hiroaki Kitano: RoboCup Rescue: Search and Rescue in Large-Scale Disasters as a Domain for Autonomous Agents Research, SMC99 (1999)
- 3) 田所, 北野: RoboCup-Rescue 技術委員会, The RoboCup Federation ロボカップ日本委員会編: ロボカップレスキュー, 共立出版 (2000)
- 4) The Robocup Soccer Simulator:
http://server.sourceforge.net/index.html
- 5) A. Birk, S. Coradeschi and S. Tadokoro (Eds.): RoboCup2001: Robot Soccer World Cup V, Lecture notes in artificial intelligence, **2377** (1999)
- 6) N. R. Jennings: Controlling cooperative problem solving in industrial multiagent system using join intention, Artificial Intelligence, **75** (1995)
- 7) M. Tambe: Toward flexible teamwork, Journal of Artificial Intelligence Research, **7**, 83/124 (1997)
- 8) P. R. Cohen and, H. J. Levesque: Teamwork, Nous, **25-4**, Special Issue on Cognitive Science and Artificial Intelligence, 487/512 (1991)
- 9) E. H. Durfee and V. R. Lesser: Partial global planning: a coordination framework for distributed planning. IEEE transactions on Systems, Man and Cybernetics, **21-5**, 1167/1183 (1991)
- 10) K. Decker and V. Lesser: Designing a family of coordination algorithms. In Proc. of the International Conference on Multi-Agent Systems, 73/80 (1995)
- 11) B. Grosz and S. Kraus: Collaborative plans for complex group actions. Artificial Intelligence, **86**, 269/358 (1996)
- 12) C. Rich and C. Sidner: COLLAGEN: When agents collab-

orate with people. In Proc. of the International Conference on Autonomous Agents(Agent'97), 284/291 (1997)

- 13) D. V. Pynadath and M. Tambe: The Communicative Multiagent Team Decision Problem: Analyzing Teamwork and Theories and Models, Journal of Artificial Intelligence Research **16**, 389/423 (2002)
- 14) P. Paruchuri, M. Tambe, F. Ordonez and S. Kraus: Towards a Formalization of Teamwork with Resource Constraints, In Proc. of AAMAS 2004, 596/603 (2004)
- 15) 高橋, 畑山, 松野俊: 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)

《付 録》

A. 協調モデルの擬似コード

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

```

mystatus := free; mytask = 'none';
repeat
  receive M; /*メッセージキュー*/
  observe tasks T;
  observe mytask if mytask is not 'none';

case mystatus of
leader: /*リーダー*/
  if Breakup_Group(mygroup, mytask) then
    /*グループの解散*/
    send(cancel, myid);
    mystatus := free; mytask := 'none';
  else
    foreach message m=(leave, agentid) in M do
      if agentid in mygroup then
        /*メンバーの離脱*/
        mygroup := mygroup - {agentid};
        act for mytask as a leader;

member: /*メンバー*/
  if there is a message(cancel, agentid) in M
    /*グループの解散要請の処理*/
  then
    if (agentid = myleaderid) then
      mystatus := free; mytask := 'none';
    else if Release_Group(mygroup, mytask) then
      /*タスクに貢献出来ないと判断 → 離脱*/
      send(leave, myid);
      mystatus := free; mytask := 'none';
    else
      act for mytask with myleader;

free: /*単独*/
  if there is a message
    (request, agentid, task_info, priority);
  in M then /*グループ参加要請の処理*/
    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 := 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)
/*タスクが存在している*/
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(request, 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ー*/
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);
/*リーダーの競合が発生*/
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 myid in group then
mystatus := member; mygroup := group;
mytask := mygrouptask;
else
mystatus := free;
act for mytask;

timeleft := timeleft - 1;
endrepeat

```

[著者紹介]

伊藤 暢 浩 (正会員)

1994年名古屋工業大学・工学部・電気情報工学卒。96年同大学大学院博士前期課程修了。99年同大学院博士後期課程修了。博士(工学)。同年同大学工学部助手。現在に至る。オブジェクト指向技術およびエージェント指向技術に関する研究に従事。電子情報通信学会、情報処理学会、人工知能学会、IEEE各会員。

浅井 義 樹

2001年名古屋工業大学・工学部・電気情報工学卒。2003年同大学大学院博士前期課程修了。同年JR東海情報システム(株)入社。現在に至る。エージェント指向技術に興味をもつ。

江 崎 哲 也

2000年名古屋工業大学・工学部・電気情報工学卒。2002年同大学大学院博士前期課程修了。同年松下電器産業(株)入社。現在に至る。エージェント指向技術に興味をもつ。

犬 塚 信 博 (正会員)

1987年名古屋工業大学・工学部・情報工学卒。92年同大学大学院工学研究科博士課程電気情報工学専攻修了。博士(工学)。同年、同大学工学部助手。2000年同大学助教授。現在に至る。94年より96年まで英国インペリアルカレッジを訪問。人工知能、特に機械学習、知識表現に関する研究に従事。電子情報通信学会、情報処理学会、人工知能学会、ソフトウェア科学会、ACM、IEEE Computer、AAAI各会員。

和 田 幸 一

1978年大阪大学・基礎工学部・情報卒。83年同大学院博士後期課程了。同年同大・基礎工・助手。84年名工大・情報(現情報)講師。現在教授。工博。87年~88年ミネソタ大学、ウイスコンシン大学客員助教授。2000アーヘン工科大学客員教授。VLSIの複雑さ、並列分散アルゴリズム、グラフ理論の研究に従事。電子情報通信学会、情報処理学会、ソフトウェア科学会、日本応用数理学会、ACM、IEEE、EATCS各会員。