平成 23 年度　特別研究報告書

災害救助ロボットのための MANETを用いた自動エリア探索手法

龍谷大学　理工学部　情報メディア学科
学籍番号　T 070391　池嶋　隆史
指導教員　三好　力　教授

内容梗概

東北地方太平洋沖地震によって、広域にわたって建物が倒壊し、多くの人々がその下敷きになった。「72 時間の壁」という言葉があるように、このような形で栄養の供給を断たれた人々は迅速に発見されなければならない。

広い被災地あるいは、内部の探索が困難な倒壊跡の探索には、ロボットを用いて、自律的に探索するのが効率がよいと考えられる。しかし、現在の研究ではこの形態で探索をさせる上で決定的な問題点がある。それは、探索でロボットが得た情報を基地局に迅速に伝えるために、ロボットと基地局が常に通信できる状態にしておく必要があるが、そうすることで、ロボットが探索エリアの遠方まで行くことができなくなるというものである。

本研究ではこの問題を解決するために、探索中にロボットに通信を中継するノードを置くということを提案し、シミュレーションによってその探索効率を調べた。ノードを置く位置とノードを置く条件が異なる提案手法を 3 つ考案したが、シミュレーションでの実験の結果、どの手法も既存技術より高い探索カバー率を得ることができた。

目次

1 はじめに

1.1 本研究の背景

2012 年 1 月 13 日の東北地方太平洋沖地震ではマグニチュード 9.0 という高い値を記録した。9 月 11 日時点の警察庁の発表[1]では、死者 15,844 人、行方不明者 3,394 人、負傷者 5,893 人という甚大な被害をもたらし、図 1 のように、宮城、福島を中心に広い地域で震度 6 を越える地震を観測し、広い範囲において建物やインフラが破壊された。

この地震が起こる以前の日本国内では東北地方の地震よりもむしろ、東海・東南海の地震の発生を危惧しており、現在の地震予知能力の限界がわかった。

さらに、仮に地震の発生が予知できたとしても、地震の発生を防ぐことは不可能であり、人間にできることは地震が起こった時の被害を減らす（＝減災）ことである。

世界の地震の 2 割が日本で起こることもあり、日本国内の建物の建造は、耐震を強く意識して行われている。しかしながら、建物の老朽化や小さい地震の積み重ねなどでわずかに破損し、必ずしもすべての建物が地震に耐えられるものではない。したがって今後、巨大地震が起こった際には必ず、崩壊する建物が存在する、と考えた方がよい。

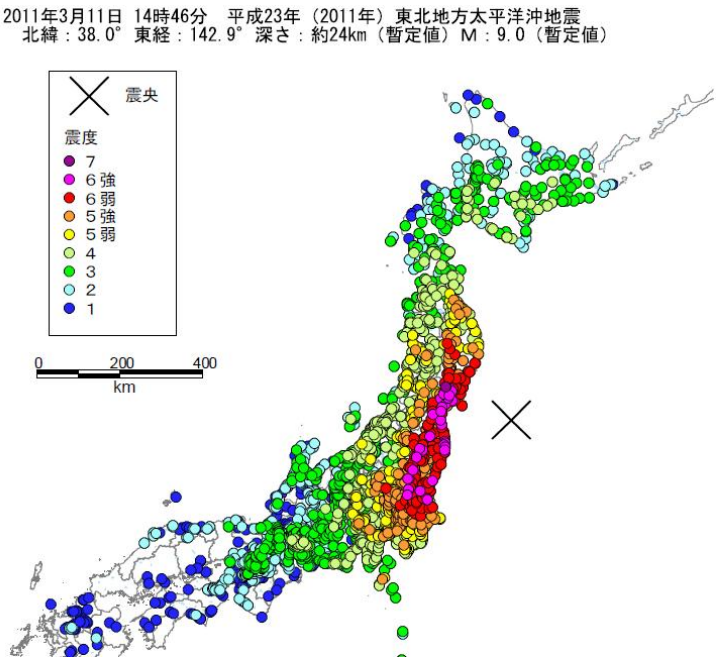

図 1:地震発生の際の震度のデータ　×が震源

建物が崩壊した際に生き埋めになった人々を助けることは減災に不可欠である。そして救助の際、スピードを意識してやる必要がある。救助の現場でよくいわれる経験則で「72 時間の壁」というものがある。これは、倒壊した建物で生き埋めになった人々の救助にかかる時間が長いほど、救出した時の生存率が減っていき、72 時間経過した時点では、生存率は 6%にも下がるというものである。主な原因は脱水症状による。

現在主に行われている倒壊建物内の探索は、人間が赤外線カメラや CO2 の濃度を使って人間の有無を確認している。その例として、東北地方太平洋沖地震に先立って発生したニュージーランドでの地震における建物内の探索の様子を図 2 に示す。

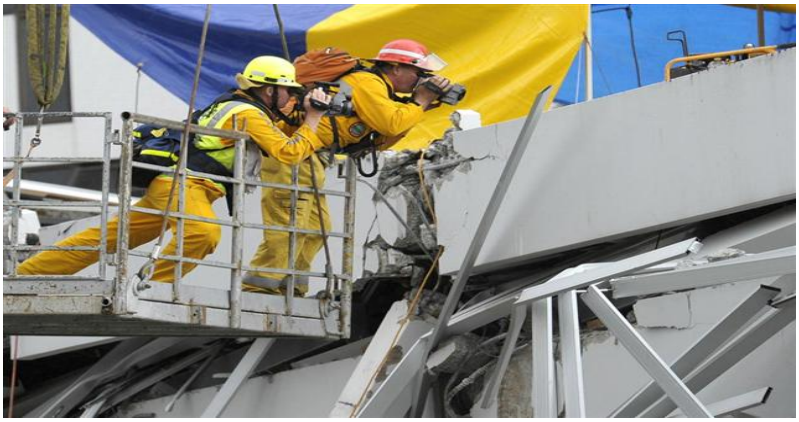

図 2:ニュージーランドで発生した地震の際の救助の様子

被害の規模が局所的かつ、倒壊した瓦礫の状態が比較的平坦ならば、この手法でも有効に思われる。しかしながら、倒壊した状態が 9.11 のワールドトレードセンターのような高さを持っていたら、倒壊した建物の奥に入るための空間的余地は極めて狭い一方、人間の体は体積が大きいため、瓦礫の撤去が必要である。結果、瓦礫の撤去に時間がかかり、瓦礫の奥に閉じ込められた人間の発見はかなり遅れるはずである。したがって、小型のロボットを用いて建物内を、センサーを持って動き回ってくれれば、現在の人命救助に多大な貢献をしてくれると考えられる。

1.2 本研究の目的

前節にあるように、現在の人手による倒壊建物内の探索は、巨大地震による広域の災害や超高層ビルの崩壊などの甚大な災害では効率的とはいえない。倒壊した建物内での生存者確認を迅速に行うためには、人間よりも小さいロボット複数台を内部に送り込み、互いに協調しながら探索を行う方がよい。

また、探索において、現在多く採用されている無線を使った有人の操作では、広い範囲で建物が倒壊した場合に、操縦者の確保、またそれに伴う人員の圧迫が予想される。

したがって本研究では、倒壊した建物内の探索を、ロボットにさせることを最終目的として、アドホックネットワークを用いて複数のロボット間で通信による情報交換を行う事により、未知の地域をロボットに自律的に、効率よく探索させる手法を提案、評価する。

1.3 本論文の流れ

本論分は以下に示す流れで記述する

第 1 章では本研究の概要を述べる

第 2 章ではアドホックネットワークと、既存の災害用ロボットの説明と問題点を述べる

第 3 章では既存の、ロボットの自律的な探索アルゴリズム説明をする

第 4 章では提案手法の説明をする

第 5 章では提案手法の実験について述べる

第 6 章では実験結果

第 7 章では実験結果をうけた考察と今後の課題について述べる

2 アドホックネットワークと既存の災害用ロボット

2.1 アドホックネットワーク

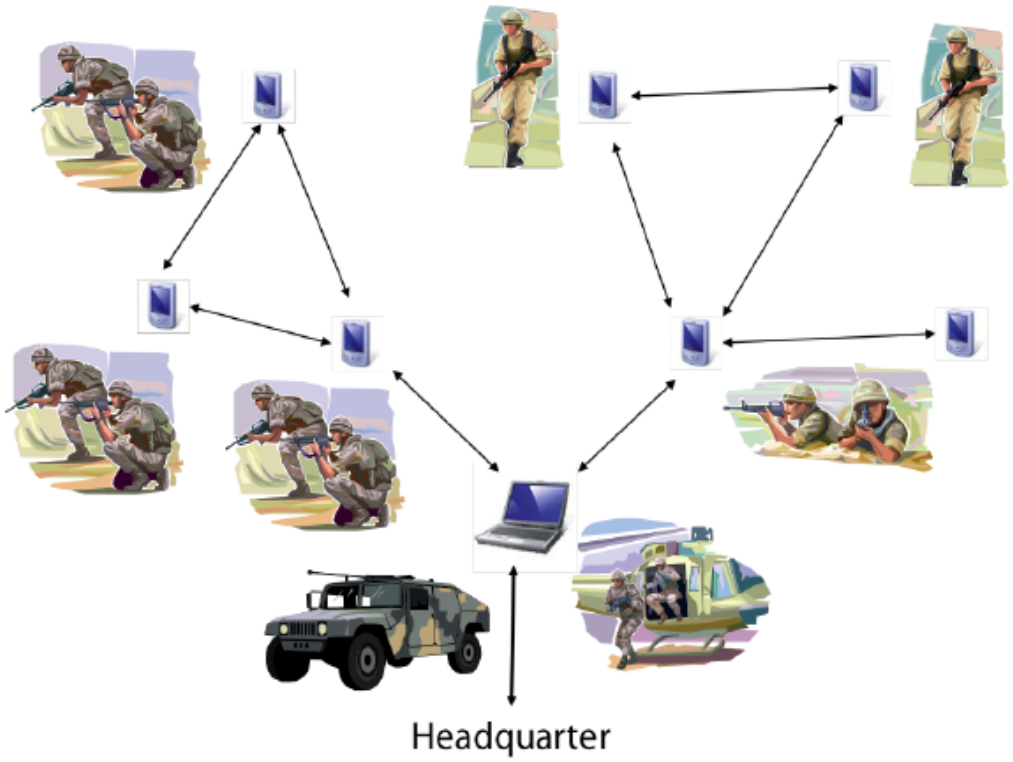

図 3:アドホックネットワークの軍事への応用例

アドホックネットワークは Perkins[2]が提唱したネットワークの形態である。Ad -hoc とはラテン語で「その場限りの」という意味であり、ネットワークの機能を有するノードが各自、他のノードとその場限りでネットワークを形成して情報の共有をしたり、マルチホップによって遠くのノードに情報を渡すといった使い方をする。このネットワーク形態は、第一に軍事的な応用を目指していたが、現在では企業や各大学において研究や商品開発に活かされている。

上図 3 は、アドホックネットワークを軍事的に適用する1つの例である。各兵士が通信可能な端末を携帯しており、近くにいる仲間とその場限りでネットワークをつなぎ、得られた敵の情報や自陣の情報をマルチホップを用いて本部(Headquarter)に伝える。

本論文では地震等によってインフラが破壊された地域でのロボットによる探索を考えているので、アドホックネットワークはまさしく、このような条件下で複数のロボットを連携させるのに最適な概念である。

2.2 既存の災害用ロボット

日本国内において、東北地方太平洋沖地震の際に注目された災害用ロボットについて紹介する。

2.2.1 能動スコープカメラ

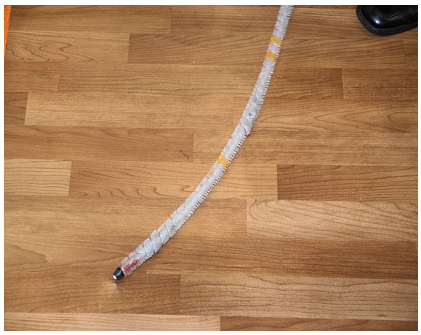

図 4:能動スコープカメラ

東北大学の田所教授が開発した建物内の探索を目的としたロボットである[3]。先端にカメラがついており、ロボットは繊毛で被われている。バイブレータによって繊毛を振動させ、蛇のようにロボットを操作することができる。無線で操縦が可能である。図 4 に能動スコープカメラの前部の写真を示す。

2.2.2 クインス

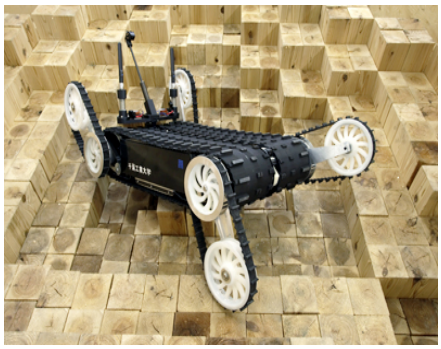

図 5:クインス

千葉工業大未来ロボット技術研究センターの小柳栄次副所長や東北大の田所諭教授が作成した、クローラで動くタイプのロボット。

CO2 探知機、アドホックネットワークの機能、探索地域の 3 次元マッピングなど、高機能で高性能なロボットである[4]。図 5 にクインスの写真を示す。

日本国内における災害用ロボットを2つ紹介した。いずれも高い性能を有していることがわかる。しかし、上記に紹介したロボットは共に、無線による人の操作を必要とする。人間によってロボットを操作することは、ロボットの動作の最適化という観点では極めて効率的である。しかしながら、大規模な災害があった場合、広い被災地をカバーするために大量のロボットが導入されると、導入したロボット分の、大量の操縦者が必要となる。今後も東北地方太平洋沖地震のような大規模地震の発生が予想されるが、広い地域に被害が及んだ際に、大量のロボットを導入するとなると、その分、大量の操縦者が必要となり、救助に加わえるべき人員を圧迫する可能性がある。操縦者の人員の確保などを考えると、そのような体制では今回のような大災害では人手不足が予測されるため、対応できる災害はかなり限定的になる。したがって、大規模災害の対応を視野に入れると、ロボットを自律的に動かすようにしたほうが効率が良い。自律的な行動をさせれば、操縦者として確保する人員は削減でき、その分を救助に向かわせることもできる。このことから、本論文では、被災地の探索を自律的に行う手法について考える。

3 既存の探索アルゴリズム

3.1 Frontier Based Exploration

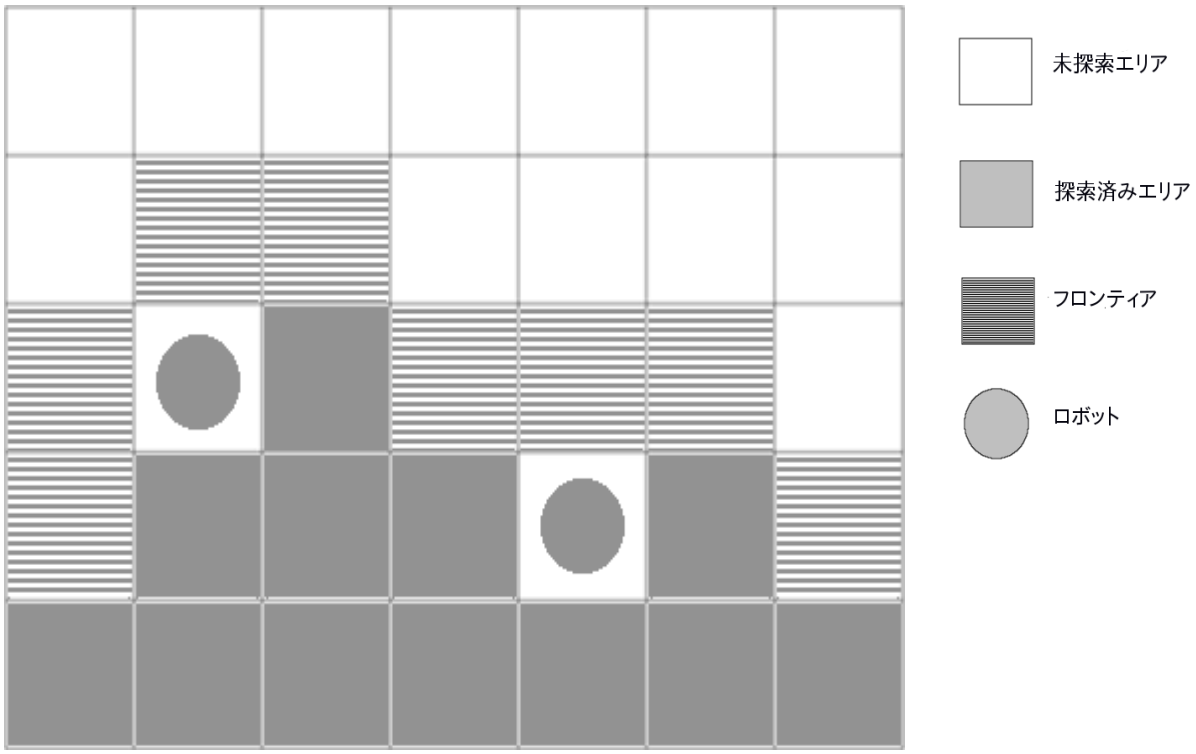

図 6:Frontier Based Exploration

ロボットにとって未知の地域を自律的に探索するための既存の手法として、Yamauchi が提案した Frontier Based Exploration[5]がある。

これは、ロボットが探索した場所と、探索していない場所との境界を「Frontier」と定義し、ロボットを Frontier に向かわせるように仕組むことで、未探索エリアを探索させ、徐々に探索範囲を広げていくという概念である。

例えば図 6 のようなケースになると、丸で表されたロボットは縞模様のフロンティアに積極的に移動し、探索範囲を広げようと動く。

3.2 Frontier Based Exploration の災害現場への適用

この技術を災害現場などにおける探索に適用した研究として、Martijn N. Rooker らの研究[6]がある。

危機的な現場での探索において、ロボットが得たリアルタイムの情報が司令部に届けられるか、ということは大変に重要な要素である。したがって彼らは、そういった現場における探索において、ロボットが探索によって得た情報は素早く基地局に送られる必要があると考え、Frontier Based Exploration の手法に、「通信維持」という制約を加えた。具体的には、探索中にすべてのロボットが、マルチホップを用いて、基地局と通信ができる状態を常に保つように、ロボットの動きに制限をかけた。そして、上記の Frontier Based Exploration のアルゴリズムをベースに、主に 4 台のロボットと1つの基地局を無線でつなぎながら、自動的に平面を探索させるシミュレーションを行った。図 7 にその様子を示す。図 7 では、4 台のロボットが無線での通信を介して鎖のようにつなぐことで、マップの奥まで探索している様子が示されている。

彼らの実験の結果、基地局とロボットの間の通信を常に維持させながら探索を行うことに成功した。しかし、一方で、通信の維持を堅持させつつ探索を行うと、ロボットの台数にもよるが、探索対象の現場全体を探索する

ことができなくなった。さらに、探索範囲を広げようとロボットの台数を増やすと、ロボットの行動の最適化が難しくなり、未探索のエリアにロボットが中々足を踏み入れることができなくなったため、投入しているロボット数で探索できる最大範囲すら達成することが困難になるという欠点が明らかとなった。

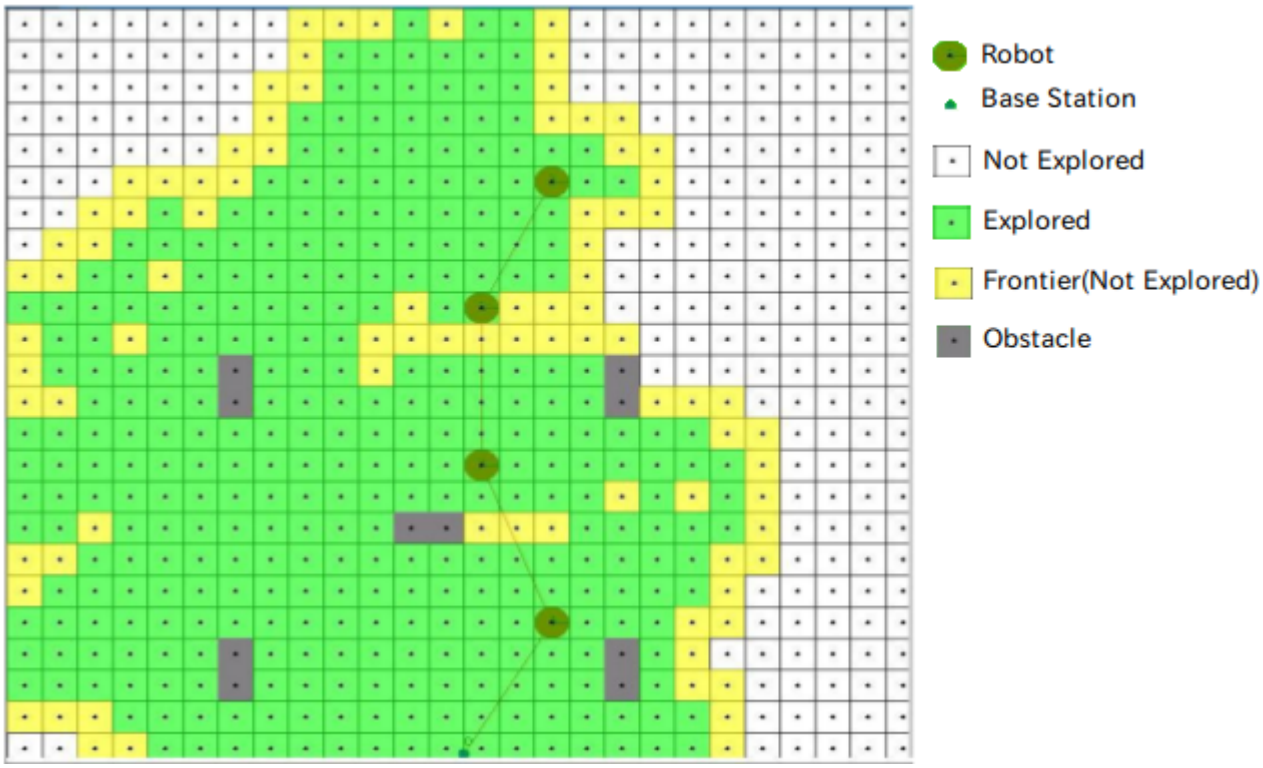

図 7:Martin.N.Rooker らの実験におけるシミュレーションの様子

4 提案手法

既存技術の欠点である、通信維持の制約による探索範囲の縮小の解決と、ロボットの最適な行動の向上を実現するため本論文では、探索中に通信を中継するノードを配置することを提案する。

4.1 提案手法 1

提案手法1では、図 8 のようにロボット全体が、各ロボットを結ぶ線がほぼ直線を描くフォーメーションをとった時に、基地局から一番遠い位置にいるロボットが中継ノードを置く(図 9)。これによって、ロボットが探索できる範囲を伸ばすことができる。

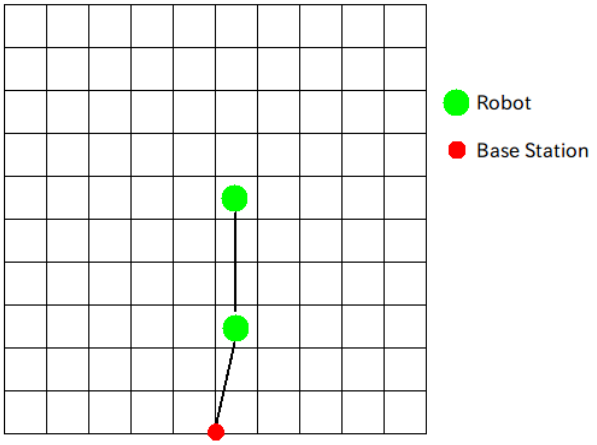

図 8:ロボットの配置が直線になった瞬間

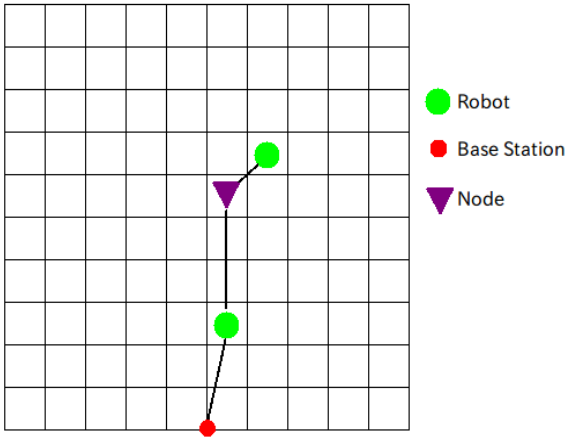

図 9:基地局から遠い位置にいるロボットが中継ノードを設置

4.2 提案手法 2.1

提案手法 2.1 では、提案手法 1 と同様、ロボット全体が図 10 のような直線状のフォーメーションをとった際、基地局から一番近い位置にいるロボットが中継ノードを置く(図 11)。この手法の場合、基地局が探索地域の内部に入ることと同様の効果が得られ、イメージとしては探索範囲を下から押し広げるようになる。

提案手法 1 との違いは中継ノードを設置するロボットが近いか遠いか、ということである。

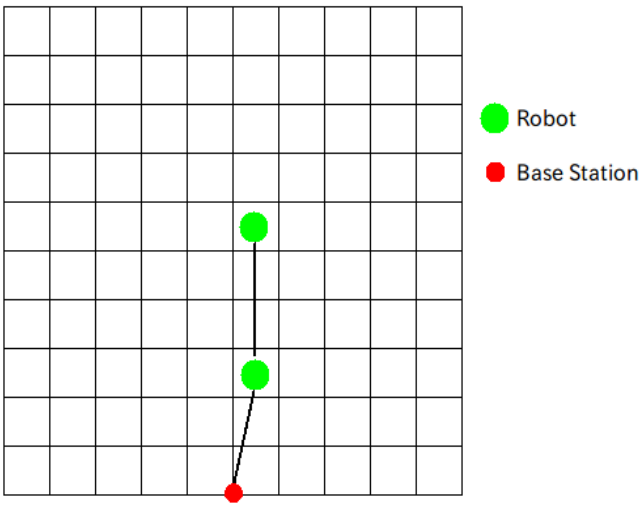

図 10:ロボットの配置が直線になった瞬間

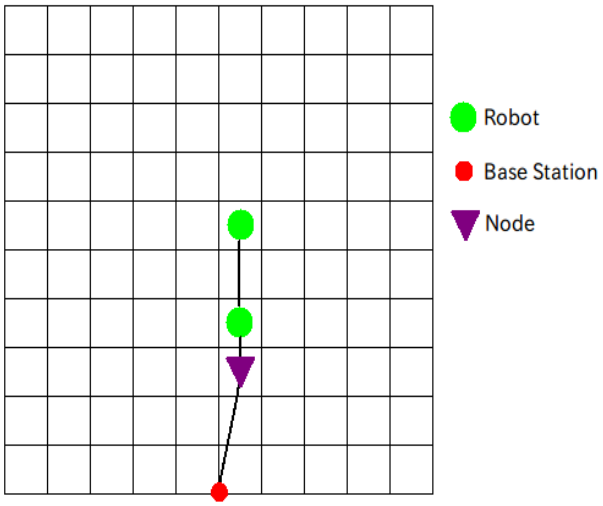

図 11:基地局に一番近いロボットが、自分の位置に中継ノードを置く

4.3 提案手法 2.2

提案手法 1 と提案手法 2.1 では、中継ノードを設置する条件として、基地局と基地局から最も遠いロボットとの距離を測って、直線になっているか否かを判断した。この方法では、中継ノードが増える度に、中継ノードの通信半径分だけ、基地局-ロボット間の最大距離は伸びることになる。したがって、探索範囲の幅が通信半径よりも十分に大きい場合は、一旦ロボットが探索範囲の上端に到達した際、中継ノード設置条件の距離は長くなる一方、それ以上ロボット-基地局間の最大距離が増えることはありえなくなるため、中継ノードを置く条件が満たされなくなる恐れがある。その改善案として、ロボットの直線配置の検出を緩くし、探索中に中継ノードを等間隔で設置するようなアルゴリズムにすることで、結果的に探索カバー率を上げるという方法が考えられる。しかしそれでは、ロボット間の距離が短く、ある箇所に密集している状態でも中継ノードを置く可能性があるため、ロボットの次ステップの移動先が重複してしまい、探索効率が落ちてしまう可能性がある。従って、提案手法 1 及び提案手法 2.1 と比べて、中継ノードを置くタイミングを変える事を考えた。

提案手法 2.2 では、最初にロボットのフォーメーションが、各ロボットを結ぶ線が直線になるようになっているかをチェックし、その条件が満たされたら、基地局に一番近いロボットが、今自分が属している基地局・中継ノードの通信限界のすぐ内側に位置しているかをチェックする。それが満たされて始めて中継ノードを設置する。
提案手法 1 及び提案手法 2.1 では、ロボットが直線になっているかどうかの判定に、「基地局から最も遠いロボットが、現在使われている中継ノードの個数上到達できる最大距離にいるか」という判断を用いているが、この手法ではロボットの直線的なポジショニングの判定に基地局の情報を用いず、ロボット間のみの距離を使っている。

図 12 に示しているのは本提案で中継ノードが設置される一例である。図中の(1),(2)はそれぞれ
(1):ロボット同士がほぼ直線状になるような配置になっている。
(2):基地局に最も近いロボットが、基地局あるいは中継ノードの通信限界のすぐ内側に位置している
ということを表している。

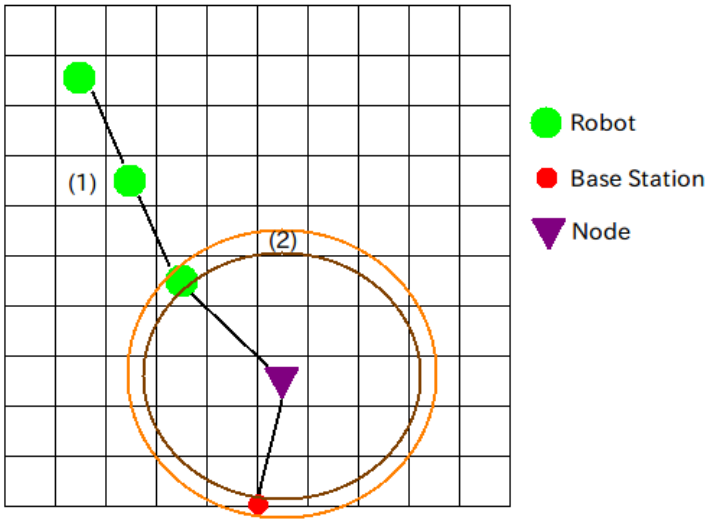

図 12:提案手法 2.2 における、中継ノード設置の条件を満たしている状態

5 既存及び提案手法のアルゴリズムの実装

提案手法による探索におけるロボットの遷移の把握と既存手法と提案手法との探索効率の比較を行うため、各手法のシミュレーションを行う。この章では、実験で用いるロボットの移動アルゴリズム及び、既存手法、提案手法のアルゴリズムの実装について述べる。

5.1 のロボットの移動アルゴリズム及び 5.2 のロボットの通信については、Martin.N.Rooker らの実験に使用されたものを採用する。

5.1 既存手法

Martin らの手法の概要を以下に示す。

5.1.1 前提条件

ロボットの移動速度は一定とする。各ロボットの移動は、全てのロボットが同じタイミングで行われる。ロボットが移動するエリアは 2 次元平面をセルで分割し、エリア上には障害物を配置する。ロボットは探索対象の地域とそうでない地域の境界がわかるということとし、ロボットが探索対象のマップからロストしないようにプログラムで設定する。なお、シミュレーションでは探索のマップのみを表示する。移動アルゴリズムは Martin らの手法を採用するが、その手法においては探索中にロボットが基地局と通信が途絶えることは起こらないため、事実上、探索は基地局との通信を前提として行われる。

5.1.2 移動アルゴリズム

時間 t におけるロボット i の位置を $P[i](x_i, y_i)$ で表すとする。各ロボットが次に移動する先は、「その場に留まる」という場合も含めると、図 13 にあるように 9 通りあり、各ロボットは等確率でその移動先を決める。

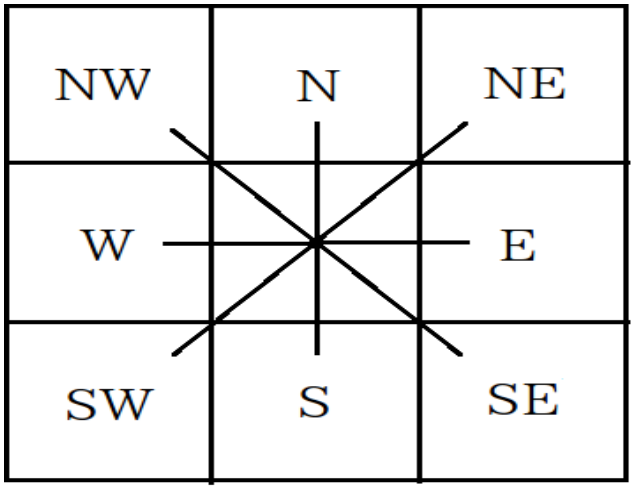

図 13:ロボットが次ステップに移動可能な方向

従ってロボットが n 台投入されると、一回の移動に際して 9^n 通りのロボットの配置が考えられる。このように複数の移動パターンがあるため、ロボットの移動先の状態によってロボット一台一台に重み(Utility と呼ぶ)をつけ、各配置パターンの優劣を決めて結果的に1つのパターンに絞る。

ロボット i が移動する先の位置を $P'[i](x'_i, y'_i)$ とすると、移動先の状態によって以下の重みをつける

$$U(P'[i](t))=\begin{cases} -100 & \text{行き先に障害物・複数ロボットと行き先が同じ場合} \\ -10 & \text{基地局との通信が途絶える場合} \\ 1 & \text{前線に向かう場合} \\ 0 & \text{既に探索された場所に行く・同じ場所に留まる場合} \end{cases}$$

$1 \le i \le n$

その後、各配置パターンに対して

$$Utility = \sum U(P'[i](t))$$

を計算し、この値が最も高い配置パターンを採用する。

例えば、ロボットが 3 台(N1,N2,N3)いるとする。

ある時間を t とすると、その次の時間 t+1 の時に各ロボットが移動する先の候補として、簡単のため以下の 2 種類を挙げるとする。

1. N1->前線に行く(+1)
   N2->同じ場所に留まる(+0)
   N3->通信が途絶える(-10)
   総和:1+0-10=-9
2. N1->同じ場所に留まる(+0)
   N2->同じ場所に留まる(+0)
   N3->前線に行く(+1)
   総和:0+0+1=1

総和は 1 よりも 2 の方が大きいため、この場合は 2 の移動先に決定される。

本論文では、探索に用いるロボットを 3 台に固定し、この移動先の候補を 9^3 =729 パターンまで考えることにする。また、実験において、探索範囲の底に基地局を置くこととする。

5.2 各ロボットの通信

各ロボットは自分の通信範囲内においてアドホックネットワークを形成し、基地局から遠い位置にいるロボットも、直近にいるロボットを用いてマルチホップにより通信を維持する(図 14)。

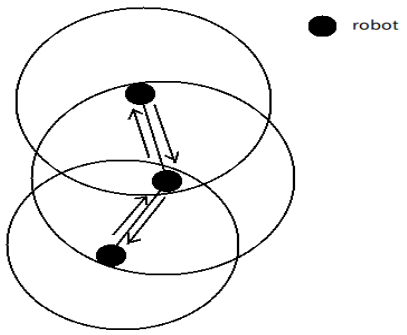

図 14:マルチホップの様子

5.3 提案手法 1

実験におけるロボットの移動及び通信については、以下の 3 つの手法全てに、上記に述べた既存手法を共通で用いる。以下では中継ノードの設置について説明する。

下図 15 に示す例において、基地局と、基地局から最も離れているロボットとの最大距離は 3r。つまり、ロボットの台数を n、通信半径を r とすると、ロボットを結ぶ線が直線となる距離は基地局から測ると nr となる。

したがって、探索中に、基地局から最も遠いロボットとの距離を測り、その値が nr 付近となったときに、基地局から最も遠いロボットが、通信半径 r をもつ中継ノードを自分の位置に設置する。中継ノードを置く際に、別の中継ノードが、他の中継ノードのすぐ近くに設置すると効率が悪い。しかし離れすぎていると中継ノード同士の通信ができず、一番奥にいるロボットの通信を基地局まで中継することができなくなるので、中継ノード設置の条件として、中継ノード同士の距離が r を越えず、かつ r に近くなる距離まで離れていないと設置しないように設定する。図 16 に提案手法 1 の流れ図を示す。

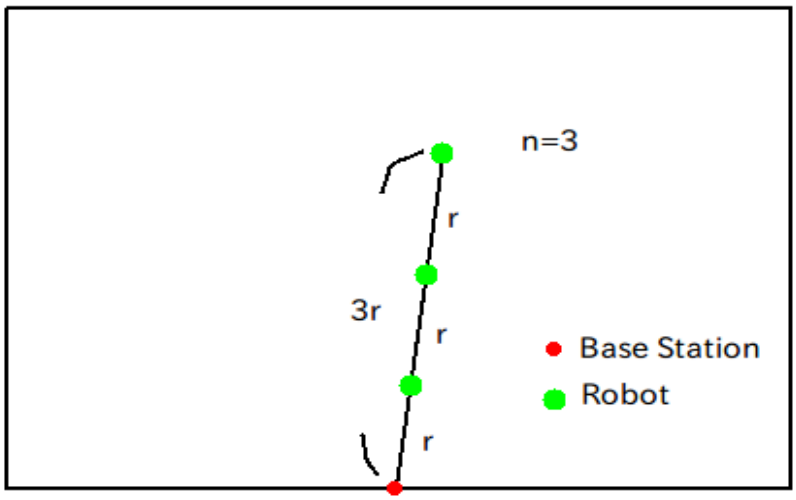

図 15:ロボットと基地局における、距離と通信半径との関係

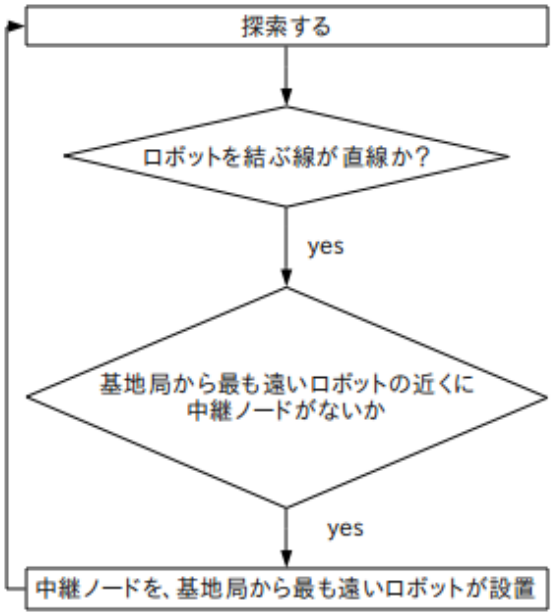

図 16:提案手法 1 のノード設置アルゴリズム

5.4 提案手法 2.1

基地局から最も遠いロボットと基地局との距離を測る部分は、提案手法 1 と同じである。その後に、中継ノードを、基地局及び他の中継ノードから r 離れた場所に、「基地局から最も近い」ロボットが自分の位置に設置する。したがって、設置された中継ノードは基地局との通信を保ったままその場に存在することになる。これを提案手法 2.1 とする。図 17 に提案手法 2.1 の流れ図を示す。

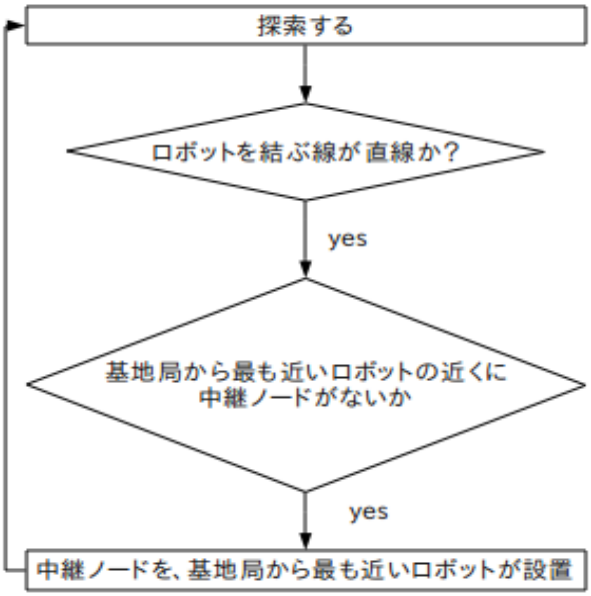

図 17:提案手法 2.1 のノード設置アルゴリズム

5.5 提案手法 2.2

提案手法 2.2 ではロボットと基地局との距離は使わず、通信しているロボット間の距離を測る。ロボットのポジションは下図 19 のような場合が考えられる。中継ノードを置くのに最も適したケースは(a)である。

(b)の場合、点線で示した部分も距離と考えられる。最も適したケース(a)を検出するために、ある 1 台のロボットを基準にし、そのロボットとその他のロボットとの距離を全て測り、その値が $r(n^*n-n)/2$ 付近になれば、第 1 のチェックを満たしたと判断する。次に、基地局から最も近いロボットが、基地局あるいは中継ノードの通信限界のすぐ内側に入っているかどうかを調べ、入っていたら、中継ノードを置く。図 18 に提案手法 2.2 における中継ノード設置までの流れを示す。

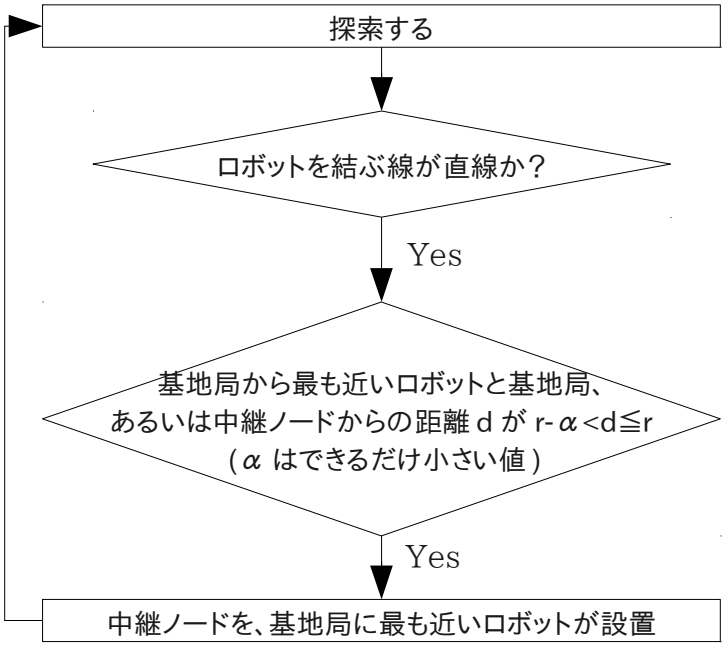

図 18:提案手法 2.2 のノード設置アルゴリズム

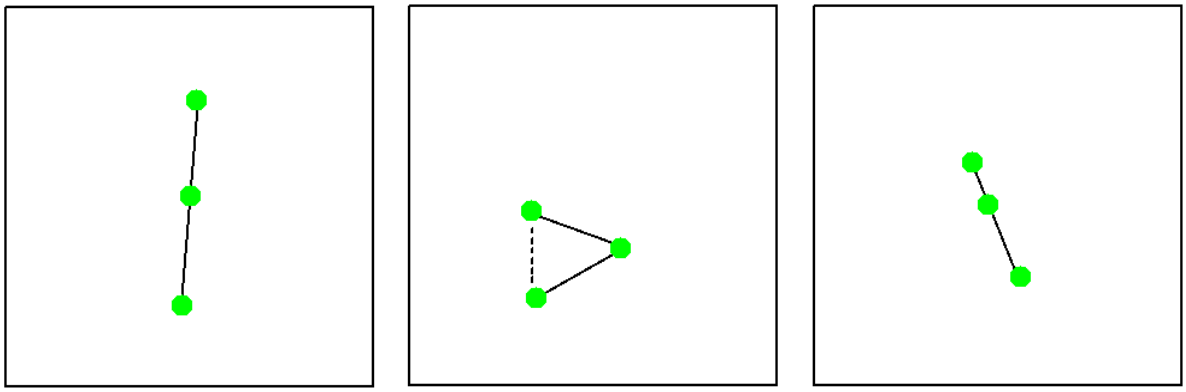

図 19:ロボットのフォーメーションの種類

6 実験結果

実験の条件を表 1 に示す。

表 1:実験条件

探索領域	20×20
障害物の数	4
通信半径	3
ロボットの台数	3
終了条件	10000 ステップ経過するか、全体を探索し終わる
中継ノード数	上限なし

以下に各手法の結果を示す。

6.1 既存技術の状態遷移

はじめに既存技術のシミュレーションにおける探索の状態遷移を図 20、図 21 に示し、結果を図 22 に示す。以下に続く提案手法の結果も、同じような遷移で探索を行っている。

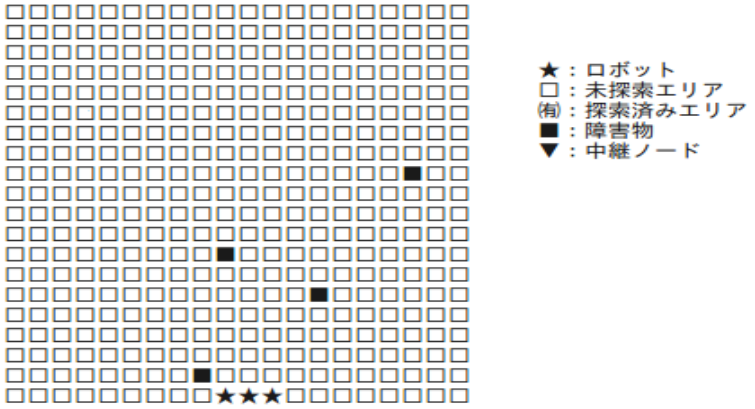

図 20:探索初期の状態

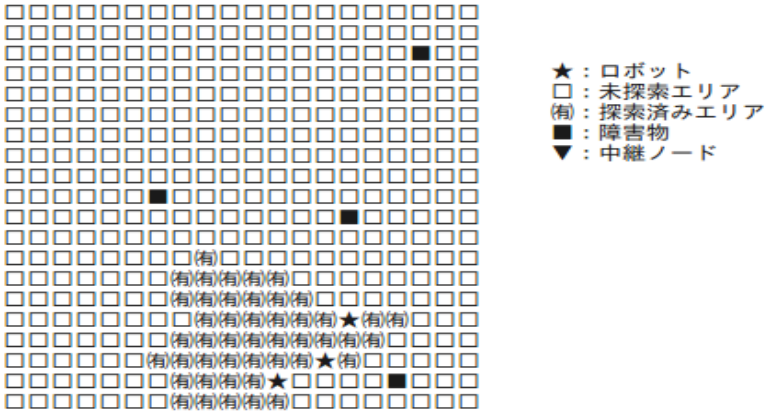

図 21:探索の途中経過の状態

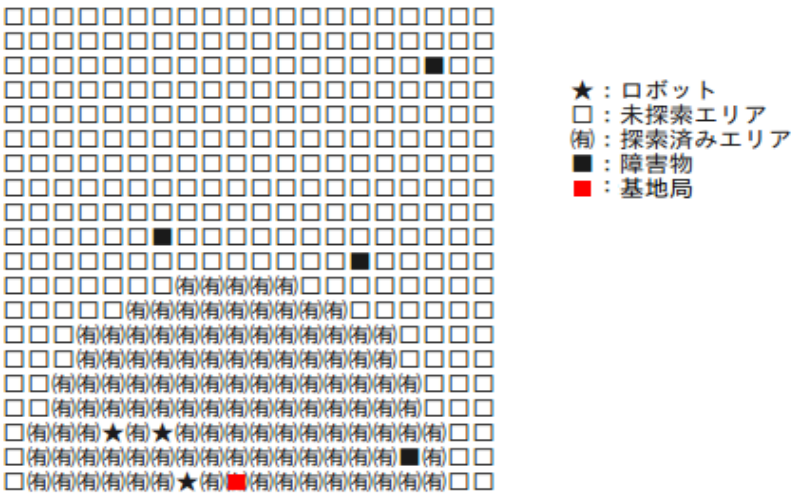

図 22:既存技術の探索結果 カバー率 30.3%

既存技術では探索済み領域は必ず図 22 のような半円を描く。

6.2 提案手法 2.1 の状態遷移の一例

提案手法 2.1 において、中継ノードを設置する毎に、中継ノード追加設置の条件となるロボット-基地局間の距離を r だけ足していった場合の探索結果を図 23～図 25 に示す。通信半径に対して探索範囲が大きい場合、下図 23～25 のように、中継ノードが右に偏る、左に偏る、中央のみに配置される、という 3 通りの場合があった。

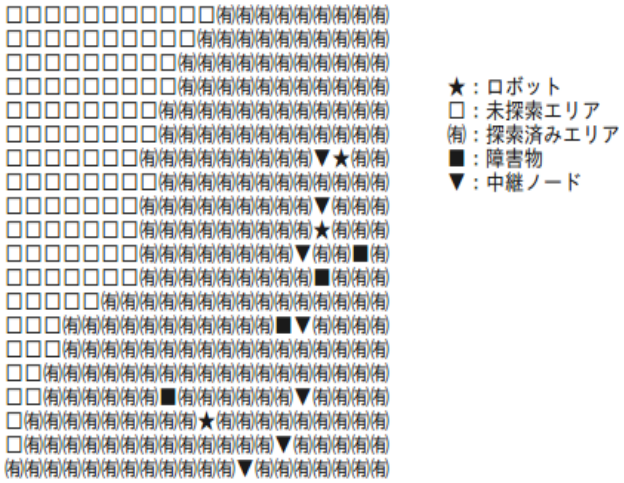

図 23:提案手法 2.1 の結果-右に偏った場合 カバー率 61.62%

図 24:提案手法 2.1 の結果-左に偏った場合 カバー率 61.64%

図 25:提案手法 2.1 の結果-中央に設置された場合 カバー率 82.58%

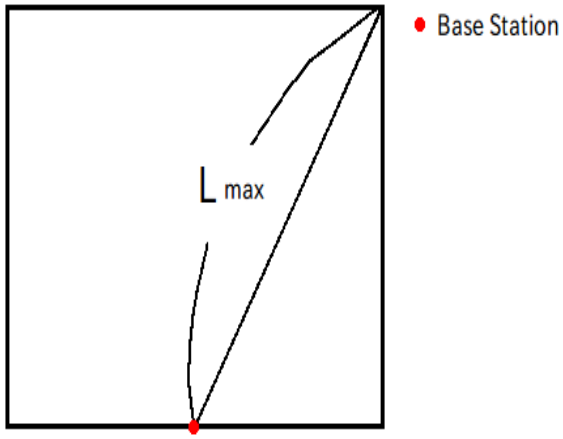

図 26:探索フィールドでとれる最大の距離

シミュレーションで用いている探索フィールドは正方形だが、その場合にロボットと基地局との距離が最大になるのは、図 26 に示す場合である。しかし中継ノードの設置毎に、ノード設置条件であるロボット-基地局間の最大距離を r ずつ増加させていくと、その距離が、図 26 に示す Lmax を越えてしまい、結果、それ以降設置条件を満たせずにノードが置かれなくなる。したがって、上図のように偏った中継ノードの配置になってしまう。

なので、提案手法 1 と提案手法 2.1 における中継ノード設置条件について、ロボット-基地局間の最大距離をノードの数によらず一定とする。つまり、ロボット 3 台かつ中継ノード 0 個の時にとれる最大の距離 3r を設置条件として設定し、ノードを 1 つ設置した後は、基地局-ロボットとの最大距離が 3r 以上である限り、手法毎に決められているノード設置ロボットが中継ノードあるいは基地局との距離が r である場合に中継ノードを設置するようにする。従ってロボットから見ると、中継ノードを等間隔に設置していく作業をしている間に探索をしていくという見方となる。

6.3 提案手法 1 の結果

ノード設置条件を 6.2 の後半に述べたように変更した状態での、提案手法 1 における探索結果の一例を図 27 に示す。

図 27:提案手法 1 の結果

中継ノードの位置は毎回ほぼ同じ位置になる。また、カバー率はバラつき（約 80～100%）がある。

原因としては、実質奥を探索するのは 1 台のロボットになるので、そのロボットの周りが探索済みエリアのみになると、ロボットが探索済みエリアをさまよってしまう可能性が高く、提案手法 1 を複数回行うと、ほぼさまよわず

にうまくいく場合と、さまよってしまったためにうまくいかない場合があるためだと考えられる。

6.4 提案手法 2.1 の結果

提案手法 2.1 における探索結果の一例を図 28 に示す。提案手法 2.1 も、6.2 の後半に述べたノード設置条件に変えている。

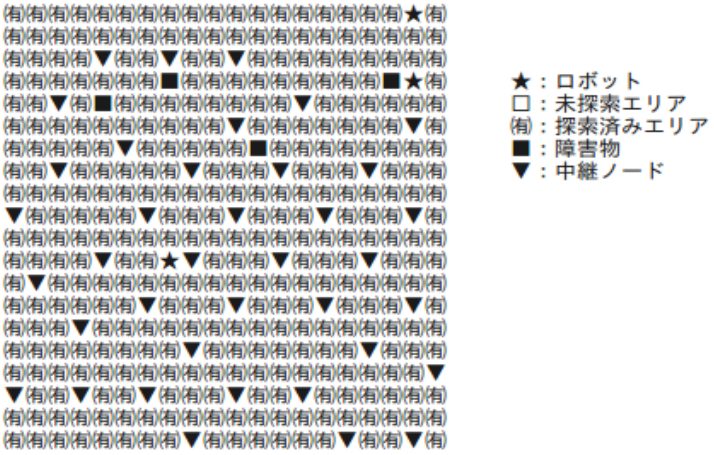

図 28:提案手法 2.1 の結果

カバー率は 80%を越えることが多く、探索のスピードも早かった。各手法のカバー率、探索スピードの統計の結果は 6.6 と 6.7 に掲載した。

6.5 提案手法 2.2 の結果

提案手法 2.2 の実験結果の一例を図 29 に示す。

図 29:提案手法 2.2 の結果

カバー率は提案手法 1 と同様バラつきがある、(約 80～100%)。

6.6 カバー率と使用中継ノード数

既存技術の場合はシミュレーションにおいて 10000 ステップまで探索させれば、探索する範囲とロボット数、また通信半径に依存して決まる最大カバー率は、必ず達成される。

それに対して提案手法の場合は探索時間として 10000 ステップの猶予を与えても、試行毎の達成カバー率は一定ではない。そこで、提案手法 1,提案手法 2.1 及び提案手法 2.2 におけるカバー率の傾向の把握と 3 つの手法との比較を行うため、各手法における 100 回の試行毎の達成カバー率を調べた。また、各手法において、探索毎に使用した中継ノードの数の比較を行うため、使用中継ノード数も調査した。実験条件は表 1 と同様で行った。

得られたデータの上端下端と平均を表 2 に示す。

表 2:提案手法 1、2.1 及び 2.2 の 100 回分の試行結果

	提案手法 1	*提案手法* 2.1	*提案手法* 2.2
カバー率(max/min)(%)	100/58	100/84	100/77
平均カバー率(%)	94	98	93
カバー率の分散	52.79	9.983	37.58
使用ノード数(max/min)(*個*)	39/14	40/22	40/22
平均使用ノード数(*個*)	32	33	29
使用ノード数とカバー率との相関係数	+0.758	+0.656	+0.801

カバー率の分散の値から、探索カバー率は提案手法 2.1 の方が提案手法 1 と提案手法 2.2 よりも安定しているといえる。平均カバー率は提案手法 2.1 が最も高かった。

使用ノード数の平均は提案手法 2.2 が最も少なかった。提案手法 1 における 14 個という数値は、最低カバー率 58%だった際に記録されたものである。

また、使用ノード数と達成カバー率との相関係数の値は、3 つの手法ともに、カバー率が高い時には多めのノードを消費している傾向があるが、同じカバー率であっても使用ノード数が異なるケースが両方の手法ともに見受けられた。

6.7 既存手法、提案手法 1,2.1 及び 2.2 における探索スピードの比較

既存手法、提案手法 1、提案手法 2.1 さらに提案手法 2.2 における探索の早さを比較するため、表 1 と同じ実験条件でシミュレーションを行った。

実験では各手法毎に100回の試行を行い、各ステップ毎の平均を算出し、グラフ化した。図 31 にその結果を示す。

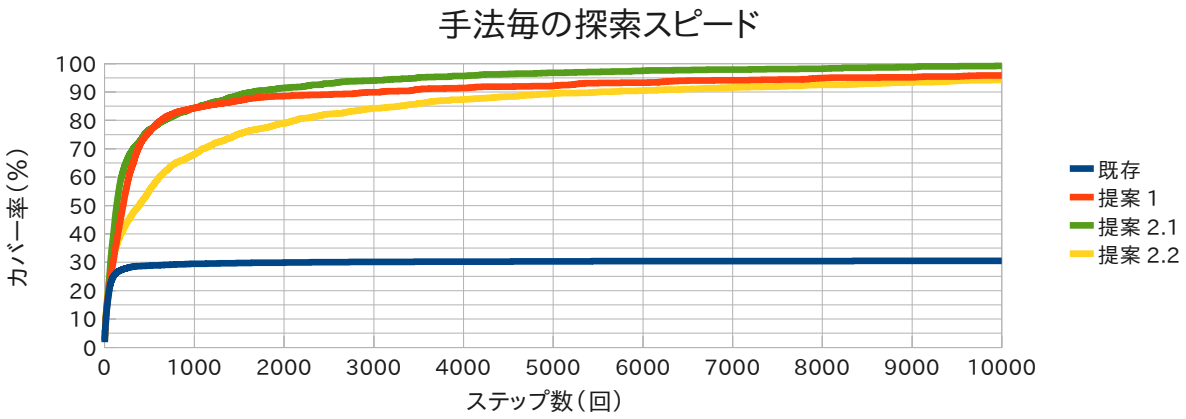

図 31:3 つの手法の探索スピード

提案手法 1、提案手法 2.1 そして提案手法 2.2 とを比較すると、カバー率が 80%に到達するまでに、提案手法 1 は 628 ステップ、提案手法 2.1 は 664 ステップ、提案手法 2.2 は 2124 ステップかかる。

結果を見ると、提案手法 2.2 の探索スピードが比較的遅い。

提案手法 2.2 では、中継ノード設置の条件として、3 台のロボットが図 12 のように直線状に集まる必要がある。したがって、フロンティアに接しているロボットが、遠方の探索のために中継ノード設置を必要としていても、全体のロボットのフォーメーションが図 12 のような形態をとらない限り、中継ノードが置かれない。結果、図 32 のように、全ロボットが概探索エリアに固まってしまうケースが多く見受けられる。ロボットは隣接した 8 近傍のセルの状態しか確認ができないため、運良く 1 台のロボットが未探索エリアに接するまでさまようことになり、その間、事実上探索ができない。従って提案手法 1 と提案手法 2.1 よりも、無駄な行動が増えてしまう。このことから、図 31 のように、探索スピードが遅くなったと考えられる。

被災地における探索においてスピードが重要視されるため、提案手法 2.1 がスピードが早く、最終的なカバー率も高い最も実用的な手法だと思われる。

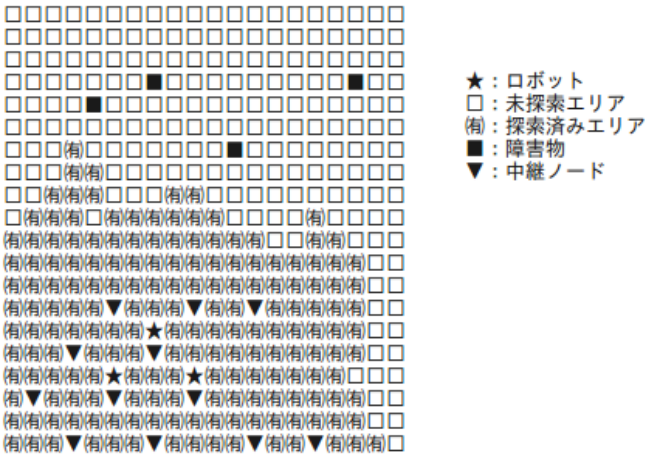

図 32:探索中にロボットが探索済みエリアをさまよっている様子

6.8 ノードの設置位置による 1 ノードあたりのカバー率向上度

提案手法 1 では、ロボット同士が直線上になったとき、基地局から最も遠いロボットがノードを設置する。それに対して提案手法 2 の系統では、基地局から最も近いロボットがノードを設置する。

上記の2つの設置場所の違いによって、ノード設置時にどの程度、探索範囲が広がるかを調査した。

2 つの手法ともに、設置する場所によって探索可能範囲の増加具合に差があった。実験条件を表 3 に、実験結果を図 33,図 34 に示す。

表 3:1 ノードによるカバー率向上度に関する実験条件

探索範囲	20×20
通信半径	3
ロボットの台数	3
中継ノード数	1
探索時間	10000 ステップ

カバー率の増加量は、上記の条件における既存技術の最大カバー率から、どの程度増えたかという表現にしている。つまり、

式:(1ノード設置後のカバー率-既存技術のカバー率)×100/既存技術のカバー率

によって算出している。図中の円は、既存手法で探索した際の概形(ノード 0 個で届く最大の距離)を示し、提案手法 1 の場合は、その円上にノードを置くことになる。提案手法 2 では、基地局と、基地局から最も近いロボットが、ある程度離れている場合にノードを設置するため、図中の橙色の円上にノードを置くことになる。

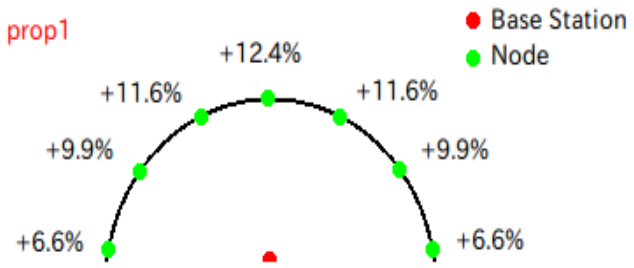

図 33:1 提案手法 1 における 1 ノードあたりのカバー率向上度

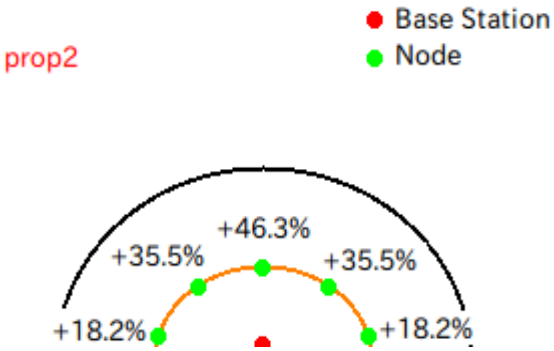

図 34:提案手法 2 における 1 ノードあたりのカバー率向上度

調査によって、基地局から最も近いロボットがノードを設置する方が、ノードの効率がよいことがわかった。特に、基地局を通り、図中の底面に垂直な直線上に中継ノードを置く場合が最もカバー率の増加が大きい。

また、図 33,図 34 のように、ノードの設置位置によってカバー率の増加度が違うため、探索におけるノードの設置場所によって、探索にかかるノードの個数が変化することが推測される。6.5 において、同じカバー率でも使用したノード数に差があるケースがあるというのは、このことが原因であると考えられる。

6.9 実験の考察

提案手法はいずれも、既存技術よりも高いカバー率を得ることができた。

中継ノードの設置の実装について、探索中、必要な時に中継ノードを置くために、現在の中継ノード使用数で実現できる最大の距離を、基地局から最も遠いロボットから基地局までの間で達したら、中継ノードを設置するように設定したが、その場合は探索範囲が広い場合に左右や中央に偏ってしまった(6 章 2 節)。従って、ノード設置の条件から「最大の距離」という制約を外し、単に設置ロボットが、中継ノードと通信できるぎりぎりの距離に位置した時にノードを置くようにすることで、ノードが偏らずに設置することができた。

提案手法 1、提案手法 2.1 は探索のスピードが早い反面、中継ノードの効率とカバー率のばらつきにおいて、提案手法 2.2 よりも劣っていた。

また、提案手法 2.2 は、中継ノードの効率が良く、カバー率も提案手法 1 よりも安定していたが、探索中の無駄な動きが提案手法 1 と提案手法 2.1 よりも多くなる傾向があるため、探索のスピードは提案手法 1、提案手法 2.1 よりも劣ることがわかった。

7 まとめと今後の課題

7.1 まとめ

大災害によってインフラが壊滅した地域での効率的な探索は重要である。そこで本論文では、現在メインで行われている人的な探索や、操縦者を必要とするロボットを使った探索の欠点を指摘し、ロボットによる自動探索の必要性を述べた。

ロボットに自律的に探索させるアルゴリズムを災害現場用にアレンジした研究として Martin らの研究がある。彼らは「ロボットと基地局との通信維持」を重視し、複数のロボットを用いて 2 次元のフィールドを探索させた。結果、通信の維持を保ちながら探索させることに成功したが、一方で探索範囲が狭くなるという欠点が明かになった。

そこで我々は、探索中に中継ノードを設置することによって、ロボットと基地局との通信を維持しながら探索範囲を拡大できると考え、3 つの探索手法を提案し、シミュレーションによって手法の調査を行った。その結果、全提案手法において平均して 90%以上のカバー率が達成できた。災害現場での探索では迅速性が重要であり、その観点からすると、探索中に最も基地局に近いロボットが中継ノードを置く提案手法 2.1 が最も探索が早かったため、この手法が提案手法の中で一番実用的なものであると考えられる。

7.2 今後の課題

今後の課題として、

・3 次元の空間についてのシミュレーション
・バッテリーを考慮した提案手法の改良
・ロボットの行動の効率化と無駄な探索の削減を目指すため探索アルゴリズムそのものの見直し
・探索領域の形をより多様（三角形や楕円など）にした際の考察

以上を考えている。

謝辞

本研究を進めるにあたり、研究や研究以外についても様々な助言、指導をして下さった三好教授に心から御礼申し上げます。

また、研究の助言や議論をして下さり、また、日常生活でもお世話になった、三好研究室 4 回生の皆さん及び大学院生の方々に、心から御礼申し上げます。

参考文献

[1]警察庁-東日本大震災について,
http://www.npa.go.jp/archive/keibi/biki/index.htm

[2]Charles E. Perkins,
Ad Hoc Networking, (Addison-Wesley, 2001)

[3]東北大学田所研究室,
http://www.rm.is.tohoku.ac.jp/index.php?Tadokoro%20Laboratory

[4]千葉工業大学未来ロボット技術研究センター,
http://www.furo.org/ja/robot/quince/index.html

[5]Brian Yamauchi,
A Frontier-Based Approach for Autonomous Exploration, Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, CA, (July 1997), pp. 146-151

[6]Martijn N. Rooker and Andreas Birk,
Combining Exploration and Ad-Hoc Networking in RoboCup Rescue, Lecture Notes in Computer Science, (2005), vol3276, pp. 236-246

付録　ソースコード

＝既存手法＝

```
#include<stdio.h>
#include<math.h>

#define X 20
#define Y 20
#define r 7
#define N 3
#define Rout 19

int map[Y][X];

struct robot{
  int x;
  int y;
}robot[N];

struct node{
  int x;
  int y;
}node[Rout];

int main()
{
  FILE *fp;
  int i,c,OBSTx,OBSTy,d;

  for(i=0;i<=Y-1;i++)//マップの初期化
   for(c=0;c<=X-1;c++)
    {
     map[i][c]=0;//0 はプレーンな地形
    }
  srand((unsigned)time(NULL));

  for(i=0;i<2;i++)//障害物
    for(c=0;c<2;c++){
      OBSTx = rand() % 20;
      OBSTy = rand() % 19;
      map[OBSTy][OBSTx]=2;
    }

  fp=fopen("move_mk15-map.txt","w");

  for(i=0;i<=Y-1;i++)
    for(c=0;c<=X-1;c++)
      {
                if(map[i][c]==0)
                  fprintf(fp,"□");
                else if(map[i][c]==2)
                  fprintf(fp,"■");
                if(c==X-1)
                  fprintf(fp,"¥n");
      }

  fprintf(fp,"------------------------------------------------------------create map¥n");

  for(i=0;i<N;i++){
    robot[i].x=9+i;
    robot[i].y=19;
  }

  for(i=0;i<N;i++){
    for(c=0;c<N;c++){
      robot[i].list[c]=1;
    }
  }

  int marukaita=0;

  for(i=0;i<=Y-1;i++){
    for(c=0;c<=X-1;c++)
      {
                for(d=0;d<N;d++){
                  if(map[i][c]==0 && i==robot[d].y && c==robot[d].x){
                    fprintf(fp,"★");
                    marukaita=1;
                  }
                }
                if(marukaita==1)
                  marukaita=0;
                else if(map[i][c]==0)
                  fprintf(fp,"□");
                else if(map[i][c]==2)
                  fprintf(fp,"■");
                if(c==X-1)
                  fprintf(fp,"¥n");
      }
  }
  fprintf(fp,"------------------------------------------------------------begin¥n");
  int experience[Y][X];//行ったことのある座標リスト。例えば(x,y)=(8,2)に行ったことあるなら、
exp[2][8]=1
  int ifX[N],ifY[N],temX[N],temY[N];//if~:仮の座標 tem~:ある時点で最も高いユーティリ
ティを持つ移動先座標
  int u=0;
  int tempu=0;
  int comp=0;//comp が 1 になったら、探索終了
  int exp=0;
  int step=0;//exp は、探索効率計算用、ステップは、探索までのステップ数
  int kaiten;
  int k=729;
  float rate;//探索率を格納
  FILE *fpr;//探索効率調査用
  int perm[N];
  int n;
  int NorWes[]={4,5,7,8};
  int Nor[]={3,4,5,6,7,8};
  int NorEas[]={3,4,6,7};
  int Wes[]={1,2,4,5,7,8};
  int Eas[]={0,1,3,4,6,7};
  int SouWes[]={1,2,4,5};
  int Sou[]={0,1,2,3,4,5};
  int SouEas[]={0,1,3,4};
  int connec[N];//基地局との通信確立されてる->1,基地局の通信範囲外->0
  int inrad=0;
  int toa,verifi[N],nodeverifi[Rout];
  int No=0;
  int qua;
  int dis=0;
  FILE *veri;

  for(n=0;n<N;n++){
   verifi[n]=0;
  }


  for(i=0;i<=Y-1;i++)
    for(c=0;c<=X-1;c++)
      experience[i][c]=0;

  for(n=0;n<N;n++)
    experience[robot[n].y][robot[n].x]=1;

  fpr=fopen("stati.txt","w");
  srand((unsigned)time(NULL));

  FILE *move;
  move=fopen("robot(1)loca.txt","w");
  veri=fopen("verifi.txt","w");

  while(1){

    for(kaiten=0;kaiten<k;kaiten++){
                  for(n=0;n<N;n++)
                  {
                    perm[n]=rand() % 9;
                  }


                  for(n=0;n<N;n++){
                    if(robot[n].x==0 && robot[n].y==0){//フィールドの左上にいる時
                      perm[n]=NorWes[rand()%4];}
                    else if(robot[n].x==19 && robot[n].y==0){//フィールドの右上
                      perm[n]=NorEas[rand()%4];}
                    else if(robot[n].x==0 && robot[n].y==19){//フィールドの左下の時
```

```
            perm[n]=SouWes[rand()%4];}
          else if(robot[n].x==19 && robot[n].y==19){//フィールドの右下の時
            perm[n]=SouEas[rand()%4];}
          else if(robot[n].y==0){//フィールドの上側
            perm[n]=Nor[rand()%6];}
          else if(robot[n].x==0){//フィールドの左側
            perm[n]=Wes[rand()%6];}
          else if(robot[n].x==19){//フィールドの右側
            perm[n]=Eas[rand()%6];}
          else if(robot[n].y==19){//フィールドの下側
            perm[n]=Sou[rand()%6];}
        }

        for(n=0;n<N;n++){
          ifX[n]=robot[n].x;
          ifY[n]=robot[n].y;

        switch(perm[n]){
        case 0: //北西
          ifX[n]=robot[n].x-1;
          ifY[n]=robot[n].y-1;
          break;
        case 1: //北
          ifX[n]=robot[n].x;
          ifY[n]=robot[n].y-1;
          break;
        case 2: //北東
          ifX[n]=robot[n].x+1;
          ifY[n]=robot[n].y-1;
          break;
        case 3: //西
          ifX[n]=robot[n].x-1;
          ifY[n]=robot[n].y;
          break;
        case 4: //キープ
          ifX[n]=robot[n].x;
          ifY[n]=robot[n].y;
          break;
        case 5: //東
          ifX[n]=robot[n].x+1;
          ifY[n]=robot[n].y;
          break;
        case 6: //南西
          ifX[n]=robot[n].x-1;
          ifY[n]=robot[n].y+1;
          break;
        case 7: //南
          ifX[n]=robot[n].x;
          ifY[n]=robot[n].y+1;
          break;
        case 8: //南東
          ifX[n]=robot[n].x+1;
          ifY[n]=robot[n].y+1;
          break;
        }
        }

for(n=0;n<N;n++){
        if(experience[ifY[n]][ifX[n]]==0)//行ったことないところなら高評価
          tempu=tempu+1;
        else if(experience[ifY[n]][ifX[n]]==1)//行ったことあるところなら 0 点
          tempu=tempu+0;

        if(map[ifY[n]][ifX[n]]==2){//障害物のマスなら低評価
          tempu=tempu - 100;
        }

        if(map[ifY[n]][ifX[n]]==3){//ロボットの位置が中継ノードとかぶってたらダメ
          tempu=tempu - 100;
        }
}

for(n=0;n<N;n++){
        for(c=0;c<N;c++){
          if(c==n)
            printf("");
          else if(ifX[n]==ifX[c]&&ifY[n]==ifY[c]){
          tempu=tempu-100;
          }
}
}
```

```
    for(n=0;n<N;n++){
            verifi[n]=0;
    }

    for(n=0;n<N;n++){
            if(sqrt(pow(ifY[n]-19,2)+pow(ifX[n]-9,2))<r)
              verifi[n]=1;
    }

    for(toa=0;toa<10;toa++){
            for(n=0;n<N;n++){
              if(verifi[n]==0){
                for(qua=0;qua<N;qua++){
                  if(qua==n){
                          printf("");}
                  else if(verifi[qua]==0)
                          printf("");
                  else{
                          if(verifi[qua]==1 && sqrt(pow(ifY[n]-ifY[qua],2)+pow(ifX[n]-
ifX[qua],2))<=r)
                            verifi[n]=1;
                  }
                }
              }
            }
    }

    toa=0;
    for(n=0;n<N;n++)
            toa=toa+verifi[n];

    if(toa<N){
            tempu=tempu-100;
    }

    if(tempu>u){
            u = tempu;
            for(n=0;n<N;n++){
              temX[n] = ifX[n];
              temY[n] = ifY[n];
            }

    }
    tempu=0;
   }

   for(n=0;n<N;n++){
    verifi[n]=0;
   }

   for(n=0;n<Rout;n++)
    nodeverifi[n]=0;

   toa=0;
   u=-1;
   tempu=0;
   for(n=0;n<N;n++){
    robot[n].x=temX[n];
    robot[n].y=temY[n];
    experience[robot[n].y][robot[n].x]=1;
   }
   printf("robot(1)[%d,%d],robot(2)[%d,
%d]¥n",robot[0].y,robot[0].x,robot[1].y,robot[1].x);

   fprintf(move,"(%d,%d)¥n",robot[1].y,robot[1].x);
   for(i=0;i<=Y-1;i++){
    for(c=0;c<=X-1;c++)
            {
              for(d=0;d<N;d++){
                if(map[i][c]==0 && i==robot[d].y && c==robot[d].x){
                  fprintf(fp,"★");
                  marukaita=1;
                }
              }
              if(marukaita==1)
                marukaita=0;
              else if(map[i][c]==3)
                fprintf(fp,"▼");
              else if(experience[i][c]==1)
                fprintf(fp,"侑");
```

```
                    else if(map[i][c]==0)
                      fprintf(fp,"□");
                    else if(map[i][c]==2)
                      fprintf(fp,"■");
                    if(c==X-1)
                      fprintf(fp,"¥n");
                   }
    }

    for(i=0;i<=Y-1;i++)
      for(c=0;c<=X-1;c++)
                exp = exp + experience[i][c];

    printf("exp=((%d))¥n",exp);

    rate=exp/396.0*100.0;

    if(exp==396)
      comp=1;
    else
      exp=0;

    fprintf(fp,"----------------------------------------------------------step%d(%.2f)¥n",step+1,rate);

    fprintf(fpr,"%d %.2f¥n",step+1,rate);

    if(comp==1)
      printf("-finish-¥n");

    if(comp==1)
      break;

    printf("%d¥n",No);
    step=step+1;
    if(step==10000)
      break;
  }

  fclose(fp);
  fclose(fpr);
  fclose(move);
}
```

＝提案手法１＝

```
#include<stdio.h>
#include<math.h>

#define X 20
#define Y 20
#define r 3
#define N 3
#define Rout 99

int map[Y][X];

struct robot{
  int x;
  int y;
}robot[N];

struct node{
  int x;
  int y;
}node[Rout];

int main()
{
  FILE *fp;
  int i,c,OBSTx,OBSTy,d;

   for(i=0;i<=Y-1;i++)//マップの初期化
    for(c=0;c<=X-1;c++)
      {
      map[i][c]=0;//0 はプレーンな地形
      }
   srand((unsigned)time(NULL));

   for(i=0;i<2;i++)//障害物
     for(c=0;c<2;c++){
       OBSTx = rand() % 20;
       OBSTy = rand() % 19;
       map[OBSTy][OBSTx]=2;
     }

   fp=fopen("move_mk16-map.txt","w");

   for(i=0;i<=Y-1;i++)
     for(c=0;c<=X-1;c++)
       {
                  if(map[i][c]==0)
                    fprintf(fp,"□");
                  else if(map[i][c]==2)
                    fprintf(fp,"■");
                  if(c==X-1)
                    fprintf(fp,"¥n");
       }

   fprintf(fp,"------------------------------------------------------------create map¥n");

   for(i=0;i<N;i++){
     robot[i].x=9+i;
     robot[i].y=19;
   }

   for(i=0;i<N;i++){
     for(c=0;c<N;c++){
       robot[i].list[c]=1;
     }
   }

   int marukaita=0;

   for(i=0;i<=Y-1;i++){
     for(c=0;c<=X-1;c++)
       {
                  for(d=0;d<N;d++){
                    if(map[i][c]==0 && i==robot[d].y && c==robot[d].x){
                      fprintf(fp,"★");
                      marukaita=1;
                    }
                  }
                  if(marukaita==1)
                    marukaita=0;
                  else if(map[i][c]==0)
                    fprintf(fp,"□");
                  else if(map[i][c]==2)
                    fprintf(fp,"■");
                  if(c==X-1)
                    fprintf(fp,"¥n");
       }
   }
}
fprintf(fp,"------------------------------------------------------------begin¥n");
int experience[Y][X];
int ifX[N],ifY[N],temX[N],temY[N];
int u=0;
int tempu=0;
int comp=0;
int exp=0;
int step=0;
int kaiten;
int k=729;
float rate;
FILE *fpr;
int perm[N];
int n;
int NorWes[]={4,5,7,8};
int Nor[]={3,4,5,6,7,8};
int NorEas[]={3,4,6,7};
int Wes[]={1,2,4,5,7,8};
int Eas[]={0,1,3,4,6,7};
int SouWes[]={1,2,4,5};
int Sou[]={0,1,2,3,4,5};
int SouEas[]={0,1,3,4};
int connec[N];
int inrad=0;
int toa,verifi[N],nodeverifi[Rout];
int No=0;
int qua;
int dis=0;
FILE *veri;

for(n=0;n<N;n++){
  verifi[n]=0;
}

for(n=0;n<Rout;n++)
  nodeverifi[n]=0;

for(i=0;i<=Y-1;i++)
  for(c=0;c<=X-1;c++)
    experience[i][c]=0;

for(n=0;n<N;n++)
  experience[robot[n].y][robot[n].x]=1;

fpr=fopen("stati.txt","w");
srand((unsigned)time(NULL));

FILE *move;
move=fopen("robot(1)loca.txt","w");
veri=fopen("verifi.txt","w");

while(1){

  for(kaiten=0;kaiten<k;kaiten++){
                 for(n=0;n<N;n++)
                {
                  perm[n]=rand() % 9;
                }

                 for(n=0;n<N;n++){
                   if(robot[n].x==0 && robot[n].y==0){//フィールドの左上にいる時
                     perm[n]=NorWes[rand()%4];}
                   else if(robot[n].x==19 && robot[n].y==0){//フィールドの右上
                     perm[n]=NorEas[rand()%4];}
                   else if(robot[n].x==0 && robot[n].y==19){//フィールドの左下の時
                     perm[n]=SouWes[rand()%4];}
                   else if(robot[n].x==19 && robot[n].y==19){//フィールドの右下の時
                     perm[n]=SouEas[rand()%4];}
                   else if(robot[n].y==0){//フィールドの上側
                     perm[n]=Nor[rand()%6];}
                   else if(robot[n].x==0){//フィールドの左側
                     perm[n]=Wes[rand()%6];}
                   else if(robot[n].x==19){//フィールドの右側
```

```
            perm[n]=Eas[rand()%6];}
          else if(robot[n].y==19){//フィールドの下側
            perm[n]=Sou[rand()%6];}
        }

        for(n=0;n<N;n++){
          ifX[n]=robot[n].x;
          ifY[n]=robot[n].y;

        switch(perm[n]){
        case 0: //北西
          ifX[n]=robot[n].x-1;
          ifY[n]=robot[n].y-1;
          break;
        case 1: //北
          ifX[n]=robot[n].x;
          ifY[n]=robot[n].y-1;
          break;
        case 2: //北東
          ifX[n]=robot[n].x+1;
          ifY[n]=robot[n].y-1;
          break;
        case 3: //西
          ifX[n]=robot[n].x-1;
          ifY[n]=robot[n].y;
          break;
        case 4: //キープ
          ifX[n]=robot[n].x;
          ifY[n]=robot[n].y;
          break;
        case 5: //東
          ifX[n]=robot[n].x+1;
          ifY[n]=robot[n].y;
          break;
        case 6: //南西
          ifX[n]=robot[n].x-1;
          ifY[n]=robot[n].y+1;
          break;
        case 7: //南
          ifX[n]=robot[n].x;
          ifY[n]=robot[n].y+1;
          break;
        case 8: //南東
          ifX[n]=robot[n].x+1;
          ifY[n]=robot[n].y+1;
          break;
        }
        }

for(n=0;n<N;n++){
        if(experience[ifY[n]][ifX[n]]==0)
          tempu=tempu+1;
        else if(experience[ifY[n]][ifX[n]]==1)
          tempu=tempu+0;

        if(map[ifY[n]][ifX[n]]==2){
          tempu=tempu - 100;
        }

        if(map[ifY[n]][ifX[n]]==3){
          tempu=tempu - 100;
        }
}

for(n=0;n<N;n++){
        for(c=0;c<N;c++){
          if(c==n)
            printf("");
          else if(ifX[n]==ifX[c]&&ifY[n]==ifY[c]){
          tempu=tempu-100;
          }
}
}

for(n=0;n<N;n++){
        verifi[n]=0;
}

for(n=0;n<N;n++){
        if(sqrt(pow(ifY[n]-19,2)+pow(ifX[n]-9,2))<=r)
          verifi[n]=1;
    }

    for(toa=0;toa<10;toa++){
            for(n=0;n<N;n++){
              if(verifi[n]==0){
                for(qua=0;qua<N;qua++){
                  if(qua==n){
                        printf("");}
                  else if(verifi[qua]==0)
                        printf("");
                  else{
                        if(verifi[qua]==1 && sqrt(pow(ifY[n]-ifY[qua],2)+pow(ifX[n]-
ifX[qua],2))<=r)
                          verifi[n]=1;
                  }
                }
              }
            }
    }

    if(No>0){
              for(n=0;n<Rout;n++)
                nodeverifi[n]=0;

              for(qua=0;qua<No;qua++){
                for(n=0;n<N;n++){
                  if(verifi[n]==1 && sqrt(pow(node[qua].y-ifY[n],2)+pow(node[qua].x-
ifX[n],2))<=r)
                        nodeverifi[qua]=1;
                }
              }
              for(n=0;n<N;n++){
                if(verifi[n]==0){
                  for(qua=0;qua<No;qua++){
                        if(sqrt(pow(ifY[n]-node[qua].y,2)+pow(ifX[n]-
node[qua].x,2))<=r && nodeverifi[qua]==1)
                          verifi[n]=1;
                  }
                }
              }

    for(qua=0;qua<No;qua++){
            for(n=0;n<No;n++){
              if(n==qua)
                printf("");
              else if(nodeverifi[n]==1 && sqrt(pow(node[qua].y-
node[n].y,2)+pow(node[qua].x-node[n].x,2))<=r)
                nodeverifi[qua]=1;
            }
    }

    for(n=0;n<N;n++){
              if(verifi[n]==0){
                for(qua=0;qua<No;qua++){
                        if(sqrt(pow(ifY[n]-node[qua].y,2)+pow(ifX[n]-
node[qua].x,2))<=r && nodeverifi[qua]==1)
                          verifi[n]=1;
                  }
                }
              }
    }

    toa=0;
    for(n=0;n<N;n++)
            toa=toa+verifi[n];

    if(toa<N){
            tempu=tempu-100;
    }

    if(tempu>u){
            u = tempu;
            for(n=0;n<N;n++){
              temX[n] = ifX[n];
              temY[n] = ifY[n];
            }

    }
    tempu=0;
  }
```

```
    for(n=0;n<N;n++){
                verifi[n]=0;
     }

     for(n=0;n<N;n++){
                if(sqrt(pow(temY[n]-19,2)+pow(temX[n]-9,2))<r)
                  verifi[n]=1;
     }

     for(toa=0;toa<10;toa++){
                for(n=0;n<N;n++){
                  if(verifi[n]==0){
                    for(qua=0;qua<N;qua++){
                      if(qua==n){
                                printf("");}
                      else if(verifi[qua]==0)
                                printf("");
                      else{
                                if(verifi[qua]==1 && sqrt(pow(temY[n]-
temY[qua],2)+pow(temX[n]-temX[qua],2))<r)
                                  verifi[n]=1;
                      }
                    }
                  }
                }
     }

     if(No>0){
                for(n=0;n<No;n++)
                  nodeverifi[n]=0;

                for(qua=0;qua<No;qua++){
                  for(n=0;n<N;n++){
                    if(verifi[n]==1 && sqrt(pow(node[qua].y-
temY[n],2)+pow(node[qua].x-temX[n],2))<=r)
                      nodeverifi[qua]=1;
                  }
                }
                for(n=0;n<N;n++){
                  if(verifi[n]==0){
                    for(qua=0;qua<No;qua++){
                      if(sqrt(pow(temY[n]-node[qua].y,2)+pow(temX[n]-node[qua].x,2))<r
&& nodeverifi[qua]==1)
                                verifi[n]=1;
                    }
                  }
                }
                }

                fprintf(veri,"verifi[0]=%d,verifi[1]=%d,verifi[2]=%d,nodeverifi=
%d¥n",verifi[0],verifi[1],verifi[2],nodeverifi[0]);

     for(n=0;n<N;n++){
       verifi[n]=0;
     }

     for(n=0;n<Rout;n++)
       nodeverifi[n]=0;

     toa=0;
     u=-1;
     tempu=0;
     for(n=0;n<N;n++){
       robot[n].x=temX[n];
       robot[n].y=temY[n];
       experience[robot[n].y][robot[n].x]=1;
     }

     dis=0;
     for(n=0;n<N;n++){
       if(sqrt(pow(robot[n].y-19,2)+pow(robot[n].x-9,2))>=(N-0.3)*r && No<Rout &&
map[robot[n].y][robot[n].x]!=3){
                if(No>0){
                  for(c=0;c<No;c++){
                    if(sqrt(pow(node[c].y-robot[n].y,2)+pow(node[c].x-robot[n].x,2))<r)
                      dis++;
                  }
                  if(dis==0){
                     node[No].x=robot[n].x;
                     node[No].y=robot[n].y;
                     map[node[No].y][node[No].x]=3;
                       No++;
                    }
                  }
                  else{
                    node[No].x=robot[n].x;
                    node[No].y=robot[n].y;
                    map[node[No].y][node[No].x]=3;
                    No++;
                  }
     }
     }

   fprintf(move,"(%d,%d)¥n",robot[1].y,robot[1].x);
   for(i=0;i<=Y-1;i++){
     for(c=0;c<=X-1;c++)
                {
                  for(d=0;d<N;d++){
                    if(map[i][c]==0 && i==robot[d].y && c==robot[d].x){
                      fprintf(fp,"★");
                      marukaita=1;
                    }
                  }
                  if(marukaita==1)
                    marukaita=0;
                  else if(map[i][c]==3)
                    fprintf(fp,"▼");
                  else if(experience[i][c]==1)
                    fprintf(fp,"備");
                  else if(map[i][c]==0)
                    fprintf(fp,"□");
                  else if(map[i][c]==2)
                    fprintf(fp,"■");
                  if(c==X-1)
                    fprintf(fp,"¥n");
                }
   }

   for(i=0;i<=Y-1;i++)
     for(c=0;c<=X-1;c++)
                exp = exp + experience[i][c];

   rate=exp/396.0*100.0;

   printf("%f¥n",rate);

   if(exp==396)
     comp=1;
   else
     exp=0;

   fprintf(fp,"----------------------------------------------------------step%d(%.2f)¥n",step+1,rate);

   fprintf(fpr,"%d %.2f¥n",step+1,rate);

   if(comp==1)
     printf("");

   if(comp==1)
     break;

   step=step+1;
   if(step==10000)
     break;
  }
  printf("%d¥n",No);
  fclose(fp);
  fclose(fpr);
  fclose(move);
}
```

＝提案手法 2.1 ＝

```
#include<stdio.h>
#include<math.h>

#define X 20
#define Y 20
#define r 3
#define N 3
#define Rout 99

int map[Y][X];

struct robot{
  int x;
  int y;
}robot[N];

struct node{
  int x;
  int y;
}node[Rout];

int main()
{
  FILE *fp;
  int i,c,OBSTx,OBSTy,d;

   for(i=0;i<=Y-1;i++)
    for(c=0;c<=X-1;c++)
      {
      map[i][c]=0;
      }
   srand((unsigned)time(NULL));

   for(i=0;i<2;i++)
     for(c=0;c<2;c++){
       OBSTx = rand() % 20;
       OBSTy = rand() % 19;
       map[OBSTy][OBSTx]=2;
     }

   fp=fopen("prop2-1-map.txt","w");

   for(i=0;i<=Y-1;i++)
     for(c=0;c<=X-1;c++)
       {
                  if(map[i][c]==0)
                    fprintf(fp,"□");
                  else if(map[i][c]==2)
                    fprintf(fp,"■");
                  if(c==X-1)
                    fprintf(fp,"¥n");
       }

   fprintf(fp,"------------------------------------------------------------create map¥n");

   for(i=0;i<N;i++){
     robot[i].x=9+i;
     robot[i].y=19;
   }



   int marukaita=0;

   for(i=0;i<=Y-1;i++){
     for(c=0;c<=X-1;c++)
       {
                  for(d=0;d<N;d++){
                    if(map[i][c]==0 && i==robot[d].y && c==robot[d].x){
                      fprintf(fp,"★");
                      marukaita=1;
                    }
                  }
                  if(marukaita==1)
                    marukaita=0;
                  else if(map[i][c]==0)
                    fprintf(fp,"□");
                  else if(map[i][c]==2)
                    fprintf(fp,"■");
                  if(c==X-1)
                    fprintf(fp,"¥n");
       }
   }
}
fprintf(fp,"------------------------------------------------------------begin¥n");
int experience[Y][X];
int ifX[N],ifY[N],temX[N],temY[N];
int u=0;
int tempu=0;
int comp=0;
int exp=0;
int step=0;
int kaiten;
int k=700;
float rate;
FILE *fpr;
int perm[N];
int n;
int NorWes[]={4,5,7,8};
int Nor[]={3,4,5,6,7,8};
int NorEas[]={3,4,6,7};
int Wes[]={1,2,4,5,7,8};
int Eas[]={0,1,3,4,6,7};
int SouWes[]={1,2,4,5};
int Sou[]={0,1,2,3,4,5};
int SouEas[]={0,1,3,4};
int connec[N];
int inrad=0;
int toa,verifi[N],nodeverifi[Rout];
int No=0;
int qua;
int dis=0;
FILE *veri;

for(n=0;n<N;n++){
  verifi[n]=0;
}

for(n=0;n<Rout;n++)
  nodeverifi[n]=0;

for(i=0;i<=Y-1;i++)
  for(c=0;c<=X-1;c++)
    experience[i][c]=0;

for(n=0;n<N;n++)
  experience[robot[n].y][robot[n].x]=1;

fpr=fopen("stati.txt","w");
srand((unsigned)time(NULL));

FILE *move;
move=fopen("robot(1)loca.txt","w");
veri=fopen("verifi.txt","w");

while(1){

  for(kaiten=0;kaiten<k;kaiten++){
                  for(n=0;n<N;n++)
                  {
                    perm[n]=rand() % 9;
                  }

                  for(n=0;n<N;n++){
                    if(robot[n].x==0 && robot[n].y==0){//フィールドの左上にいる時
                      perm[n]=NorWes[rand()%4];}
                    else if(robot[n].x==19 && robot[n].y==0){//フィールドの右上
                      perm[n]=NorEas[rand()%4];}
                    else if(robot[n].x==0 && robot[n].y==19){//フィールドの左下の時
                      perm[n]=SouWes[rand()%4];}
                    else if(robot[n].x==19 && robot[n].y==19){//フィールドの右下の時
                      perm[n]=SouEas[rand()%4];}
                    else if(robot[n].y==0){//フィールドの上側
                      perm[n]=Nor[rand()%6];}
                    else if(robot[n].x==0){//フィールドの左側
                      perm[n]=Wes[rand()%6];}
                    else if(robot[n].x==19){//フィールドの右側
                      perm[n]=Eas[rand()%6];}
                    else if(robot[n].y==19){//フィールドの下側
                      perm[n]=Sou[rand()%6];}
                  }
```

```
            for(n=0;n<N;n++){
              ifX[n]=robot[n].x;
              ifY[n]=robot[n].y;

            switch(perm[n]){
            case 0: //北西
              ifX[n]=robot[n].x-1;
              ifY[n]=robot[n].y-1;
              break;
            case 1: //北
              ifX[n]=robot[n].x;
              ifY[n]=robot[n].y-1;
              break;
            case 2: //北東
              ifX[n]=robot[n].x+1;
              ifY[n]=robot[n].y-1;
              break;
            case 3: //西
              ifX[n]=robot[n].x-1;
              ifY[n]=robot[n].y;
              break;
            case 4: //キープ
              ifX[n]=robot[n].x;
              ifY[n]=robot[n].y;
              break;
            case 5: //東
              ifX[n]=robot[n].x+1;
              ifY[n]=robot[n].y;
              break;
            case 6: //南西
              ifX[n]=robot[n].x-1;
              ifY[n]=robot[n].y+1;
              break;
            case 7: //南
              ifX[n]=robot[n].x;
              ifY[n]=robot[n].y+1;
              break;
            case 8: //南東
              ifX[n]=robot[n].x+1;
              ifY[n]=robot[n].y+1;
              break;
            }
            }

for(n=0;n<N;n++){
            if(experience[ifY[n]][ifX[n]]==0)
              tempu=tempu+1;
            else if(experience[ifY[n]][ifX[n]]==1)
              tempu=tempu+0;

            if(map[ifY[n]][ifX[n]]==2){
              tempu=tempu - 100;
            }

            if(map[ifY[n]][ifX[n]]==3){
              tempu=tempu - 100;
            }
}

for(n=0;n<N;n++){
            for(c=0;c<N;c++){
              if(c==n)
                printf("");
              else if(ifX[n]==ifX[c]&&ifY[n]==ifY[c]){
              tempu=tempu-100;
              }
}
}

for(n=0;n<N;n++){
            verifi[n]=0;
}

for(n=0;n<N;n++){
            if(sqrt(pow(ifY[n]-19,2)+pow(ifX[n]-9,2))<=r)
              verifi[n]=1;
}

for(toa=0;toa<10;toa++){
            for(n=0;n<N;n++){
                        if(verifi[n]==0){
                          for(qua=0;qua<N;qua++){
                            if(qua==n){
                                        printf("");}
                            else if(verifi[qua]==0)
                                        printf("");
                            else{
                                        if(verifi[qua]==1 && sqrt(pow(ifY[n]-ifY[qua],2)+pow(ifX[n]-
ifX[qua],2))<=r)
                                          verifi[n]=1;
                            }
                          }
                        }
                      }
    }

    if(No>0){
                for(n=0;n<Rout;n++)
                 nodeverifi[n]=0;


                for(n=0;n<N;n++){
                  for(qua=0;qua<No;qua++){
                    if(sqrt(pow(ifY[n]-node[qua].y,2)+pow(ifX[n]-node[qua].x,2))<=r)
                                verifi[n]=1;
                  }
                }

    }

    for(toa=0;toa<10;toa++){
                for(n=0;n<N;n++){
                  if(verifi[n]==0){
                    for(qua=0;qua<N;qua++){
                      if(qua==n){
                                printf("");}
                      else if(verifi[qua]==0)
                                printf("");
                      else{
                                if(verifi[qua]==1 && sqrt(pow(ifY[n]-ifY[qua],2)+pow(ifX[n]-
ifX[qua],2))<=r)
                                  verifi[n]=1;
                      }
                    }
                  }
                }
    }

    toa=0;
    for(n=0;n<N;n++)
                toa=toa+verifi[n];

    if(toa<N){
                tempu=tempu-100;
    }

    if(tempu>u){
                u = tempu;
                for(n=0;n<N;n++){
                  temX[n] = ifX[n];
                  temY[n] = ifY[n];
                }

    }
    tempu=0;
  }

  for(n=0;n<N;n++){
    verifi[n]=0;
  }

  for(n=0;n<Rout;n++)
    nodeverifi[n]=0;

  toa=0;
  u=-1;
  tempu=0;
  for(n=0;n<N;n++){
    robot[n].x=temX[n];
    robot[n].y=temY[n];
    experience[robot[n].y][robot[n].x]=1;
```

```
    }

    dis=0;
    int ndis=1000000;
    int mindis;

    for(n=0;n<N;n++){
      if(sqrt(pow(robot[n].y-19,2)+pow(robot[n].x-9,2))>=(N-0.3)*r && No<Rout){
                  for(c=0;c<N;c++){
                    if(sqrt(pow(robot[c].y-19,2)+pow(robot[c].x-9,2))<ndis)
                      {
                        ndis=sqrt(pow(robot[c].y-19,2)+pow(robot[c].x-9,2));
                        mindis=c;
                      }
                  }
                  if(No>0){
                    for(i=0;i<No;i++){
                      if(sqrt(pow(node[i].y-robot[mindis].y,2)+pow(node[i].x-
robot[mindis].x,2))<r)
                        dis++;
                    }
                    if(dis==0 && map[robot[mindis].y][robot[mindis].x]!=3){
                      node[No].x=robot[mindis].x;
                      node[No].y=robot[mindis].y;
                      map[node[No].y][node[No].x]=3;
                      No++;
                    }
                  }
                  else{
                    node[No].x=robot[mindis].x;
                    node[No].y=robot[mindis].y;
                    map[node[No].y][node[No].x]=3;
                    No++;
                  }
      }
    }

    fprintf(move,"(%d,%d)¥n",robot[1].y,robot[1].x);
    for(i=0;i<=Y-1;i++){
      for(c=0;c<=X-1;c++)
                  {
                    for(d=0;d<N;d++){
                      if(map[i][c]==0 && i==robot[d].y && c==robot[d].x){
                        fprintf(fp,"★");
                        marukaita=1;
                      }
                    }
                    if(marukaita==1)
                      marukaita=0;
                    else if(map[i][c]==3)
                      fprintf(fp,"▼");
                    else if(experience[i][c]==1)
                      fprintf(fp,"侑");
                    else if(map[i][c]==0)
                      fprintf(fp,"□");
                    else if(map[i][c]==2)
                      fprintf(fp,"■");
                    if(c==X-1)
                      fprintf(fp,"¥n");
                  }
    }

    for(i=0;i<=Y-1;i++)
      for(c=0;c<=X-1;c++)
                  exp = exp + experience[i][c];

    rate=exp/396.0*100.0;

                      printf("%f¥n",rate);

    if(exp==396)
      comp=1;
    else
      exp=0;

    fprintf(fp,"----------------------------------------------------------step%d(%.2f)¥n",step+1,rate);

    fprintf(fpr,"%d %.2f¥n",step+1,rate);

    if(comp==1)
      printf("");
      if(comp==1)
        break;

      step=step+1;
      if(step==10000)
        break;
    }
    printf("%d¥n",No);
    fclose(fp);
    fclose(fpr);
    fclose(move);
}
```

＝提案手法 2.2＝

```
#include<stdio.h>
#include<math.h>

#define X 20
#define Y 20
#define r 3
#define N 3
#define Rout 99

int map[Y][X];

struct robot{
  int x;
  int y;
}robot[N];

struct node{
  int x;
  int y;
}node[Rout];

int main()
{
  FILE *fp;
  int i,c,OBSTx,OBSTy,d;

   for(i=0;i<=Y-1;i++)
    for(c=0;c<=X-1;c++)
      {
      map[i][c]=0;
      }
   srand((unsigned)time(NULL));

   for(i=0;i<2;i++)
     for(c=0;c<2;c++){
       OBSTx = rand() % 20;
       OBSTy = rand() % 19;
       map[OBSTy][OBSTx]=2;
     }

   fp=fopen("prop2-2-map.txt","w");

   for(i=0;i<=Y-1;i++)
     for(c=0;c<=X-1;c++)
       {
                    if(map[i][c]==0)
                      fprintf(fp,"□");
                    else if(map[i][c]==2)
                      fprintf(fp,"■");
                    if(c==X-1)
                      fprintf(fp,"¥n");
       }

   fprintf(fp,"------------------------------------------------------------create map¥n");

   for(i=0;i<N;i++){
     robot[i].x=9+i;
     robot[i].y=19;
   }

   for(i=0;i<N;i++){
     for(c=0;c<N;c++){
       robot[i].list[c]=1;
     }
   }

   int marukaita=0;

   for(i=0;i<=Y-1;i++){
     for(c=0;c<=X-1;c++)
       {
                    for(d=0;d<N;d++){
                      if(map[i][c]==0 && i==robot[d].y && c==robot[d].x){
                        fprintf(fp,"★");
                        marukaita=1;
                      }
                    }
                    if(marukaita==1)
                      marukaita=0;
                    else if(map[i][c]==0)
                      fprintf(fp,"□");
                    else if(map[i][c]==2)
                      fprintf(fp,"■");
                    if(c==X-1)
                      fprintf(fp,"¥n");
       }
   }
fprintf(fp,"------------------------------------------------------------begin¥n");
int experience[Y][X];
int ifX[N],ifY[N],temX[N],temY[N];
int u=0;
int tempu=0;
int comp=0;
int exp=0;
int step=0;
int kaiten;
int k=700;
float rate;
FILE *fpr;
int perm[N];
int n;
int NorWes[]={4,5,7,8};
int Nor[]={3,4,5,6,7,8};
int NorEas[]={3,4,6,7};
int Wes[]={1,2,4,5,7,8};
int Eas[]={0,1,3,4,6,7};
int SouWes[]={1,2,4,5};
int Sou[]={0,1,2,3,4,5};
int SouEas[]={0,1,3,4};
int connec[N];
int inrad=0;
int toa,verifi[N],nodeverifi[Rout];
int No=0;
int qua;
int dis=0;
FILE *veri;

for(n=0;n<N;n++){
  verifi[n]=0;
}

for(n=0;n<Rout;n++)
  nodeverifi[n]=0;

for(i=0;i<=Y-1;i++)
  for(c=0;c<=X-1;c++)
    experience[i][c]=0;

for(n=0;n<N;n++)
  experience[robot[n].y][robot[n].x]=1;

fpr=fopen("stati.txt","w");
srand((unsigned)time(NULL));

FILE *move;
move=fopen("robot(1)loca.txt","w");
veri=fopen("verifi.txt","w");

while(1){

  for(kaiten=0;kaiten<k;kaiten++){
                 for(n=0;n<N;n++)
                {
                  perm[n]=rand() % 9;
                }

                 for(n=0;n<N;n++){
                   if(robot[n].x==0 && robot[n].y==0){//フィールドの左上にいる時
                     perm[n]=NorWes[rand()%4];}
                   else if(robot[n].x==19 && robot[n].y==0){//フィールドの右上
                     perm[n]=NorEas[rand()%4];}
                   else if(robot[n].x==0 && robot[n].y==19){//フィールドの左下の時
                     perm[n]=SouWes[rand()%4];}
                   else if(robot[n].x==19 && robot[n].y==19){//フィールドの右下の時
                     perm[n]=SouEas[rand()%4];}
                   else if(robot[n].y==0){//フィールドの上側
                     perm[n]=Nor[rand()%6];}
                   else if(robot[n].x==0){//フィールドの左側
                     perm[n]=Wes[rand()%6];}
                   else if(robot[n].x==19){//フィールドの右側
```

```
            perm[n]=Eas[rand()%6];}
          else if(robot[n].y==19){//フィールドの下側
            perm[n]=Sou[rand()%6];}
        }

        for(n=0;n<N;n++){
          ifX[n]=robot[n].x;
          ifY[n]=robot[n].y;

        switch(perm[n]){
        case 0: //北西
          ifX[n]=robot[n].x-1;
          ifY[n]=robot[n].y-1;/
          break;
        case 1: //北
          ifX[n]=robot[n].x;
          ifY[n]=robot[n].y-1;
          break;
        case 2: //北東
          ifX[n]=robot[n].x+1;
          ifY[n]=robot[n].y-1;
          break;
        case 3: //西
          ifX[n]=robot[n].x-1;
          ifY[n]=robot[n].y;
          break;
        case 4: //キープ
          ifX[n]=robot[n].x;
          ifY[n]=robot[n].y;
          break;
        case 5: //東
          ifX[n]=robot[n].x+1;
          ifY[n]=robot[n].y;
          break;
        case 6: //南西
          ifX[n]=robot[n].x-1;
          ifY[n]=robot[n].y+1;
          break;
        case 7: //南
          ifX[n]=robot[n].x;
          ifY[n]=robot[n].y+1;
          break;
        case 8: //南東
          ifX[n]=robot[n].x+1;
          ifY[n]=robot[n].y+1;
          break;
        }
        }

for(n=0;n<N;n++){
        if(experience[ifY[n]][ifX[n]]==0)
          tempu=tempu+1;
        else if(experience[ifY[n]][ifX[n]]==1)
          tempu=tempu+0;

        if(map[ifY[n]][ifX[n]]==2){
          tempu=tempu - 100;
        }

        if(map[ifY[n]][ifX[n]]==3){
          tempu=tempu - 100;
        }
}

for(n=0;n<N;n++){
        for(c=0;c<N;c++){
          if(c==n)
            printf("");
          else if(ifX[n]==ifX[c]&&ifY[n]==ifY[c]){
          tempu=tempu-100;
          }
}
}

for(n=0;n<N;n++){
        verifi[n]=0;
}

for(n=0;n<N;n++){
        if(sqrt(pow(ifY[n]-19,2)+pow(ifX[n]-9,2))<=r)
          verifi[n]=1;
    }

    for(toa=0;toa<10;toa++){
            for(n=0;n<N;n++){
              if(verifi[n]==0){
                for(qua=0;qua<N;qua++){
                  if(qua==n){
                        printf("");}
                  else if(verifi[qua]==0)
                        printf("");
                  else{
                        if(verifi[qua]==1 && sqrt(pow(ifY[n]-ifY[qua],2)+pow(ifX[n]-
ifX[qua],2))<=r)
                          verifi[n]=1;
                  }
                }
              }
            }
    }

    if(No>0){
             for(n=0;n<Rout;n++)
              nodeverifi[n]=0;

             for(n=0;n<N;n++){
              for(qua=0;qua<No;qua++){
                if(sqrt(pow(ifY[n]-node[qua].y,2)+pow(ifX[n]-node[qua].x,2))<=r)
                      verifi[n]=1;
              }
             }
    }

    for(toa=0;toa<10;toa++){
            for(n=0;n<N;n++){
              if(verifi[n]==0){
                for(qua=0;qua<N;qua++){
                  if(qua==n){
                        printf("");}
                  else if(verifi[qua]==0)
                        printf("");
                  else{
                        if(verifi[qua]==1 && sqrt(pow(ifY[n]-ifY[qua],2)+pow(ifX[n]-
ifX[qua],2))<=r)
                          verifi[n]=1;
                  }
                }
              }
            }
    }

    toa=0;
    for(n=0;n<N;n++)
            toa=toa+verifi[n];

    if(toa<N){
            tempu=tempu-100;
    }

    if(tempu>u){
            u = tempu;
            for(n=0;n<N;n++){
              temX[n] = ifX[n];
              temY[n] = ifY[n];
            }

    }
    tempu=0;
   }

   for(n=0;n<N;n++){
    verifi[n]=0;
   }

   for(n=0;n<Rout;n++)
    nodeverifi[n]=0;

   toa=0;
   u=-1;
   tempu=0;
   for(n=0;n<N;n++){
    robot[n].x=temX[n];
```

```
    robot[n].y=temY[n];
    experience[robot[n].y][robot[n].x]=1;
   }

   dis=0;//中継ノードを置くかどうかの判定をするための変数
   int ndis=1000000;//基地局とロボットの距離。これが最も小さいロボットが中継ノードを置く
   int mindis;//基地局と一番近いロボットの番号
   int kyori=0;
   int flag=0;//最近ロボットが基地局・中継ノードとの距離がギリギリにいるなら1にする

   for(n=1;n<N;n++){
    kyori=kyori+sqrt(pow(robot[0].y-robot[n].y,2)+pow(robot[0].x-robot[n].x,2));
   }

   if(kyori>=(N*N-N)/2*r-0.3 && No<Rout){
    for(c=0;c<N;c++){
                if(sqrt(pow(robot[c].y-19,2)+pow(robot[c].x-9,2))<ndis)
                 {
                  ndis=sqrt(pow(robot[c].y-19,2)+pow(robot[c].x-9,2));
                  mindis=c;
                 }
    }

    for(c=0;c<No;c++){
                if(r>=sqrt(pow(robot[mindis].y-node[c].y,2)+pow(robot[mindis].x-
node[c].x,2)) && sqrt(pow(robot[mindis].y-node[c].y,2)+pow(robot[mindis].x-
node[c].x,2))>(r-0.7)||r>=sqrt(pow(robot[mindis].y-19,2)+pow(robot[mindis].x-9,2))
&& sqrt(pow(robot[mindis].y-19,2)+pow(robot[mindis].x-9,2))>(r-0.7))
                 flag=1;
    }

    if(No>0){
                for(i=0;i<No;i++){
                 if(sqrt(pow(node[i].y-robot[mindis].y,2)+pow(node[i].x-
robot[mindis].x,2))<r)
                   dis++;
                }

                if(flag==1 && dis==0 && map[robot[mindis].y][robot[mindis].x]!=3){
                 node[No].x=robot[mindis].x;
                 node[No].y=robot[mindis].y;
                 map[node[No].y][node[No].x]=3;
                 No++;
                }
    }
    else{
                node[No].x=robot[mindis].x;
                node[No].y=robot[mindis].y;
                map[node[No].y][node[No].x]=3;
                No++;
    }
   }

  fprintf(move,"(%d,%d)¥n",robot[1].y,robot[1].x);
  for(i=0;i<=Y-1;i++){
   for(c=0;c<=X-1;c++)
             {
              for(d=0;d<N;d++){
               if(map[i][c]==0 && i==robot[d].y && c==robot[d].x){
                fprintf(fp,"★");
                marukaita=1;
               }
              }
              if(marukaita==1)
               marukaita=0;
              else if(map[i][c]==3)
               fprintf(fp,"▼");
              else if(experience[i][c]==1)
               fprintf(fp,"侑");
              else if(map[i][c]==0)
               fprintf(fp,"□");
              else if(map[i][c]==2)
               fprintf(fp,"■");
              if(c==X-1)
               fprintf(fp,"¥n");
             }
  }

  for(i=0;i<=Y-1;i++)
   for(c=0;c<=X-1;c++)
             exp = exp + experience[i][c];

  rate=exp/396.0*100.0;

                printf("%f¥n",rate);

  if(exp==396)
   comp=1;
  else
   exp=0;

  fprintf(fp,"----------------------------------------------------------step%d(%.2f)¥n",step+1,rate);

  fprintf(fpr,"%d %.2f¥n",step+1,rate);

  if(comp==1)
   printf("");

  if(comp==1)
   break;

  step=step+1;
  if(step==10000)
   break;
 }
 printf("%d¥n",No);
 fclose(fp);
 fclose(fpr);
 fclose(move);
}
```