2M1-3

# テーマパークで迷子を探す -大規模エージェントシステムとしての動的制約条件下での移動目標探索-

# Finding A Lost Child in A Crowded Amusement Park -Proposal of a Benchmark Problem for Massive Agent Systems-

村田 悠也 *1 Yuya MURATA　山本 学 *1 Gaku YAMAMOTO　寺野 隆雄 *1 Takao TERANO

*1東京工業大学大学院総合理工学研究科知能システム科学専攻
Depertment of Computational Intelligence and Systems Science Interdiscplinary, Graduate School of Science and Engineering ,Tokyo Institute of Technology

Although there are growing needs on massive agent systems such as traffic management and evacuate decision making, we have not yet had good benchmark problems to evaluate such massive agent applications. In this paper, we propose a benchmark problem and discuss the benefits and limitations. For the purpose, we extend Moving Target Search (MTS) problem. The original MTS is a real-time algorithm to let a problem solver react a moving goal. MTS assumes the constraints or obstacles of the search space are static. We have relaxed the assumption so that the constraints or obstacles are also dynamically changing. This problem seems too simple for a benchmark, however, from our experiments, a naive implementation without careful time and space management usually causes contradictive agent behaviors. We conclude that the problem we have proposed is effective to evaluate the performance and nature of massive agent systems.

## 1. はじめに

現実で発生する都市交通問題や災害時の避難シミュレーションを行うにあたって車や避難者 (エージェント) の数，シミュレーション時の環境サイズは，現実への接地の観点から大きいことが望ましい．例えば，都市レベルの交通問題を扱う場合，100 台，200 台程度の数では，問題を十分に表現しているとは言えない．大規模エージェントベースシミュレーション (Agents-based Simulation, ABS) では，1 万〜1000 万のエージェントの行動をシミュレーションすることで，この問題を解決する．例えば，交通流シミュレータである XAXIS(X10-based Agents eXecutive Infrastructure for Simulation) は並列処理言語を使用した分散基盤となっており，スーパーコンピュータ上の処理ノードを増やすことで，都市レベルの問題を解くことができる [Suzumura 12]．また，ZASE(Zonal Agents-based Simulation Environment) は，コンピュータを複数台接続した分散環境上で大規模 ABS を実行するシステムとなっている [Yamamoto 08]．

このように，大規模 ABS は分散環境上での実行が主である．大規模 ABS では，エージェント数を増やした際の変化をシミュレーションする．しかしながら，ABS ではエージェント移動とインタラクションの 2 つが毎ステップ実行されるため，計算時間がエージェント数に対して指数的に増加する．大規模 ABS 基盤では，エージェント数増加による計算コストの問題を解決するため，エージェント数に対してスケールに性能を向上させる仕組みが必要となる．そのため，大規模 ABS ではサーバーの接続により 1 サーバ上のエージェント数を減らし，システム全体の性能を向上させる分散技術が適用される．

一方，大規模 ABS 基盤の評価は，エージェント数に対する実行時間について評価される．そのため，従来の ABS は大規模化可能か，分散可能かということについては十分に考慮されず，単純にサイズを大きくした ABS がベンチマーク問題として適用されてきた．図 1 は，分散環境上での実行を目的とし設計した ABS である．このシミュレーションは，Goal を追跡する Agent とランダムに移動する Goal，全てのマスに敷き詰められた障害物からなる単純な問題をシミュレーションしている．当然ながら，隣接するマス全てに障害物が存在するため，Agent が Goal にたどり着くことはない．しかしながら，このシミュレーションを実行した場合，Agent が Goal にたどり着いてしまう．

本研究では，分散環境上で実行される大規模 ABS のベンチマーク問題として障害物が動的に変化する拡張 MovingTarget-Search(MTS) を提案する．同時にスケーラブルに設計した場合の ABS で発生がした図 1 の実行が完了する問題について考察を行う．

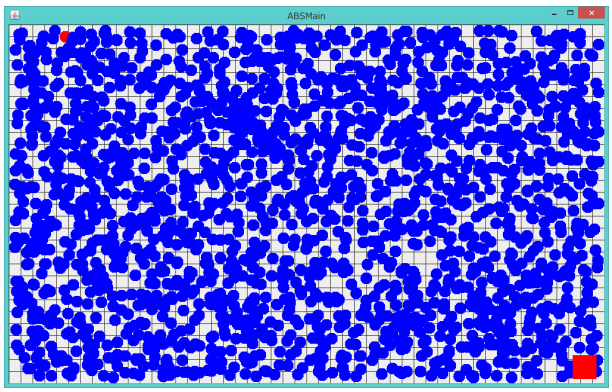


図 1: 拡張 MovingTargetSearch のシミュレーション

## 2. Moving Target Search

MTS は，目標を追跡する Solver とランダムに移動する Target からなる問題である [Ishida 91]．この問題の特徴は，目標

連絡先: 村田悠也，東京工業大学大学院総合理工学研究科知能システム科学専攻，神奈川県横浜市緑区長津田町 4529，045-924-5815，murata@trn.dis.titech.ac.jp

が探索実行中に変化するため，計算コストの高いオフライン探索が適用できない点にある．そのため，一定時間内での最適経路の推定と探索を交互に繰り返すオンライン探索が必要となる．

### 2.1 MTS アルゴリズム

MTS アルゴリズムは，Solver の移動と Target の移動の 2 つのイベントからなり，移動ごとに推定距離の計算を行うことで最短経路を探索する．Solver の最短経路探索は，Solver の現在位置 $x$ から，その近隣の $x'$ について，それぞれ推定距離を計算し，Target までの最も短い推定距離を更新する．Target の移動時には，Target が動くたびに Target の新しい位置 $y$ について，近隣の $y'$ の推定距離の計算を行い更新する．Solver と Target の移動プログラムを擬似プログラムにより実現した場合，以下のようになる．

#### 2.1.1 Solver の移動

Solver は，Solver が移動できる範囲 (隣接するマス) に対して Target までの推定距離を計算し，最短となる位置に移動する．なお，$x'$ は x の隣接マスである．

**Algorithm 1** Solver Code

$x \leftarrow InitialPositionx_0$
**while** $x \neq y$ **do**
  Computation $h(x', y)$
  $h(x, y) \leftarrow max(h(x, y), min_{x'}\{h(x', y) + 1\})$
  $x \leftarrow x'$
**end while**

#### 2.1.2 Target の移動

Target は，移動できる範囲にランダムに移動し，その隣接したマスから Solver までの最短の推定距離を更新する．なお，$y'$ は，$y$ の隣接マスで，Random がとる値の範囲は $Solver$ の移動量以下とする．

**Algorithm 2** Target Code

$y \leftarrow InitialPositiony_0$
**while** $x \neq y$ **do**
  $y' \leftarrow y + Random$
  Computation $h(x, y')$
  $h(x, y) \leftarrow max(h(x, y), \{h(x, y') - 1\})$
**end while**

## 3. 拡張 MTS

拡張 MTS は，障害物が動的に変化する環境で行う MTS である．そのため，次の 2 つの特徴を持つ．

- Agent(Solver) や Goal(Target) が移動した時点で計算した推定距離は，誤った推定となってしまう [*1]．もし，正しい推定距離を計算する場合，障害物の移動イベントに対して，推定距離を再計算する必要がある．
- 障害物の数によって，MTS の完全性 (解となる経路が存在すれば，必ず目標に到達する) が失われる可能性がある．ここでは，完全性が失われた状態を解決不可能な状態と呼ぶ．

*1 拡張 MTS は ABS によってシミュレーションを行うため，Solver は Agent，Target は Goal とした．

障害物の移動によって最短経路は更新し続けなければならず，また，上記の特徴により計算出来ない可能性もある．そのため，障害物の時間的変化を考慮した推定や探索が必要となる．しかしながら拡張 MTS は，障害物，Agent，Goal が相互に影響し合い，内部の行動ルールとそこに含まれる確率的な進路の決定から複雑なシステムとなっている．そのため，障害物が多い環境では，全ての障害物の行動に対して計算する必要があり，計算コストの問題からオンライン探索であっても適用は難しい．よって，ABS によるアプローチが必要とされる．

#### 3.0.3 Obtacle の移動

Obtacle のプログラム．

**Algorithm 3** Obtacle Code

$o \leftarrow InitialPositiono_0$
**while** $x \neq y$ **do**
  $x \leftarrow CurrentPositionx$
  **for** $n = 0$ to $NumberofObstacles$ **do**
    $o' \leftarrow o + Random$
    Computation $h(x', y)$
    $h(x, y) \leftarrow min_{x'}\{h(x', y) + 1\}$
  **end for**
**end while**

### 3.1 解決不可能な状態

解決不可能な状態は，Agent または Goal の隣接するマス数以上の障害物が存在するときに発生する．ここでは，その特徴から 3 つに分類される．

#### 3.1.1 分断状態

分断状態では，Agent または Goal の間に障害物が存在し，到達経路が無くなる状態である．到達経路は存在しないものの Agent や Goal は動くことが可能でかつ障害物をはさんで接近可能である．時間経過により解消する (図 2)．

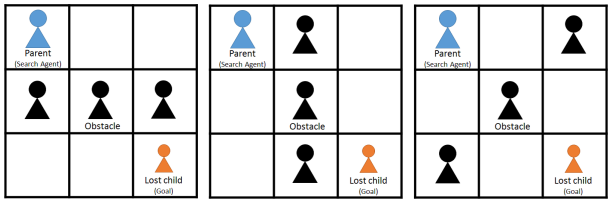


図 2: 分断状態

#### 3.1.2 八方塞がり状態

八方塞がり状態では，Agent または Goal の両方もしくは片方の隣接マス全てに障害物が存在する状態である．そのため，八方塞がり状態になった Agent または Goal は移動することができない．片方がのみが八方塞がり状態ならば，障害物をはさんで接近可能である．また，分断状態と同様に時間経過により解消される (図 3)．

#### 3.1.3 完全状態

完全状態では，Agent と Goal を除く全てのマスに障害物が存在する状態である．この状態は，障害物数がマスサイズ - 2 のときに発生し，目標には到達できない (図 4)．

### 3.2 迷子探し問題

拡張 MTS を ABS として実装したものが迷子探し問題である．この問題は，親 (Agent) が迷子になった子供 (Goal) を客

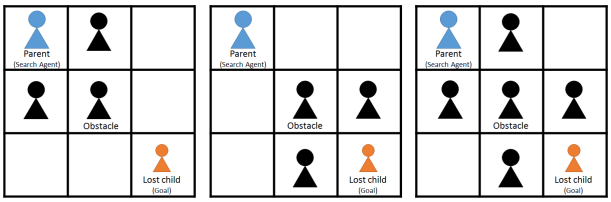


図 3: 八方塞がり状態

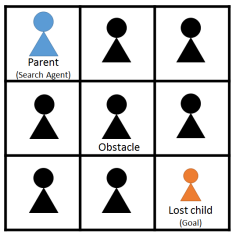


図 4: 完全状態

(Obtacle) で混雑した環境で探す問題である．親が，常に GPS により子供位置を取得できるとき，迷子になって移動する子供を見つける問題は MTS 問題と同様の問題となる．また，混雑した客を動く障害物と見た時，客は探索における動的制約とみなすことができ，その環境上での迷子探し問題は，拡張 MTS 問題へ変換可能となる．

テーマパークと限定した時，動かない障害物や混雑時の人の流れは，テーマパーク内でのアトラクションやイベントに影響を受け変化すると考えられる．しかし，今回作成したシミュレーションは，問題をより単純化し，ランダムに動く障害物 (Obtacle) と親 (Agent)，迷子 (Goal) の 3 つの要素に絞りシミュレーションを行う．

単純化した迷子探し問題のシステムの要件は以下のようになる．

- 環境には，親 (Agent)，子 (Goal)，客 (Obtacle) が存在する．
- 親は，子の位置情報を取得し，その方向に移動する．
- 客は衝突するまで進み，衝突すると逆方向に移動する．
- 子は，ランダムに移動する．
- システムは，親が子供を見つけるまで実行される．

## 4. シミュレーション実験

単純化した迷子問題をシミュレーションを実行したものが図 1，図 5，図 6，である．図 5 は，障害物数を 50 とした時のシミュレーションの様子である．この環境では，エージェントはほとんど障害物と当たること無く Goal にたどり着くことが可能である．図 6 は，エージェント数を 500 とした時のシミュレーションの様子である．この環境では，エージェントは障害物と衝突する．また，Goal は周りの障害物が原因でほとんど動くことはない．Introduction で挙げた図 1 は，エージェント数を 2000 にした時のシミュレーションの様子である．この環境は，解決不可能な状態の完全状態となっている．そのため，Agent が Goal にたどり着くことはない．しかしながら，図**??** のシミュレーションの結果からわかるように，解決不可能な状態であるにも関わらず Goal にたどり着いている．

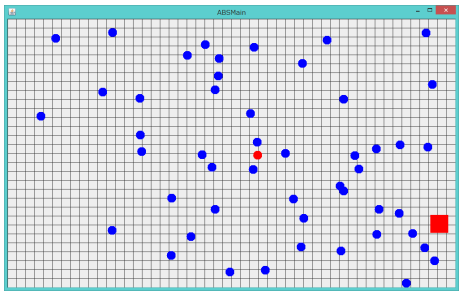

図 5: エージェント数=50

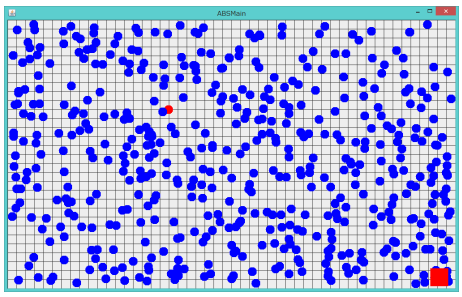

図 6: エージェント数=500

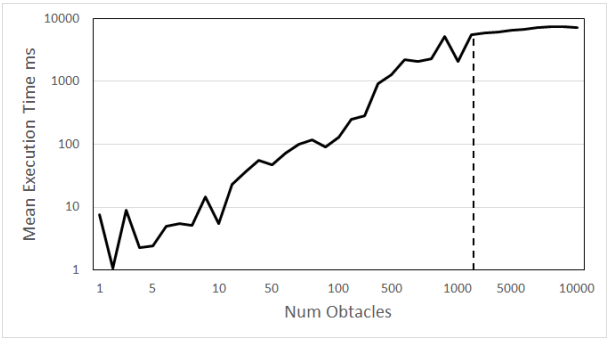


図 7: 拡張 MTS のシミュレーション結果． 縦軸は，対数表示となっている．また，グラフ中の点線はシミュレーションで用いた探査空間の解決不可能な状態 (完全状態) となる障害物数である．

## 5. 考察

今回のシミュレーションでは，分散システムへの実装を考慮し，Agent，Obtacle，Goal を非同期に設計した．このように設計することで，単に Agent や Obtacle，Goal を分散環境上に配置することで分散可能となる．そのため，スケーラビリティが高い設計であるといえる．しかしながら，実際にシミュレーションを動かすと，解決不可能な状態のうち目標へ辿りつけないはずの完全状態であっても実行が完了するプログラムとなっていた．これは，ABS の設計において必要とされる同期すべき要素を考慮せずに分散設計を行ったため誤ったシミュレーション結果となった．ここで，同期が必要な要素を考える上で，ABS の実行を考える．ABS は実行は 1 ステップに Agent が 1 回行動するステップ実行となっている．そのため，Agent の移動やインタラクション行う際にその行動に順序制約が存在する．例えば，Agent1，Agent2 をステップ実行した場合，実行順序は Agent1 → Agent2 もしくは Agent2 → Agent1 の順序で実行される．ABS では，全ての Agent は同

時に活動するという特徴を持つが，システム上で実現される場合，同時に実行されることはない．もし，同時に実行された場合，処理競合が発生する．そのため，ABS では，見かけ上の同時実行が為されてきた．例えば，同時にインタラクションが発生した場合は確率的に判定を行ったり，Agent にの行動順序をランダムに決定するといった方法により実現される．しかしながら，分散環境では並列的に Agent を動かすことで，同時に活動する可能性がある．

今回の迷子問題では，ABS はステップ実行ではく，個々の Agent が内部に持つ行動ルールにより並列的に実行している．そのため，ステップ実行による時間同期を考慮していなかった．これにより発生する問題は，処理競合による誤ったシミュレーションである．しかしながら，この問題は ABS の統計的結果から判断することは難しい．また，大規模 ABS では Agent 数が非常に多いことからシミュレーション動作中の目視による確認は難しい．一方，拡張 MTS は解決不可能な状態が存在することにより，分散可能な ABS を設計した場合の同期問題を検出する能力を持っていることがわかる．

### 5.1 拡張 MTS の時間同期問題

拡張 MTS において完全状態でも実行が完了してしまう問題は，次の Agent 移動時のプログラムステップにある．

1. Agent は現在位置 (0, 0) を環境から削除する．
2. Agent のルールに従って移動 (0, 1) する．
3. Agent は現在位置 (0, 1) を環境へ書き込む．

このように各エージェントの移動は，位置の削除，移動，登録の 3 ステップからなる．これが時間同期が考慮されていない，ABS を実行した場合，次のような実行ステップとなる可能性がある．

1. Agent ’ B ’ は現在位置 (0, 1) を環境から削除する．
2. Agent ’ A ’ は現在位置 (0, 0) を環境から削除する．
3. Agent ’ B ’ のルールに従って移動 (0, 0) する．
4. Agent ’ A ’ のルールに従って移動 (0, 1) する．
5. Agent ’ A ’ は現在位置を環境へ書き込む．
6. Agent ’ B ’ は現在位置を環境へ書き込む．

Agent ’ A ’ と Agent ’ B ’ は，互いの位置に移動を行うため衝突するはずが，この場合，Agent ’ B ’ の割り込みタイミングにより互いにすり抜け目的の位置に到達する．

### 5.2 拡張 MTS の精度とスケーラビリティの測定

拡張 MTS の精度，スケーラビリティの測定では，netlogo などのエージェント型プログラミング言語 [Wilensky 14] で設計した拡張 MTS と比較を行うことで，精度やスケーラビリティを測定可能である．もし，同一エージェント数で目標に辿り着くまでのステップ数の比較を行い，シミュレーション結果が netlogo で作成したものよりも悪い場合，分散 ABS の精度が低いことがわかる．また，プログラムの実行時間を分散数を変えて測定することでスケーラビリティの評価が可能である．

## 6. おわりに

本稿で提案した拡張 MTS は，エージェントベースアプローチにより設計される問題である．また，MTS という簡易な問題を拡張したため，エージェント数の増加や環境サイズを大きくするといった大規模化が非常に容易となっている．また，解決不可能な状態を持つことから，分散により正当性が失われた場合の検知器としての役割を持つ．精度やスケーラビリティの測定についてもエージェント型プログラミング言語により設計したものと比較を行うことで測定が可能である．拡張 MTS を大規模 ABS に適用した場合の特徴をまとめると以下のようになる．

- 完全状態により，分散時に時間同期が崩れた場合の検出器となる．
- netlogo で設計したプログラムと実行ステップを比較することで精度を測定可能．
- a)，b) を満たしたプログラムを各分散数での実行時間を比較することでスケーラビリティを測定可能．

このことから，拡張 MTS は分散基盤上で実行される大規模 ABS のベンチマークに適しているといえる．

今後の課題として，netlogo により設計した拡張 MTS との比較を行い精度をどのように比較するか，また正当性を保った状態で分散する場合，同期すべき要素と非同期で設計してもよい要素について調べる．

## 参考文献

[Ishida 91] Ishida, T. and Korf, R. E.: Moving Target Search., in *IJCAI*, Vol. 91, pp. 204–210 (1991)

[Suzumura 12] Suzumura, T. and Kanezashi, H.: Highly Scalable X10-based Agent Simulation Platform and its Application to Large-scale Traffic Simulation, in *Proceedings of the 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications*, pp. 243–250IEEE Computer Society (2012)

[Wilensky 14] Wilensky, U.: NetLogo (2014), `http://ccl.northwestern.edu/netlogo/`

[Yamamoto 08] Yamamoto, G., Tai, H., and Mizuta, H.: A platform for massive agent-based simulation and its evaluation, in *Massively Multi-Agent Technology*, pp. 1–12, Springer (2008)