アルゴリズム 42－9
（1994．11．18）

停止故障耐性を考慮した自律移動ロボット群のための協調問題解法について

吉田 大輔†, 増澤 利光†, 藤原 秀雄†

† 奈良先端科学技術大学院大学 情報科学研究科
〒 630-01 奈良県生駒市高山町 8916-5

E-mail: daisuk-y@is.aist-nara.ac.jp

自律移動ロボット群が協調作業を行なうための分散アルゴリズムについて考える．ロボットは，同じプログラムを非同期に実行し，視覚センサーによって得られた他のロボットの位置から，自分の動作を決定するものとする．本稿では，初期停止故障ロボットが混在するロボット群について考察する．まず，正常なロボットのみからなるグループを構成する問題を正常ロボット選出問題として定義し，過半数のロボットが正常なときに，この問題を解くアルゴリズムを示す．そして，正常ロボット選出問題の解を利用することにより，故障耐性のないアルゴリズムを故障耐性のあるアルゴリズムに変換する方法について考察する．

Distributed Algorithms for Autonomous Mobile Robots with Crash Failures

Daisuke Yoshida†, Toshimitsu Masuzawa†, Hideo Fujiwara†

† Nara Institute of Science and Technology (NAIST)
8916-5 Takayama, Ikoma, Nara, 630-01 Japan

E-mail: daisuk-y@is.aist-nara.ac.jp

We consider distributed algorithms for a system consisting of many autonomous mobile robots. Every robot asynchronously executes the same algorithm to move to a new position that depends on the other robots' positions obtained from its eye sensor. The system we consider includes the initial crash failures such that faulty robots can make no move. We define the Active Robots Selection Problem (ARSP), and presents an algorithm to solve the ARSP if a majority of the robots is fault-free. A technique is also proposed for transforming a non-fault-tolerant algorithm to fault-tolerant one using the solution of the ARSP.

1. まえがき

平面上に配置された多数の自律移動ロボットが, 協調して作業する状況を考える. 自律移動ロボットは, 宇宙空間での任務, 危険な環境下での作業など, 将来的に様々な分野での活躍が期待されている. 複数のロボットの動作を制御する方法として, コントロール・センターがすべてのロボットの状態を把握し, そこで集中的に動作を制御するという集中制御型解法も考えられる. しかし, 通常のシステムと同様, 処理性能, 拡張性, 信頼性などの点から, 各ロボットが, 自分の得た情報から自らの動作を決定する分散型解法が望ましい. これまでに, このような自律移動ロボット群による分散型問題解法に関する研究はいくつか行なわれているが, マイクロロボット技術の進歩などによって, 今後, ますます重要になってくると思われる.

文献 [1],[3],[4],[5],[6] では, 次のような仮定のもとで, 自律分散移動ロボット群の協調動作に関する基礎的研究を行なっている.

- ロボットは大きさを持たない点ロボットである.
- ロボットは通信機能を持たず, 視覚センサーによって得られた他のロボットの位置から, 自分の次の動作を決定する.
- すべてのロボットは, 識別子を持たず, 同じプログラムを実行する. また, 外観などによって, 他のロボットと区別することはできない.
- 各ロボットは, 独自のローカル直交座標系を持ち, 自分や他のロボットの位置をこのローカル直交座標系で認識している. ロボットによって, そのローカル直交座標系の原点の位置, x 軸の方向, 単位距離は異なる.
- ロボットは, 非同期に動作する ([1] 以外). つまり, 動作の速いロボットや、遅いロボットが混在している.
- ロボットの移動は, 瞬時に行なうことができる.

文献 [3] では, すべてのロボットを円状に整列する問題などについて, アルゴリズムを提案し, その性能をシミュレーションによって評価している. また, 文献 [4],[5] では, ロボットを 1 点に集めるという問題や, すべてのロボットに対して共通の原点や単位距離を認識させるという一種の合意問題について, そのアルゴリズムや非可解性を理論的に議論している. これらの文献では, 各ロボットは, 視覚センサーによって, 他のすべてのロボットの位置を知ることができると仮定している. 一方, 文献 [6] では, 視覚センサーの有効距離に制限をおいた場合の円状整列問題について, アルゴリズムを提案し, その性能をシミュレーションによって評価している. また, 文献 [1] では, 視覚センサーに同様の制限をおいた場合について, 理論的な検討を加えている.

上に述べた研究では, いずれも, すべてのロボットは正常で, プログラムを正しく実行すると仮定している. しかし, ロボットの数が多くなるにつれ, ロボット群に故障ロボットが含まれる状況は避けられない. 従って, 故障ロボットが存在するときに, 故障していないロボットが協調して作業するためのアルゴリズムを設計することは, 通常の分散システムの場合と同様に, 非常に重要なことである.

そこで, 本稿では, 故障ロボットが混在するロボット群でのアルゴリズムについて考察する. 自律分散移動ロボット群のための故障耐性のあるアルゴリズムの第 1 ステップとして, 本稿では, 初期停止故障のみを考慮する. 初期停止故障とは, 故障ロボットは一切動作せず, また, アルゴリズム実行開始時から故障している (つまり, アルゴリズム実行中の故障は起こらない) という故障のモデルである. 本稿で扱う非同期式ロボット群では, 停止故障ロボットと, 単に動作の遅い正常なロボットを有限時間内に区別することができない. そのため, 停止故障は, 故障が非同期式システムに及ぼす影響の本質的な一面を特化させたものと考えられる. また, 計算機がネットワークで接続された通常の分散システムにおいて, アルゴリズム実行中に, 1 台の計算機でも停止故障を起こすと, 合意問題のような基本的な問題が解けないことが知られている [2]. そこで, 本稿では, アルゴリズム実行中に生じる停止故障は考慮せず, 初期停止故障のみを扱う.

初期停止故障ロボットが混在するロボット群で, 協調作業を行なう一つの方法として, 正常なロボットのみからなるグループを選びだし, そのグループに属するロボットが, 互いにグループ内の他のロボットを認識することにより, そのグループ内のロボットだけで協調して問題を解くという方法が考えられる. 先に述べたように, 非同期式ロボット群においては, 初期停止故障ロボットを検出することはできない. しかし, 一度でも動作したロボットは正常ロボットであると判断できるので, 正常ロボット (の一部) からなるグループを構成することは不可能ではない. そこで, 本稿では, 正常なロボットのみからなるグループを構成し, そのグループを構成するロボットが, 互いにグループ内の他のロボットを認識するという問題を正常ロボット選出問題として定義する. そして, 過半数のロボットが正常のときに, この問題を解くアルゴリズムを示す. そして, 正常ロボット選出問題の解を利用することにより, 故障耐性のないアルゴリズムを故障耐性のあるアルゴリズムに変換する方法について考察する.

以下,**2** では, ロボット群に関するいくつかの定義を行なう. **3** では, 正常ロボット選出問題を定義し, それを解くアルゴリズムを示す. **4** では, 正常ロボット選出問題の解を利用することにより, 故障耐性のないアルゴリズムを故障耐性のあるアルゴリズムに変換する方法について考察する.

2. 諸定義

n 台の点ロボット $\mathcal{N} = \{r_1, r_2, \cdots, r_n\}$ が,x-y 直交座標系を持つ平面上に存在する状況を考える. この直交座標系を絶対座標系 Z とよび,Z 上でのロボット r の位置座標 (x, y) (x, y は実数) をロボット r の絶対座標とよぶ. 各ロボット r は絶対座標系 Z を知らず,r 独自の x-y 直交座標系 Z_r (ローカル座標系とよぶ) を持つ. Z_r の原点, 単位距離, 座標軸の方向は, 絶対座標 Z や他のロボット q のローカル座標系 Z_q の原点, 単位距離, 座標軸の方向とそれぞれ一致するとは限らない. つまり, ある定数 $d_{x,r}, d_{y,r}, \theta_r, a_r$ (ただし, $a_r \neq 0$) が存在し, 絶対座標 Z 上の任意の点 (x, y) は, 次式によって定まる Z_r 上の点 (x', y') に一致する. ただし, ロボット r は,$d_{x,r}, d_{y,r}, \theta_r, a_r$ を知らず, これらの値は, アルゴリズムでは利用できないものとする.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = a_r \begin{bmatrix} \cos\theta_r & -\sin\theta_r \\ \sin\theta_r & \cos\theta_r \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_{x,r} \\ d_{y,r} \end{bmatrix}$$

[定義 1 (ロボット)] 各ロボット r は, ローカル座標系 Z_r での自分の位置を知っている. また, 視覚センサーを持ち, 他のすべてのロボットのその瞬間での Z_r での位置を, 座標のマルチセットとして知ることができる. ここで, 他のロボットの位置情報が, 座標の (系列ではなく) マルチセットとして得られることは,r が他のロボットを区別できないことを表している.

各ロボット r は, 平面上を移動可能な決定性 (無限) 状態機械 (Q, L, ψ, q_0, Fin) で定義される. ここで,Q,L,ψ,q_0,Fin は, それぞれ, (無限) 状態集合,Z_r での座標の (無限) 集合, 状

態遷移関数, 初期状態 ($q_0 \in Q$), 最終状態の集合 ($Fin \subseteq Q$) を表す. 状態遷移関数$\psi : Q \times L \times \{L^{n-1}\} \to Q \times L$ (ここで, $\{L^{n-1}\}$ は n-1 個の座標からなるマルチセットの集合を表す) は, ロボット rの状態,rの Z_rでの位置, センサーで得られた他のすべてのロボットの Z_rでの位置から,rの新たな状態と移動先を決定する. ただし, 最終状態からは, 状態遷移も移動も起こらないものとする (最終状態は, プログラムの終了状態に対応している). □

ロボット群のためのアルゴリズムとは, 各ロボットに対応する状態機械 (各ロボットが実行するプログラム) を定めるものである.

本稿では, 以下のように仮定する.

[仮定 1] すべてのロボットは, 同じ状態機械である (固有の識別子を持たず, 同じプログラムを実行する). □

[仮定 2] 視覚センサーによる位置情報の獲得, 状態遷移, 移動の一連の動作は, 原子動作であり, 瞬時に実行できる. □

本稿では, 故障ロボットを含むロボット群のためのアルゴリズムを考える. 故障のモデルとしては, 様々なモデルが考えられるが, 本稿では, すべての故障ロボットは初期状況において既に故障しており (つまり, アルゴリズム実行中に, 新たな故障は生じない), 故障ロボットは, 一切動作しないと仮定する. このような故障を, 初期停止故障という (以下では, 単に故障という). また, 故障していないロボットを正常ロボットとよぶ. 以下では, 故障ロボットの集合を $\mathcal{F}$, 正常ロボットの集合を $\mathcal{C}(=\mathcal{N}\backslash\mathcal{F})$ と表す.

[定義 2 (大域状況)] ロボット群 $\mathcal{N}$の大域状況は, すべてのロボットの状態, 絶対座標系 Zでの位置からなる. ただし, 故障ロボットを表すために, ロボットの状態として, 新たに故障状態 $f(\notin Q)$ を導入する. つまり, ロボット群 $\mathcal{N} = \{r_1, r_2, \cdots, r_n\}$ の大域状況は, $\langle \alpha_1, \beta_1, \alpha_2, \beta_2, \cdots, \alpha_n, \beta_n \rangle$ で表される. ここで, 各 $i(1 \le i \le n)$ について,r_iが正常ロボットならα_iは r_i の状態を表し,r_iが故障ロボットなら $\alpha_i = f$である. また,β_iは r_iの Zでの座標を表す. 特に, すべてのロボットが, 初期状態または故障状態であるような大域状況を, 初期大域状況といい, 各ロボットの初期大域状況での位置を初期位置という. また, すべてのロボットの状態が, 最終状態または故障状態であるような大域状況を, 最終大域状況という. □

本稿では, 初期大域状況でのロボットの配置について, 次のように仮定する.

[仮定 3] 初期大域状況において, すべてのロボットは異なる位置に存在する. □

[定義 3 (イベント)] ロボットの原子動作の実行をイベントという. イベントには, 内部状態が変化するだけでロボットが移動しないイベントもあるが, 特に, ロボットが移動するイベントを移動イベントという. 大域状況γにおいて, あるロボット rがイベントϵを起こすことが可能であるとき, ϵはγに適用可能なイベントであるという. □

大域状況γにおいて最終状態または故障状態にあるロボット rについて,γに適用可能な rのイベントは存在しない. 最終状態でも故障状態でもないロボット rについて, γに適用可能な rのイベントは, (rが決定性状態機械なので) 1つしか存在しない.

大域状況γで適用可能なすべてのイベントの集合を $\mathcal{E}(\gamma)$ とする. $\mathcal{E}(\gamma)$ の任意の空でない部分集合 $\mathcal{E}'(\gamma)$ のイベントが同時に起こることにより,γは別の大域状況δに変化する. このとき, $\gamma \to_{\mathcal{E}'(\gamma)} \delta$と表す. つまり, $\gamma = \langle \alpha_1, \beta_1, \alpha_2, \beta_2, \cdots, \alpha_n, \beta_n \rangle$, $\delta = \langle \alpha'_1, \beta'_1, \alpha'_2, \beta'_2, \cdots, \alpha'_n, \beta'_n \rangle$ とするとき, 各 $i(1 \le i \le n)$ について,r_iのイベントϵ_iが $\mathcal{E}'(\gamma)$ に含まれているならば, α'_i,β'_iはϵ_i後の r_iの状態, 及び, 位置を表す. r_iのイベントが $\mathcal{E}'(\gamma)$ に含まれていなければ, $\alpha'_i = \alpha_i$,$\beta'_i = \beta_i$ である.

[定義 4 (実行)] アルゴリズムの実行は, 大域状況の (有限または無限) 列$\gamma_0, \gamma_1, \gamma_2, \cdots$として表される. ここで,$\gamma_0$は初期大域状況であり, 実行が有限列$\gamma_0, \gamma_1, \gamma_2, \cdots, \gamma_f$であるならば, γ_fは最終大域状況である. また, 各 $j(\ge 0)$ について,γ_jで適用可能なイベントの空でない (部分) 集合 $\mathcal{E}'(\gamma_j)$ が存在し, $\gamma_j \to_{\mathcal{E}'(\gamma_j)} \gamma_{j+1}$が成り立つ. □

実行の定義において,γ_jで適用可能なイベント $\mathcal{E}(\gamma_j)$ の任意の空でない部分集合 $\mathcal{E}'(\gamma_j)$ のイベントが同時に起こるということは, 各ロボットは非同期に動作する, すなわち, ロボットの動作速度に関する仮定がないことをモデル化している. このことにより, 大域状況γにおいて,$\mathcal{E}(\gamma)$ のうち, どのイベントがγ で起きるかはわからない. 従って, 一般に, 同じ初期大域状況から始まる実行は複数存在する. ただし, 動作可能な正常ロボットは, いつか必ず動作することを仮定する.

[仮定 4] 任意の実行の任意の大域状況をγとする. γにおいて, 最終状態でも故障状態でもない任意のロボット rについて, γ以降に rのイベントは少なくとも 1 回起こる. □

[定義 5 (問題)] 分散問題 Pは, 実行に関する条件Ψ^Pで定義される. kを任意の非負整数,A を任意のアルゴリズムとする. このとき,$|\mathcal{F}| \le k$を満たす任意の初期大域状況から始まる A の任意の実行が, 条件Ψ^Pを満たすならば, アルゴリズム A は k個の故障に関わらず, 問題 P を解くという. □

3. 正常ロボット選出問題

3.1 問題の定義

故障ロボットを含むロボット群において, 故障ロボットを検出できれば, 正常なロボットだけが協調して問題を解くことにより, 故障耐性のあるアルゴリズムを構成できる. しかし, 非同期システムにおいては, あるロボット rが移動しないとき, rが故障ロボットであるのか, 動作の遅い正常ロボットであるのかを有限時間内に区別することはできない. しかし, 一度でも移動したロボットは正常ロボットであると判断できるので, 正常ロボットの選出は可能である. そこでこの節では, 故障ロボット数の上界 kを仮定し, n-k 台以上の正常ロボットからなる集合 $\mathcal{M}(\subseteq \mathcal{C})$ を求める問題について考察する. ただし, n-k台以上からなる互いに素な正常ロボット集合が異なる集合を求めないために, ここでは, $0 \le k < \lceil n/2 \rceil$ であるとする.

[定義 6 (正常ロボット選出問題)] 正常ロボット選出問題を解く任意の実行において, ある $\mathcal{M}(\subseteq \mathcal{C}, |\mathcal{M}| \ge n\text{-}k)$ が存在し, 任意の $r(\in \mathcal{M})$ は $\mathcal{M}$ を求めて最終状態に遷移する. 任意の $q(\in \mathcal{C}\backslash\mathcal{M})$ は $q \notin \mathcal{M}$ であることを知って最終状態に遷移するか, または, 最終状態に遷移していない[1]. このロボット集合 $\mathcal{M}$ を正常ロボット解集合という. □

[1] 正常ロボット解集合 $\mathcal{M}$ に属するすべてのロボットがアルゴリズムの実行を終了した後に始動するロボットが存在するかも知れない. このロボットに, 解が既に求まっていることを知らせるには, ある 2 台以上のロボットが同じ位置に移動して終了しなければならない. しかし, 本稿では $\mathcal{M}$ を他の問題を解くことに利用するので, $\mathcal{M}$ のすべてのロボットが異なる座標に位置する状況で停止させる. そのため,$\mathcal{M}$ に属さない正常ロボットについては, 最終状態に遷移しないことを許している.

3.2 アルゴリズム *A_Select*

ここでは,k ($0 \leq k < \lceil n/2 \rceil$) 個の故障に関わらず, 正常ロボット選出問題を解くアルゴリズム *A_Select* (*Active_Select*) を示す.

アルゴリズムの説明, 及び, 正当性の証明のために, すべてのロボットに共通な絶対時刻を導入する[2]. ロボット $r \in \mathcal{N}$ が $i(\geq 1)$ 回目のイベントを実行する絶対時刻を $E_r(i)$, このとき視覚センサーにより得られる他のすべてのロボットの位置情報を $P_r(i)$, $t(\geq 1)$ 回目の移動イベントを実行する絶対時刻を $T_r(t)$ と表す. ただし, 便宜上,$T_r(t) = 0$ とし,rが t_r回しか移動イベントを実行しないとき, 任意の $t'(> t_r)$ について,$T_r(t') = \infty$ とする. $E_r(t)$ についても同様にする. また,$\{x | a \leq x < b\}$ なる絶対時刻の集合 (期間) を $[a, b)$ で表す.

ここで $P_r(i)$,$P_r(i+1)$ $(i \geq 1)$ はそれぞれ n-1 個の座標からなるマルチセットであるが, 同じロボットの座標を対応づけることは, 一般には不可能である. しかし, アルゴリズムによってはロボットの動作の規則性などから, 対応づけが可能である場合がある. そのとき, 追跡可能という. 本稿のアルゴリズム *A_Select* では, 各ロボットが移動する範囲を制限することで, 追跡可能としている (補題 2). 従って,*A_Select* では, 各ロボットは他のすべてのロボットを区別することが可能である.

[アルゴリズム A_Select (各ロボット r の動作)]

(変数)

ct_r : 移動回数を表すカウンター (初期値は 0).

d_r : 移動距離を表す変数 (初期値は 0).

$flag_r$:ループの脱出を判定する論理型変数 (初期値は $true$).

MR_r : $ct_r = t$ のとき, ロボットの集合 $\{p | T_r(t)$ 以降に p の移動イベントを rが観測した $\}$ (初期値は ϕ).

$preMR_r$: $ct_r = t$ のとき, ロボットの集合 $\{p | [T_r(t\text{-}1), T_r(t))$ に p の移動イベントを rが観測した $\}$ (初期値は ϕ).

L_r : ロボットの集合 $\{p | p$ が左折したことを rが観測した $\}$ (初期値は ϕ).

(動作)

(a) $ct_r = 0$ のとき (起動時) : $P_r(1)$ において, rに最も近いロボットと rとの距離を 1/16 倍した距離 d_rを求める. Z_rの x 軸の正の方向に距離 d_rだけ移動し,$ct_r := 1$ を実行する.

(b) $ct_r = 1$ のとき : MR_rを求め, $|MR_r| \geq n\text{-}k$が成り立つとき, rは Z_rの x 軸の正の方向に距離 d_rだけ移動し, $ct_r := 2$,$preMR_r := MR_r$,$MR_r := \{r\}$ を実行する.

(c) $ct_r = 2$ のとき : MR_rを求め, $MR_r \supseteq preMR_r$が成り立つとき, rは Z_rの x 軸の正の方向に距離 d_rだけ移動し, $ct_r := 3$,$preMR_r := MR_r$,$MR_r := \{r\}$ を実行する.

(d) $ct_r = 3$ のとき : MR_rを求め, $MR_r \supseteq preMR_r$が成り立つとき,rは次のいずれかの動作をする.

$(d\text{-}1)$ $ct_p = 5$ であるロボット p が存在するならば (rは p の位置の履歴から $ct_p = 5$ かどうかを判定できる (補題 6)), rは Z_rの x 軸の正の方向に距離 d_rだけ移動する (このような rを直進ロボットという). このとき,rは rが正常ロボット解集合に属さないことを知って, アルゴリズムを終了する.

$(d\text{-}2)$ $ct_p = 5$ であるようなロボット p が存在しないとき, rは Z_rの y軸の正の方向に距離 d_rだけ移動する (このような rを左折ロボットという). $ct_r := 4$,$preMR_r := MR_r$,$MR_r := \{r\}$ を実行する.

[2]絶対時刻は説明, 及び, 正当性の証明のためだけに導入するものであり, 各ロボットが実行するプログラムでは利用できない.

(e) $ct_r = 4$ のとき : MR_rを求め, $MR_r \supseteq preMR_r$が成り立つとき, rは初期位置と現在の位置の中点に移動し,$ct_r := 5$ を実行する.

(f) $ct_r = 5$ のとき : 左折ロボットの集合 L_r (rは各ロボットの位置の履歴から左折ロボットかどうかを判定できる (補題 6)) を解とし, アルゴリズムを終了する. □

アルゴリズム *A_Select* の詳細 (ロボット rのプログラム) を図 1 に, ロボットの移動の様子を図 2に示す. プログラム中の手続き *watch* は, 視覚センサーで他のロボットの位置情報を知り, 位置の履歴に追加するための手続きである.

```
A_Select() {
  watch;
  Action_A_Select();
  do {
    Wait_A_Select();
    Action_A_Select();
  } while (flag_r);
}
```

(a) メインプログラム

```
Wait_A_Select() {
  preMR_r:=MR_r;  MR_r:={r};
  do {
    watch;
    MR_rの更新; /* T_r(ct_r)以降に移動したロボット
                   を新たに見つけたら, MR_rに追加 */
  } while (MR_r⊂preMR_r∨|MR_r|<n-k);
}
```

(b) 手続きWait_A_Select

```
Action_A_Select() {
  switch (ct_r) {
    case 0 :
      d_r:=(P_r(1)においてrに最も近い
            ロボットとrとの距離)/16;
      Z_rのx軸の正の方向にd_rだけ移動;
      ct_r:=ct_r+1;
    case 1 : case 2 :
      Z_rのx軸の正の方向にd_rだけ移動;
      ct_r:=ct_r+1;
    case 3 :
      if (∃p∈MR_r[ct_p=5]) then {
        Z_rのx軸の正の方向にd_rだけ移動;
                              /* 直進ロボット */
        flag_r:=false;
      } else
        Z_rのy軸の正の方向にd_rだけ移動;
                              /* 左折ロボット */
      ct_r:=ct_r+1;
    case 4 :
      初期位置との中点に移動;
      ct_r:=ct_r+1;
    case 5:
      L_r:={p|pは左折ロボット};
      flag_r:=false;
  }
}
```

(c) 手続きAction_A_Select

図 1: アルゴリズム *A_Select*

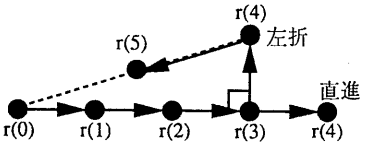

図 2: アルゴリズム *A_Select* での rの軌跡

3.3 アルゴリズム *A_Select* の正当性

アルゴリズム *A_Select* が k個の故障に関わらず, 正常ロボット選出問題を解くことを示す. 以下では, ロボット $r \in \mathcal{N}$の初期位置 (絶対座標) を $r(0)$, $t(\geq 1)$ 回目の移動イベントにより移動した後の絶対座標を $r(t)$ とする.

最初に, アルゴリズム *A_Select* は追跡可能であることを示す. *A_Select* では, 各ロボット rは任意の $i(\geq 2)$ に対し,$P_r(i)$ の各座標を $P_r(i\text{-}1)$ の中で最も近い座標と対応づける. つまり,$P_r(i)$ で座標 vにあるロボットを, $P_r(i\text{-}1)$ で以下のように定まる座標 v'にあったロボットとみなす.

$d(v,v') = min\{d(v,w)|w \in P_r(i\text{-}1)\}$

($d(v,w)$ は v,w間のユークリッド距離を表す)

この対応づけが正しいことを示す. これを示すには, 任意の $r \in \mathcal{N}$について次式を証明すれば良い.

$max\{d(r(i),r(j))|0 \leq i < j \leq 5\} <$
$\quad min\{d(r(i),q(j))|q \in \mathcal{N}\backslash\{r\}, 0 \leq i \leq 5, 0 \leq j \leq 5\}$

[補題 1] 任意のロボット rに対し, 初期大域状況において rに最も近いロボットと rとの距離を D_rとする. 任意の実行において, $max\{d(r(i),r(j))|0 \leq i < j \leq 5\} < 1/3D_r$ が成り立つ.

(証明) 任意のロボット rが $P_r(1)$ から得る rに最も近いロボット $r'(r \neq r')$ と rとの距離を D'_rとする. このとき,r の 1 回の移動距離は $d_r = 1/16D'_r$である. rが位置する座標のうち, その間の距離が最大であるのは, $r(0)$ と直進ロボットの場合の $r(4)$ であり, その距離は $1/4D'_r$である (図 2). つまり,$max\{d(r(i),r(j))|0 \leq i < j \leq 5\} \leq 1/4D'_r$が成り立つ.

初期大域状況において rに最も近いロボットを qとする ($d(r(0),q(0)) = D_r$である).

(*a*) $E_q(1) \leq E_r(1)$ のとき,$D'_q \leq D_r$が成り立つ.

$max\{d(q(i),q(j))|0 \leq i < j \leq 5\} \leq 1/4D'_q$より, $D'_r \leq D_r + max\{d(q(i),q(j))|0 \leq i < j \leq 5\} \leq D_r + 1/4D'_q \leq (1+1/4)D_r = 5/4D_r$が成り立つ. よって,$max\{d(r(i),r(j))|0 \leq i < j \leq 5\} \leq 1/4D'_r \leq 5/16D_r < 1/3D_r$ が成り立つ.

(*b*) $E_q(1) > E_r(1)$ のとき,$D'_r \leq D_r$が成り立つ.

よって,$max\{d(r(i),r(j))|0 \leq i < j \leq 5\} \leq 1/4D'_r \leq 1/4D_r < 1/3D_r$が成り立つ. □

[補題 2] アルゴリズム *A_Select* は追跡可能である. すなわち, 任意の $r \in \mathcal{N}$について,

$max\{d(r(i),r(j))|0 \leq i < j \leq 5\} <$
$\quad min\{d(r(i),q(j))|q \in \mathcal{N}\backslash\{r\}, 0 \leq i \leq 5, 0 \leq j \leq 5\}$

が成り立つ.

(証明) 背理法で証明する. あるロボット $r,q(\in \mathcal{N}\backslash\{r\})$, 整数 i,j,i',j' $(0 \leq i \leq 5, 0 \leq j \leq 5, 0 \leq i' < j' \leq 5)$ が存在し, $d(r(i),q(j)) \leq d(r(i'),r(j'))$ が成り立つと仮定する. 補題 1より, $d(r(i'),r(j')) < 1/3D_r$であるから, $d(r(i),q(j)) < 1/3D_r$が成り立つ. また $d(r(0),r(i)) < 1/3D_r$, $d(q(0),q(j)) < 1/3D_q$も成り立つ. よって,$d(r(0),q(0)) \leq d(r(0),r(i)) + d(r(i),q(j)) + d(q(0),q(j)) < 2/3D_r + 1/3D_q$ が成り立つ.

$D_r \leq d(r(0),q(0)) < 2/3D_r + 1/3D_q$より, $D_r < D_q$が成り立つ. 同様に,$D_q \leq d(r(0),q(0)) < 2/3D_r + 1/3D_q$より, $D_q < D_r$が成り立つ. これらは矛盾する. □

アルゴリズム *A_Select* の任意の実行について, 次の 2 つの性質が明らかに成り立つ.

[性質 1] 任意の $r \in \mathcal{C}, t(\geq 2)$ に対し,$[T_r(1),T_r(t))$ の間に n-k台以上のロボットが移動イベントを実行する. □

[性質 2] 任意の $r \in \mathcal{C}, t(\geq 3)$ に対し,$T_r(t) \neq \infty$ ならば, $[T_r(1),T_r(t\text{-}1))$ に移動イベントを実行したすべてのロボットは $[T_r(t\text{-}1),T_r(t))$ に移動イベントを実行する. □

性質 2から, 次の補題を証明できる.

[補題 3] 任意の 2 つのロボット $r,r' \in \mathcal{C}$, 任意の t,t' $(t \geq 2, t' \geq 2)$ について, $T_r(t) < T_{r'}(t')$ かつ $T_r(t+1) \neq \infty$ が成り立つならば, $T_r(t+1) < T_{r'}(t'+1)$ が成り立つ.

(証明) 背理法で証明する. $\exists r,r' \in \mathcal{C}, \exists t,t'(\geq 2)$ $[(T_r(t) < T_{r'}(t')) \wedge (T_r(t+1) \neq \infty) \wedge (T_r(t+1) \geq T_{r'}(t'+1))]$ が成り立つと仮定する.

(*a*) $T_{r'}(1) \leq T_r(t)$ のとき (図 3(*a*)) : $T_r(t) < T_{r'}(t')$ より, rは $[T_{r'}(1),T_{r'}(t'))$ に移動イベントを実行している. $T_r(t+1) \neq \infty, T_{r'}(t'+1) \leq T_r(t+1)$ より, $T_{r'}(t'+1) \neq \infty$ である. 従って性質 2より,rは $[T_{r'}(t'),T_{r'}(t'+1))$ に移動イベントを実行する. これは $T_{r'}(t'+1) \leq T_r(t+1)$ に矛盾する.

(*b*) $T_r(t) < T_{r'}(1)$ のとき (図 3(*b*)) : $[T_r(1),T_r(t))$ に移動イベントを実行したロボットの集合を $\mathcal{C}_1$, $[T_{r'}(1),T_{r'}(t'))$ に移動イベントを実行したロボットの集合を $\mathcal{C}_2$とする. 性質 1より,$|\mathcal{C}_1| \geq n\text{-}k, |\mathcal{C}_2| \geq n\text{-}k$である. $k < \lceil n/2 \rceil$ より,$|\mathcal{C}_1 \cap \mathcal{C}_2| \geq 1$ である. 従って, ある $q \in \mathcal{C}_1 \cap \mathcal{C}_2$が存在し, ある a,b $(1 \leq a < b)$ に対し, $T_r(1) \leq T_q(a) < T_r(t)$, $T_{r'}(1) \leq T_q(b) < T_{r'}(t')$ が成り立つ. また性質 2より qは $[T_{r'}(t'),T_{r'}(t'+1))$ に移動イベントを実行する. つまり, ある $c(> b)$ に対し, $T_{r'}(t') \leq T_q(c) < T_{r'}(t'+1)$ が成り立つ. rは $[T_q(a),T_q(b))$ に移動イベントを実行しているので, 性質 2より rは $[T_q(b),T_q(c))$ に移動イベントを実行する. すなわち,$T_r(t+1) < T_q(c)$ である.$T_q(c) < T_{r'}(t'+1)$ より $T_r(t+1) < T_{r'}(t'+1)$ が成り立つが, これは $T_{r'}(t'+1) \leq T_r(t+1)$ に矛盾する. □

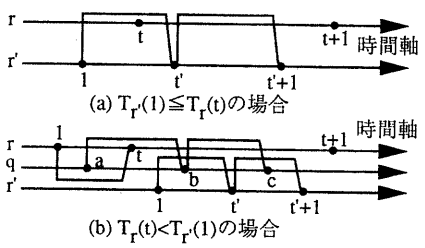

図 3: $T_{r'}(1)$ と $T_r(t)$ の大小関係による場合分け

補題 3より, 次の系 1,2が成り立つ.

[系 1] 任意の 2 つのロボット $r,r' \in \mathcal{C}$, 任意の t,t' $(t \geq 3, t' \geq 3)$ について, $T_r(t) < T_{r'}(t')$ が成り立つならば, 任意の $u(1 \leq u \leq min\{t-2, t'-2\})$ について $T_r(t-u) \leq T_{r'}(t'-u)$ が成り立つ. □

[系 2] 任意の 2 つのロボット $r,r' \in \mathcal{C}$, 任意の t,t' $(t \geq 3, t' \geq 3)$ について, $T_r(t) = T_{r'}(t')(\neq \infty)$ が成り立つならば, 任意の $u(1 \leq u \leq min\{t-2, t'-2\})$ について $T_r(t-u) = T_{r'}(t'-u)$ が成り立つ. □

[補題 4] アルゴリズム *A_Select* の任意の実行において，カウンター ct の値が5になるロボットは存在する．

(証明) 背理法で証明する．ある実行において，ある絶対時刻 T においてすべてのロボットのカウンター ct の値が4以下であり，T 以降どのロボットの ct も値が変化しないと仮定する．

任意の正常ロボットは1回目の移動イベントは無条件に実行するので，T 以前にすべての正常ロボットは少なくとも1回は移動イベントを実行している．正常ロボット r が最後に移動イベントを実行した絶対時刻を T_r とし，$T_{min} = min\{T_r | r \in \mathcal{C}\}$, $T_{max} = max\{T_r | r \in \mathcal{C}\}$ とする．$T_p = T_{min}$ を満たす任意のロボットを p とする．$ct_q \geq 5$ なる q は存在しないので，p は直進ロボットではない．従って，p はアルゴリズムを終了していない．任意の正常ロボット r について，$r(0)$, $r(1)$, $r(2)$, $r(3)$, $r(4)$ の位置は異なること，及び，T_{min} 以降に少なくとも1回は移動イベントを実行することから，p は T_{max} 以降にイベントを実行したとき，$MR_p = \mathcal{C} \supseteq preMR_p$ が成り立ち，p は移動イベントを実行する．これは T_p の定義に矛盾する． □

以下では最初にカウンター ct の値が5になる任意のロボットを r^* とし，$\mathcal{C}^* = \{r \in \mathcal{C} | T_{r^*}(4) \leq T_r(4) \leq T_{r^*}(5)\}$ とする．

[補題 5] $T_r(i) \leq T_{r^*}(5) < T_r(i+1)$(ただし，$i \leq 4$) を満たす任意のロボット r について，$T_r(i+1) \neq \infty$ が成り立つ．

(証明) 背理法で証明する．$T_r(i) \leq T_{r^*}(5) < T_r(i+1)$(ただし，$i \leq 4$)，かつ，$T_r(i+1) = \infty$ なる r が存在すると仮定する．任意の正常ロボットは1回目の移動イベントは無条件に実行するので，$i \neq 0$ である．すなわち，r は少なくとも1回は移動イベントを実行しており，r のうち，最後の移動イベントを実行した絶対時刻が最小である任意のロボットを p とする．p はアルゴリズムを終了していない．

任意の正常ロボット r について，$r(0)$, $r(1)$, $r(2)$, $r(3)$, $r(4)$, $r(5)$ の位置は異なるので，r が移動イベントを実行するならば，他のロボットは r が移動イベントを実行したことを知ることができる．

(*a*) $i = 1$ であるとき：

(*a*-1) $T_p(i) \leq T_{r^*}(4)$ のとき：$[T_{r^*}(4), T_{r^*}(5))$ に n-k 台以上のロボットが移動イベントを実行することから，p は $T_{r^*}(5)$ 以降にイベントを実行したとき，$|MR_p| \geq n$-k が成り立ち，p は移動イベントを実行する．これは $T_p(i+1) = \infty$ に矛盾する．

(*a*-2) $T_p(i) > T_{r^*}(4)$ のとき：$[T_{r^*}(4), T_p(i))$ に移動イベントを実行したロボットの集合を $\mathcal{C}'$ とする．任意のロボット $q \in \mathcal{C}'$ は，p の定義から $T_p(i)$ 以降に少なくとも1回は移動イベントを実行する．$T_p(i)$ 以降に初めて移動イベントを実行した絶対時刻を T_q とすると，補題3から，$T_{r^*}(5) \leq T_q$ である．$[T_{r^*}(4), T_{r^*}(5))$ に n-k 台以上のロボットが移動イベントを実行するが，$[T_{r^*}(4), T_p(i))$ に移動イベントを実行した任意のロボットは，$T_{r^*}(5)$ 以降にも移動イベントを実行する．従って，p は $max\{T_q | q \in \mathcal{C}'\}$ 以降にイベントを実行すると，$|MR_p| \geq n$-k が成り立ち，p は移動イベントを実行する．これは $T_p(i+1) = \infty$ に矛盾する．

(*b*) $i \geq 2$ であるとき：p の定義から，$[T_p(i\text{-}1), T_p(i))$ に移動イベントを実行した任意のロボットは，$T_p(i)$ 以降に少なくとも1回は移動イベントを実行する．従って，いつか $MR_p \supseteq preMR_p$ が成り立ち，p は移動イベントを実行する．これは $T_p(i+1) = \infty$ に矛盾する． □

図1のアルゴリズム *A_Select* では，各ロボット r が，他のロボット p について，p が左折ロボットであるか，また，$ct_p = 5$ であるかどうかを判定している．これは，実際には，次のように行なう．

- r がこれまでに知っている p の位置のうち，一直線上にない3点があれば，r は p を左折ロボットと判断する．
- r がこれまでに知っている p の位置のうち，$\angle p_1p_2p_3 < \pi/2$ (鋭角) なる3点 p_1, p_2, p_3 (ただし，p の位置履歴の順に p_1, p_2, p_3 とする) があれば，r は $ct_p = 5$ と判断する．

このとき，明らかに，次の補題が成り立つ．

[補題 6] 任意のロボットを r，任意の左折ロボットを p とする．r が $p(2)$, $p(3)$, $p(4)$, $p(5)$ のうちの任意の3つの位置を知れば，r は p が左折ロボットであると判断する．さらにこのとき，r が $p(5)$ を知っていれば，r は $ct_p = 5$ と判断する． □

[補題 7] 任意のロボット $r \in \mathcal{C}^*$ は $ct_r = 5$ となり，アルゴリズムの実行を終了する．このとき求めた左折ロボットの集合を L_r とすると，$\mathcal{C}^* \subseteq L_r$ が成り立つ．

(証明) 補題5より，任意の $r \in \mathcal{C}^*$ は $ct_r = 5$ となり，アルゴリズムの実行を終了する．

任意の2つのロボットを $r, r' \in \mathcal{C}^*$ とする．一般性を失うことなく，$T_r(5) \leq T_{r'}(5)$ と仮定できる．このとき，

(*a*) $T_r(5) = T_{r'}(5)$, (*b*) $T_{r'}(4) < T_r(5) < T_{r'}(5)$, (*c*) $T_r(5) = T_{r'}(4) < T_{r'}(5)$ のいずれかが成り立つ．

(*a*) 系2より，各 $i(2 \leq i \leq 5)$ について，$T_r(i) = T_{r'}(i)$ である．このとき，r' は $\{r(j) | 1 \leq j \leq 5\}$ を知る．また，r は $\{r'(j) | 1 \leq j \leq 5\}$ を知る．

(*b*-1) 各 $i(2 \leq i \leq t)$ について $T_r(i) = T_{r'}(i)$，かつ，各 $i(t+1 \leq i \leq 5)$ について $T_r(i) < T_{r'}(i)$ を満たす $t(2 \leq t \leq 4)$ が存在するとき，r' は $\{r(j) | 1 \leq j \leq 5, j \neq t\}$ を知る．また，r は $\{r'(j) | 1 \leq j \leq 4\}$ を知る．

(*b*-2) 各 $i(2 \leq i \leq 5)$ について $T_r(i) < T_{r'}(i)$ であるとき，r' は $\{r(j) | 2 \leq j \leq 5\}$ を知る．また，r は $\{r'(j) | 1 \leq j \leq 4\}$ を知る．

(*c*) 系2より，各 $i(2 \leq i \leq 4)$ について $T_r(i+1) = T_{r'}(i)$ である．このとき，r' は $\{r(j) | 2 \leq j \leq 5\}$ を知る．また，r は $\{r'(j) | 1 \leq j \leq 4\}$ を知る．

以上から，r は $\{r'(j) | 2 \leq j \leq 5\}$ のうち少なくとも3つの位置を知ることができ，r' も $\{r(j) | 2 \leq j \leq 5\}$ のうち少なくとも3つの位置を知ることができる．従って，補題6から，r, r' は互いに他が左折ロボットであると判断する．すなわち $\mathcal{C}^* \subseteq L_r$ が成り立つ． □

[補題 8] $T_{r^*}(4) \leq T_r(3) \leq T_{r^*}(5) < T_r(4)$ を満たす任意のロボット r は直進ロボットである．

(証明) (*a*) $T_{r^*}(4) = T_r(3)$, (*b*) $T_{r^*}(4) < T_r(3) < T_{r^*}(5)$, (*c*) $T_r(3) = T_{r^*}(5)$ のいずれかが成り立つ．また，補題5より $T_r(4) \neq \infty$ である．

(*a*) 系2より $T_{r^*}(3) = T_r(2)$ が成り立つ．r は $T_r(4)$ までに，$\{r^*(j) | 2 \leq j \leq 5, j \neq 4\}$ を知る．

(*b*) 系1より $T_{r^*}(3) \leq T_r(2)$ が成り立つ．

(*b*-1) $T_{r^*}(3) = T_r(2)$ であるとき，r は $T_r(4)$ までに，$\{r^*(j) | 2 \leq j \leq 5, j \neq 3\}$ を知る．

(*b*-2) $T_{r^*}(3) < T_r(2)$ であるとき，r は $T_r(4)$ までに，$\{r^*(j) | 3 \leq j \leq 5\}$ を知る．

(*c*) 系2より $T_r(2) = T_{r^*}(4)$ が成り立つ．r は $T_r(4)$ までに，$\{r^*(j) | 3 \leq j \leq 5\}$ を知る．

以上から，r は $T_r(4)$ までに，$\{r^*(j) | 2 \leq j \leq 4\}$ のうち少なくとも2つの位置と $r^*(5)$ を知る．従って，補題6から，r は $ct_{r^*} = 5$ であると判断し，$T_r(4)$ において直進する． □

また補題3より次の補題が明らかに成り立つ．

[補題 9] $T_{r^*}(5) < T_r(3)$ を満たす任意のロボット rについて,$T_r(4) = \infty$ である. □

[補題 10] $|\mathcal{C}^*| \geq n\text{-}k$が成り立つ.

(証明) $T_{r^*}(4) \leq T_r(4) < T_{r^*}(5)$ を満たすロボット rが $n\text{-}k$ 台以上存在することを示す.

性質1より,$[T_{r^*}(1), T_{r^*}(2))$ に少なくとも $n\text{-}k$台以上のロボットが移動イベントを実行している. 性質2より, これらのロボットは $[T_{r^*}(2), T_{r^*}(3))$,$[T_{r^*}(3), T_{r^*}(4))$, $[T_{r^*}(4), T_{r^*}(5))$ にそれぞれ1回移動イベントを実行する. よって,r^*が最初に $ct_{r^*} = 5$ となることから, $T_{r^*}(4) \leq T_r(4) < T_{r^*}(5)$ が成り立つ. rは $n\text{-}k$台以上存在する. □

[定理 1] $0 \leq k < \lceil n/2 \rceil$ であるとき, アルゴリズム *A_Select* は k個の故障に関わらず, 正常ロボット選出問題を解く.

(証明) 補題 8, 補題 9 から, $\mathcal{C} \backslash \mathcal{C}^*$のロボットは左折ロボットではない. よって, 補題 7により, 任意の $r \in \mathcal{C}^*$は $L_r = \mathcal{C}^*$を解集合として, アルゴリズムを終了する. 補題10より,$|L_r| \geq n\text{-}k$である. 補題 8より, $T_{r^*}(4) \leq T_r(3) \leq T_{r^*}(5) < T_r(4)$ を満たす任意の rは $T_r(4)$ において, 自分が解集合に含まれないことを知り, アルゴリズムを終了する. また補題9より, $T_{r^*}(5) < T_r(3)$ を満たす任意のロボット r は最終状態に遷移することはない. □

また, アルゴリズムから, 次の定理が明らかに成り立つ.

[定理 2] アルゴリズム *A_Select* の任意の実行において, 任意のロボットは高々5回しか移動イベントを実行しない. □

4. 故障耐性のためのアルゴリズム変換

4.1 アルゴリズム変換

故障ロボットが存在しない状況で, 問題 Pを解く任意のアルゴリズムを A とする. ここでは, アルゴリズム A を, 故障にかかわらず問題 Pを解くアルゴリズム A^*に変換する手法について述べる.

簡単のため, アルゴリズム A での各ロボット rのプログラムは, 図 4 の形式で与えられるものと仮定する. ただし, 図 4において,rが使用する変数 $flag_r$,MR_rはアルゴリズム *A_Select* と同様に使用される. また,$Action_A$では, rの内部状態 (それ以前に視覚センサーから得た各ロボット $q \in \mathcal{N}$の位置履歴を含む) から決定される動作を実行する. 実際には, 具体的に $Action_A$を決めることによって, アルゴリズムが決まる.

[アルゴリズム A* (各ロボット r の動作)]

rが使用する各変数はアルゴリズム *A_Select* と同様.

(*a*) $ct_r \leq 4$ のとき：*A_Select* と同様に動作する.

(*b*) $ct_r = 5$ のとき：左折ロボットの集合 L_rを求めていなければ,L_rを求める. また, rは MR_rを求め,$MR_r \supseteq L_r$が成り立ち, かつ, 任意の $q \in L_r$について,$ct_q \geq 5$ (rは qの位置履歴から,$ct_q \geq 5$ かどうかを判定できる (後述の仮定 (*A*))) が成り立つならば, L_rに対してアルゴリズム A の実行を開始する. つまり,rの内部状態をアルゴリズム A の初期状態に設定し[3], 以降は,L_r以外のロボットを無視して A を実行する. □

アルゴリズム A^* (rのプログラム) の詳細を図5に示す. $Wait_{A\_Select}$,$Action_{A\_Select}$,$Wait_A$はそれぞれ, 図 1(b), 図 1(c), 図 4(b) に示した手続きである.

[3]直前の *watch* は A の最初の *watch* に相当するので, そこで得られた L_rのロボットの位置情報は初期化してはいけない.

```
A() {
  watch;
  Action_A(W);
  do {
    Wait_A(W);
    Action_A(W);
  } while (flag_r);
}
```

(a) メインプログラム

```
Wait_A(G) {
  MR_r := {r};
  do {
    watch;
    MR_rの更新;
  } while (MR_r⊂G);
}
```

(b) 手続き$Wait_A$

図 4: アルゴリズム A

```
A*() {
  watch;
  Action_A_Select();
  do {
    Wait_A_Select();
    Action_A_Select();
  } while (flag_r);
  if (ct_r=5) then {
    while (∃q∈L_r[ct_q≦4]) do
      watch;
    内部状態の初期化;
    do {
      Wait_A(L_r);
      Action_A(L_r);
    } while (flag_r);
  }
}
```

図 5: アルゴリズム A^*

カウンター *ct* の値が 4 以下のときの A^*の動作は *A_Select* と同じである. 補題 4よりカウンター *ct* の値が 5 になるロボットは存在する. 最初に $ct = 5$ になる任意のロボットを r^*で表し, $\mathcal{C}^* = \{r \in \mathcal{N} | T_{r^*}(4) \leq T_r(4) \leq T_{r^*}(5)\}$ とする.

C^*に属する任意のロボットを r,r'とする. $T_r(5) = T_{r'}(5) < T_r(6) < T_{r'}(6) \neq \infty$ のとき, r'は rの位置 $r(5)$ を知ることができない. このため,$r(4) = r(6)$ が成り立つと,r'は rの移動を検知できず, アルゴリズム A^*はデッドロック状態に陥ってしまう. また, アルゴリズム A^*では, どのロボットが移動したかを知る必要がある. すなわち,A^*は追跡可能である必要がある.

そこで, 次の仮定をおく.

(*A*) $T_r(6) \neq \infty$ なる任意のロボット rについて, $r(4) \neq r(6)$ が成り立つ.

(*B*) アルゴリズム A^*は追跡可能である.

故障ロボットが存在しない状況で問題 Pを解くアルゴリズムのうち, 仮定 (*A*),(*B*) を満たさないものに対しても, アルゴリズム *A_Select* のように, ロボットの移動範囲を制限することにより, 仮定 (*A*),(*B*) を満たすように変形できることが多い. 例えば, 文献 [4],[5] において示された, 各

ロボットの原点を一致させるアルゴリズム, 単位距離を一致させるアルゴリズムなどは, 仮定 (A),(B) を満たすように, 容易に変形できる.

[定理 3] アルゴリズム A は, 故障ロボットが存在しない状況で, 問題 Pを解くとする. ただし A は, 図 4の形式で与えられるものである. $0 \leq k < \lceil n/2 \rceil$ であるとき,A より得られる, アルゴリズム A^*の任意の実行において, ある正常ロボットの集合 $\mathcal{M}$ $(\subseteq \mathcal{C}, |\mathcal{M}| \geq n\text{-}k)$ が存在し, $\mathcal{M}$ に属する任意のロボットは,$\mathcal{M}$ に対する Pの解を求めて停止する. □

(注1) 問題によっては, その解がロボットの初期位置に依存する場合がある. そのような問題に対しては, 任意の $r \in \mathcal{M}$ は A の実行を開始する前に, その初期位置 $r(0)$ に移動するように変更することで対応できる.

(注2) A^*には, 正常ロボット解集合 $\mathcal{M}$ に属さない正常ロボット qが左折ロボット集合 $L_q(\supseteq \mathcal{M} \cup \{q\})$ を求めてしまうことがある. これは, 例えば,$\mathcal{M}$ が求まった後に qが始動し, かつ, アルゴリズム A によるロボットの動作がアルゴリズム *A_Select* のロボットの動作と偶然に一致するときなどである. しかしながら, 解を求めて停止するまでのロボットの移動回数が,L_qに対して A を実行した場合の方が,$\mathcal{M}$ に対して A を実行した場合より少なくなければ[4], このようなロボット qは停止しない.

4.2 変換の適用例：合意問題

[定義 7 (合意問題)] 合意問題とは, 各ロボット $r \in \mathcal{N}$ が初期データ $I_r(\in \{0,1\})$ を持つとき, すべてのロボットが同じ値 $O(\in \{0,1\})$ を決定して停止する問題である. ただし, すべてのロボットの入力データが同じ (その値を I とする) なら, $O = I$ でなければならない. □

各ロボット r は, 任意の方向に進んだ後,$I_r = 0$ なら左折,$I_r = 1$ なら右折することにより, 自分の初期データ I_r を他のロボットに知らせることができる. 従って, 故障ロボットが存在しなければ, 合意問題は容易に解ける. 合意問題を解くためのアルゴリズム *Agreemet* は A の手続き $Action_A$を図 6に示す手続き $Action_{Agreement}$ に置換したものである.

```
Action_Agreement(G) {
  switch (ct_r) {
    case 0 :
      d_r:=(P_r(1)においてrに最も近い
            ロボットとrとの距離)/16;
      Z_rのx軸の正の方向にd_rだけ移動;
    case 1 :
      Z_rのx軸の正の方向にd_rだけ移動;
    case 2 : case 3 :
      if (I_r=0) then
        Z_rのy軸の正の方向にd_rだけ移動;
      else
        Z_rのy軸の負の方向にd_rだけ移動;
    case 4 :
      if (左折ロボットの数>|G|/2) then O:=0;
      else O:=1;
      flag_r:=false;
  }
  ct_r:=ct_r+1;
}
```

図 6: 手続き $Action_{Agreement}$

[4]移動回数がロボットの総数に対して単調非減少であるときなど.

アルゴリズム *Agreement* は, 左折/右折によって, 情報を伝えているので, 移動距離を制限することにより, 仮定 (A),(B) を満たすように変更できる. 従って, 上で述べた変換で得られるアルゴリズム *Agreement** は, 故障にかかわらず合意問題を解くアルゴリズムである.

5. あとがき

本稿では, 初期停止故障ロボットを含むロボット群で, 協調して問題を解くためのアルゴリズムについて考察した. 特に, 故障耐性のないアルゴリズムを故障耐性のあるアルゴリズムに変換する方法について, 検討を加えた. 今後の課題として, 初期停止故障以外の故障を考慮することなどが考えられる.

参考文献

[1] H. Ando, I. Suzuki, M. Yamashita, "Formation and agreement problems for synchronous mobile robots with limited visibility," 1995 IEEE International Conference on Robots and Automation (submitted).

[2] M.J.Fischer, N.A.Lynch, M.S.Paterson, "Impossibility of distributed consensus with one faulty process," JACM, Vol.32, No.2, pp.374-382 (April 1985).

[3] K. Sugihara, I. Suzuki, "Distributed motion coordination of multiple mobile robots," Proceedings of the 5th IEEE International Symposium on Intelligent Control, Philadelphia, Pennsylvania, pp.138-143 (September 1990).

[4] I. Suzuki, M. Yamashita, "Cooperative control algorithms for anonymous mobile robots," 第6回回路とシステム軽井沢ワークショップ論文集, pp.523-530 (1993).

[5] I. Suzuki, M. Yamashita, "Formation and agreement problems for anonymous mobile robots," Proc. 31st. Ann. Allerton Conference on Communication, Control and Computing, Monticello, Illinois, pp.93-102 (Oct. 1993).

[6] 吉田大輔, "自律移動ロボット群を円状配置する協調分散型解法," 大阪大学基礎工学部特別研究報告 (1993年2月).