芸術科学会論文誌 Vol. 22, No. 3, pp. 6:1 – 6:13 (2023)

バトルロイヤルゲームのアイテム探索に適したマルチエージェント経路探索の定式化

柴山叶 [1)](学生会員)　阿部雅樹 [2)](正会員)　渡辺大地 [2)](正会員)

1) 東京工科大学大学院 バイオ・情報メディア研究科　2) 東京工科大学 メディア学部

Multi-agent route search formulation suitable for item search in battle royal games.

Kanau Shibayama[1)]　**Masaki Abe**[2)]　**Taichi Watanabe**[2)]

1) Graduate School of Bionics, Computer and Media Science, Tokyo University of Technology
2) School of Media Science, Tokyo University of Technology

{g312100844, abemsk} @ edu.teu.ac.jp, earth@gamescience.jp

概要

一般的にマルチエージェントでの効率的な移動経路取得に応用される配送計画問題は，全てのノードを必ず巡回するため，エージェントの数が少なくノードが多い場合は 1 エージェントあたりのコスト (移動距離や時間など) が非常に大きくなる．そのため求めた経路が配送計画問題においては最適解でも実用性を伴わないケースが見られる．例えば時間制限のあるバトルロイヤルゲームにおいて，全ての建物を巡回すると時間が足りないため，配送計画問題の既存手法をバトルロイヤルゲームの AI に実装しても最適な移動経路を求めることは出来ない．そこで本研究では上記のような状況下でも実用的かつ効率的な経路を求めるべく，コスト制約下でノードを取捨選択出来る移動経路取得の定式について提案する．そのうえで一般的な配送計画問題である VRP，コスト制約を設けることのできる DCVRP の二つの問題の既存解法による探索結果と効率を比較した．その結果，コスト制約下で全てのノードを訪れることが出来ない状況下では，提案手法が他手法より効率の良い経路を求められることが示された．

Abstract

In general, the delivery planning problem used to obtain efficient routes for multi-agents requires all nodes to be visited, so the cost per agent (travel distance, time, etc.) becomes very high when the number of agents is small and the number of nodes is large. Therefore, there are cases where the route obtained is not practical, even if it is the optimal solution in the delivery planning problem. For example, in a time-limited battle royale game, the optimal route cannot be obtained by implementing an existing method for the delivery planning problem in the AI of a battle royale game because there is not enough time to visit all the buildings. Therefore, we propose a formula for obtaining a travel route that can select nodes under cost constraints in order to obtain a practical and efficient route even under the above circumstances. We then compare the efficiency of the existing solution methods with that of VRP, a general delivery planning problem, and DCVRP, which allows cost constraints to be set. The results show that the proposed method is more efficient than the others when not all nodes can be visited under cost constraints.

1　はじめに

1.1　研究背景

バトルロイヤルゲームでは，まずプレイヤーは 3，4 人のチームを組んで何も持たずに空から降りていく．そして地上に着いたら市街地の中を探索し，武器や道具を集めて自身を強化して他のチームのプレイヤーを倒し，最後 1 チームの生き残りを懸ける．その間，時間経過で安全地帯が狭まっていく．以上がバトルロイヤルゲームのルールであるが，近年大流行しており，代表的なタイトルでは PUBG:BATTLEGROUNDS(以下 PUBG)[1] や荒野行動 [2] が挙げられる．バトルロイヤルゲームは 1 試合に 100 人程の大人数を要するものが一般的であるが，プレイヤーのみだとマッチングに時間がかかるためゲーム AI[3] を採用しているタイトルも多い．また AI と戦うことで練習をしたいプレイヤーも多く，バトルロイヤルゲームの AI には相応の需要が見られる．しかし，新しいゲームジャンルかつルールが複雑ということもありゲーム AI は決して充実していない．特に AI の移動に関しては，アイテム探索や戦闘，長期の戦略的に有利な位置取りなど取るべき行動が多岐にわたるため，一つの定式による経路取得では成り立たない．そこで本研究ではアイテム探索の部分に着目し，アイテムをより多く発見できる経路探索の定式化を目指す．なお一連のゲームの流れと研究の対象とする部分を図 1 に表す．

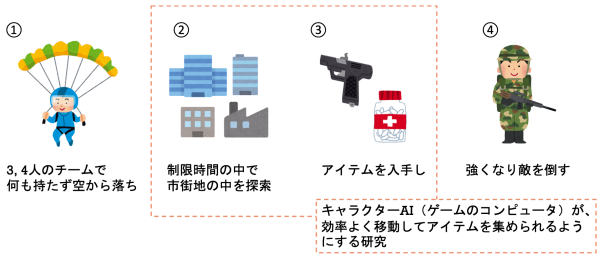

図 1　バトルロイヤルゲームとは．

1.2　先行研究

経路探索はグラフ理論 [4] を元に行われ，日常的な例であるとカーナビゲーション [5] や電車の乗り換え案内 [6] に応用されるが，本研究の対象であるゲームキャラクターの移動にも応用されている．Zeyad ら [7] はゲームにおける経路探索は深く研究，活用されていることを示した．具体的な一例としては，Zikky[8] が示したプレイヤーを最短距離で追いかけるゲームの敵キャラクターなどである．

これら経路探索においては地図上の地点群と地点間を結ぶ道を，それぞれグラフ理論のノードとエッジとして表現する．その中でもバトルロイヤルゲームのアイテム探索に応用できる理論の礎としては，まず巡回セールスマン問題 [9](Traveling Salesman Problem, TSP) が挙げられる．これは，セールスマンがある都市から出発し，全ての都市を訪問して出発地点に帰還する場合，どの順番で都市を回るのが最短かを求める最短経路問題 [10] である．図 2 は TSP の経路取得の例である．

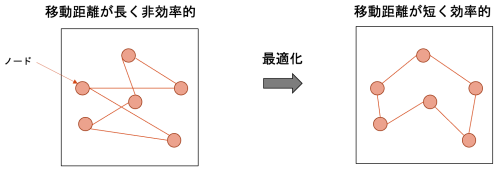

図 2　TSP とは．

この TSP を複数人で分担して行うものが配送計画問題 [11] (Vehicle Routing Problem, VRP) である．具体的には複数人で分担して街の全地点を 1 度ずつ訪れる経路の中で，移動距離が最小のものを求める問題で，名前の通り複数台のトラックで街の全顧客に効率的に荷物を配送出来る経路を求めるシステム [12] などに応用される．図 3 に VRP の模式図を示す．バトルロイヤルゲームのアイテム探索においても，複数人でアイテムがある地点を効率的に巡回する必要があるため，この VRP が応用できると考えた．

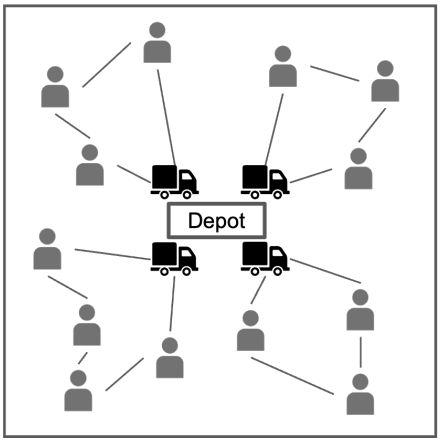

図 3　VRP の例．

また，VRP を拡張した問題として距離制約付き配送計画問題 [13](Distance-Constrained Vehicle Routing Problems, DCVRP) が挙げられる．DCVRP は VRP にコスト制約を設けたものである．

代表的な複数人での経路探索問題は上記 2 つであるが，これらは全地点を巡回することが前提であるため，エージェントの数が少なくノードが多い場合は 1 エージェントあたりのコスト（移動距離や時間など）が非常に大きくなる．バトルロイヤルゲームも一般的に 1 チーム 4 人程度に対し建物の数は非常に多く，加えて時間制限もあるため既存解法での最適解は実用的であるとは言い難い．

次に，コスト制約下で経由ノードの利益の最大化をする本手法に類似した配送計画問題として Team Orienteering Problem[14][15](TOP) が挙げられる．こちらの問題は最大化する対象は経由ノードの利益ではあるが，コスト制約は移動距離であり経由したノードで行う作業のコストは考慮していない．バトルロイヤルゲームのアイテム探索のコスト及び効率には，移動時間に加え建物の探索に要する時間も大きく影響するため TOP の解法で最適な経路は求めることが出来ないが，本研究の問題も TOP を拡張したものであると言える．

また，VRP に一つのエージェントが運べる荷物数の制約を設けた問題として容量制約付き配送計画問題 [16]，TOP に同様の制約を設けた問題として Capacitated Team Orienteering Problem[17][18] が挙げられる．バトルロイヤルゲームでも 1 エージェントが持てるアイテムの数は決まっている．しかしバトルロイヤルゲームのアイテム探索では多くのアイテムを運ぶのではなく，必要なものだけを手に入れるために多くのアイテムを見つけることが重要なため，上記の先行研究に設けられた制約は本研究において不要であると言える．

加えて，バトルロイヤルゲームの特徴である時間の制約に着目すると時間枠付き配送計画問題 [19][20](Vehicle Routing Problem with Time Windows, VRPTW) が先行研究として挙げられる．こちらは各顧客に対して荷物を配送できる時間帯を制限した VRP である．バトルロイヤルゲームのアイテム探索にも制限時間があるが，相違点は時間内に安全地帯へ辿り着くことが出来れば他の地点にいつ訪れてもよい点である．

上記の先行研究は全てノードの巡回に焦点を当てたものであるが，エッジの巡回に焦点を当てた経路探索問題もある．代表的なものに全てのエッジを巡回する最短経路を求める中国人郵便配達問題 [21] や，VRPTW のようにエッジに対して巡回すべき時間帯の制約をつけた Time-Constrained Chinese Postman Problem[22] がある．Fortnite[23] のように通路からもアイテムを得られるバトルロイヤルゲームもあるが極めて稀で，基本的にアイテムは通路に落ちていないため，エッジに焦点を当てたアプローチは汎用的で無いと考えた．

1.3 研究目標

本研究ではバトルロイヤルゲームのアイテム探索を TOP の拡張問題として扱う．各建物の位置関係や探索に要する時間，発見できるアイテムの数からコスト制約下で巡回する建物を取捨選択し，探索時間に対し発見できるアイテム数を増大させる経路探索の定式について提案する．

我々は，本研究の初期成果を NICOGRAPH2022 にて発表 [24] し，経路算出アルゴリズム記述の不足に関する指摘を受けた．本論文では，設定した制約条件に基づき経路を算出するアルゴリズムを詳細に記すと共に，評価記述についてより体系化した記述に改めた．

2 提案手法

2.1 前提

本研究においてはグラフ理論を使用し，アイテムを得られる建物をノード，エージェントの移動経路をエッジとする．バトルロイヤルゲームの市街地とエージェントの移動経路を真上から見たイメージが図 4 であり，この建物の位置とエージェントの移動経路をグラフとして表す．

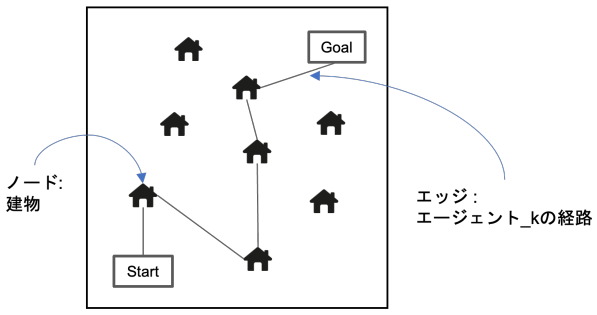

図 4　本研究におけるグラフの表し方．

また本研究で使用するのは重み付き無向グラフである．一般的な配送計画問題は移動距離をコストとする [25] が，移動距離と移動時間は比例する点と建物を探索する時間もコストとして考慮しなければいけない点から，エージェントが移動にかかる”時間”を移動コストとする．また各ノードには巡回する利益として発見できるアイテムの数，建物の探索に要する時間として探索コストを設定する．今回は建物が点在する屋外の地形を想定し，いずれのノード間でも移動出来ることとする．

その上でバトルロイヤルゲームではアイテム探索以外で孤立することは危険であり，チームでまとまって移動することが望ましいため，エージェントには同じ始点から出発し同じ終点に集合する制約を設ける．

2.2 定式化

ノードの数 を p とする．前述した通りノードは建物であるため，市街地にある建物の数であると言える．次にエージェントの数を a とおく．これはバトルロイヤルゲームの 1 チームの人数であり，a 人全体で最もアイテムを発見できる経路を探すこととなる．そしてノード k で発見できるアイテムの数を V_k，ノード k の探索に要する時間 (s) を T_k，ノード k をエージェント m が訪れたかを $x_{k,m}$，エージェント m が始点から終点までの移動にかかった時間 (s) を E_m と定義する．また全員が終点に集合するまでの制限時間を L とする．

まずエージェント m の経路において，ノード k に対して $x_{k,m}$ は以下のようになる．

$$x_{k,m} = \begin{cases} 0 \text{ (訪れていない)} \\ 1 \text{ (訪れた)} \end{cases} \tag{1}$$

この時，全エージェントが経路を通して得た利益の合計（発見出来たアイテムの合計）V は

$$V = \sum_{m=1}^{a} \sum_{k=1}^{p} V_k x_{k,m} \tag{2}$$

で表すことができる．$x_{k,m}$ はノード k を訪れていない時は 0, 訪れた時は 1 であるため，上記の式は訪れたノードのみの利益の合計であると言える．提案手法はこの式 (2) を最大化する $x_{k,m}$ の順列を見つけることで経路を算出する．

またエージェント m の経路の総コスト C_m は

$$C_m = E_m + \sum_{k=1}^{p} T_k x_{k,m} \tag{3}$$

で表すことができる．つまり，始点から終点の移動に要した時間と訪れたノードの探索に要した時間の合計である．

加えてエージェントの中で最も高い C_m（経路の総コスト）を $C_{\max}$ とした時，コスト制約として以下の制約を設ける．

$$C_{\max} \leq L. \tag{4}$$

また求めた経路全体の効率 R を以下のように定義する．

$$R = \frac{V}{C_{\max}}. \tag{5}$$

つまり利益を全員が集合するまでの時間で割ったものであり，R が高いほどアイテム探索の効率が高い経路であると言える．

3 既存の配送計画問題の拡張と定式化

本研究ではマルチエージェントの移動経路算出において最も代表的な手法である VRP 及び，提案手法と同様のコスト制約を設けることのできる DCVRP を比較対象とする．そのため 2.1 節と同様の前提条件でも成り立つよう拡張した問題を定式化し以下に記す．

3.1 VRP

VRP はコスト制約が無いため式 (4) 以外の提案手法と同様の式が成り立つ．また，訪れた地点数の合計 N を以下の式で表す．

$$N = \sum_{m=1}^{a} \sum_{k=1}^{p} x_{k,m}. \tag{6}$$

加えて VRP は全地点を訪れることが前提であるため，以下の制約を設ける．

$$N = p. \tag{7}$$

VRP は全地点を分担して訪れる経路の組み合わせの中で，全員が集合するまでの時間を最小化する問題であると考え $C_{\max}$ を最小化する $x_{k,m}$ の順列を見つける．これは最大値最小化問題であり，エージェント全体の集合を $\boldsymbol{g}$ とすると，その名の通り目的関数が

$$\begin{aligned} &\text{Minimize} : \max(C_m) \\ &\text{subject to} : m \subset \boldsymbol{g} \end{aligned}$$

となるため非線形である．しかしこれは次と等価である．

$$\begin{aligned} &\text{Minimize}: \zeta \\ &\text{subject to}: \zeta \geq C_m, m \subset \boldsymbol{g} \end{aligned} \tag{8}$$

なぜなら $\max(C_m)$ は全ての C_m の値以上のうちで，最小の値 ζ を求めることに等しいからである．あとは式 (3) より線形計画問題として以下の式

$$\begin{aligned} &\text{Minimize}: \zeta \\ &\text{subject to}: \zeta \geq E_m + \sum_{k=1}^{p} T_k x_{k,m},\ m \subset \boldsymbol{g} \end{aligned} \tag{9}$$

が，VRP の目的関数となる．

3.2 DCVRP

DCVRP はコスト制約付きの VRP であるため，式 (4) のコスト制約下で全地点を訪れることができる場合は VRP と同様である．しかし式 (4) の制約下で全地点を訪れられない場合は，なるべく多くの地点を訪れる経路が優れていることは明白なため式 (6) を最大化する $x_{k,m}$ を見つけるよう拡張する．

4 その他の制約

第 2 章および第 3 章で述べた制約の中で，まだ定式化していないものに関してこちらに記す．まず移動記録のデータについて a を「どのノードから」，b を「どのノードへ」，c を「対象となるエージェント」として $x_{a,b,c}$ と定義し，加えて探索の始点のノード番号を F，終点のノード番号を L，そのマップのノード全体の集合を $\boldsymbol{P}$，エージェント全体の集合を $\boldsymbol{g}$ とする．

この時，全員が同じ F から出発するという制約は，

$$\begin{aligned} &\sum_{i=1}^{p} x_{i,F,m} - \sum_{i=1}^{p} x_{F,i,m} = -1 \\ &\text{subject to}: m \subset \boldsymbol{g} \end{aligned} \tag{10}$$

となる．つまり「あるノードから F を訪れた回数」より「F からあるノードを訪れた回数」が 1 度だけ多ければ F から出発していると言える．m はエージェントの番号であり，全てのエージェントに対して同様の制約を加える．

次に全員が同じ L に集合するという制約は，

$$\begin{aligned} &\sum_{i=1}^{p} x_{i,L,m} - \sum_{i=1}^{p} x_{L,i,m} = 1 \\ &\text{subject to}: m \subset \boldsymbol{g} \end{aligned} \tag{11}$$

となる．つまり「あるノードから L を訪れた回数」の方が「L からあるノードを訪れた回数」より 1 度だけ多ければ L で経路が終わっていると言える．m に関しては式 (10) と同様である．

一方で F と L 以外のノードで経路が終わってはいけないため，

$$\begin{aligned} &\sum_{i=1}^{p} x_{i,j,m} - \sum_{i=1}^{p} x_{j,i,m} = 0 \\ &\text{subject to}: i \neq j, j \subset \boldsymbol{P}, m \subset \boldsymbol{g} \end{aligned} \tag{12}$$

となる．つまり F と L 以外のノードは，エージェントが訪れた回数と出発した回数が同じである．これを全ての F と L 以外のノードに制約として与える．

また同じ建物を探索しても非効率なため，誰かが一度探索した建物は訪れないよう制約を与える．まず提案手法と DCVRP は全地点を巡回する必要がないため以下の式となる．

$$\begin{aligned} &\sum_{k=1}^{a} \sum_{i=1}^{p} x_{i,j,k} \leq 1 \\ &\text{subject to}: i \neq j, j \subset \boldsymbol{P} \end{aligned} \tag{13}$$

あるノード j に対して，j 以外の全てのノードから，全てのエージェントが来たか否かを足して 1 以下あるいは 1 であれば，ノード j に訪れたのは多くとも 1 エージェントが 1 度だけということである．

一方 VRP は全地点を巡回する必要があるため次の式で制約を与える．

$$\begin{aligned} &\sum_{k=1}^{a} \sum_{i=1}^{p} x_{i,j,k} = 1 \\ &\text{subject to}: i \neq j, j \subset \boldsymbol{P} \end{aligned} \tag{14}$$

式 (13) と同様に，あるノード j に対して，j 以外の全てのノードから，全てのエージェントが来たか否かを足して 1 であれば，必ず j に訪れたのは 1 エージェントが 1 度だけということである．

式 (13) あるいは式 (14) 式の制約を全てのノードに対して与えることで，同じ建物を 2 度訪れてしまう非効率的な経路を避けることができる．

5 実装

5.1 データ構造

実装にあたり，ノードの数を p，エージェントの人数を a とした時に以下の 3 つのデータを保持する．

図 5 は，各ノードのデータであり，見つかるアイテム数は式 (2) の V_k，探索にかかる時間は式 (3) の T_k である．

	x 座標	y 座標	見つかるアイテム数	探索にかかる時間
ノード_0				
ノード_1				
⋮				
ノード_n				

図 5　各ノードのデータを保持する $p \times 4$ 行列.

図 6 は，a 人の各エージェントの移動を記録した物である．これは第 2 章および第 3 章の定式における $x_{k,m}$ に当たるが，実装上「どのノードから訪れたか」も図 6 のように保持する．

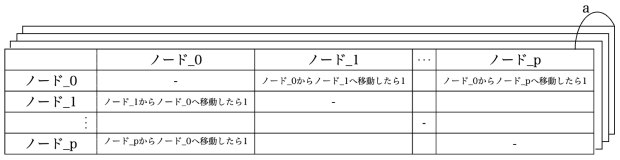

図 6　各エージェントの移動を記録する $p \times p \times a$ 行列.

図 7 は各ノード間の移動時間のデータである．本研究ではどのノード間でも往来できるため，ワーシャル-フロイド法 [26] による各ノード間の距離を更新する必要はなく，各データは単純な直線距離である．図 6 の各ノードデータと図 7 の各要素を乗算し合計することで，式 (3) の E_m を求める．なおノード_0 からノード_1 のデータと，ノード_1 からノード_0 のデータは明らかに等しいが，インデックスの入れ替え処理を削減するため両方保持することとする．

	ノード_0	ノード_1	⋯	ノード_p
ノード_0	-	ノード_0からノード_1への移動時間		ノード_0からノード_pへの移動時間
ノード_1	ノード_1からノード_0への移動時間	-		
⋮			-	
ノード_p	ノード_pからノード_0への移動時間			-

図 7　各ノード間の移動時間を保持する $p \times p$ 行列.

5.2 経路算出方法

第 2 章および第 3 章で述べた制約条件は，5.1 章のデータ構造を用いると線形計画問題 [27] として実装することができる．

本研究では，制約条件を PuLP[28] というライブラリを用いて算出した．線形計画法の算出アルゴリズムをコーディングする際，通常は目的関数や制約条件の係数から行列やベクトルの成分を抽出する工程が必要となるが，PuLP はそれらの成分を抽出せず，目的関数や制約条件をコードに直接記述するだけでよい．

PuLP によるコーディングは，大きく

- 数理モデル設定 (LpProblem メソッド)
- 変数設定 (LpVariable メソッド)
- 目的関数設定
- 制約条件設定
- ソルバー実行 (solve メソッド)

に分かれる．

数理モデルの設定は，目的関数を最大化するか最小化するかを設定するものである．以下のソースコード 1 にその具体的な例を示す．

ソースコード 1　数理モデル設定

```
problem = LpProblem(sense=LpMaximize)
```

変数の設定は LpVariable メソッドを用いる．以下のソースコード 2 は，図 6 の変数配列を生成する箇所のコード例である．

ソースコード 2　変数設定

```
# ary_a : エージェントの配列
# ary_p : 建物の配列

x = [[[ \
  LpVariable("x%s_%s,%s"%(i,j,k), \
  cat="Binary") if i != j \
  else None for k in range(ary_a)] \
  for j in range(ary_p)] \
  for i in range(ary_p)]
```

目的関数は，LpProblem で生成したインスタンスに対し「+=」演算子によって式を指定することにより設定できる．以下のソースコード 3 は，目的関数である式 (2) をコードとして記述した例である．lpSum メソッドは，

LpVariable による変数インスタンスの配列を入力し，その総和を求めるメソッドである．

ソースコード 3　目的関数設定

```
# ary_a : エージェントの配列
# ary_p : 建物の配列

problem += \
  lpSum(df.exp_item[i] * x[i][j][k] \
  if i != j \
  else 0 for k in range(ary_a) \
  for j in range(ary_p) \
  for i in range(ary_p))
```

制約条件は，目的関数と同様に「+=」演算子で条件式を指定することにより設定できる．以下のソースコード 4 は，制約条件の一つである式 (12) をコードとして記述した例である．

ソースコード 4　制約条件設定

```
# ary_p : 建物の配列
# j : 建物ごとのカウンター
# k : エージェントごとのカウンター

problem += \
  lpSum(x[i][j][k] for i in range(ary_p)) - \
  lpSum(x[j][i][k] for i in range(ary_p)) == 0
```

制約条件を列挙したら，最後に solve メソッドを呼び出すことで解算出処理を実行する．以下のソースコード 5 は実行処理のコード例である．

ソースコード 5　ソルバー実行

```
problem.solve()
```

なお，PuLP では線形計画法アルゴリズムとして COIN-OR Branch and Cut (CBC) ソルバー [29] が同梱されており，本手法でもこの CBC を採用した．

6　実験

人数やマップの広さ，制限時間の条件を変えて，提案手法と VRP，DCVRP で求められるそれぞれの経路と効率を比較する実験を行った．検証に使用したコンピュータの OS は Windows10，CPU は Intel Core i7-7820X，メモリは 32GB，プログラム言語は Python，ライブラリは PULP である．

6.1　実験 1

まずは一般的なバトルロイヤルゲームを想定し，

- 建物の棟数: 15
- 人数: 4
- 探索時間: 120(秒)

の環境下で地形をランダムに生成し実験に用いた．生成した地形図を図 8 に示す．

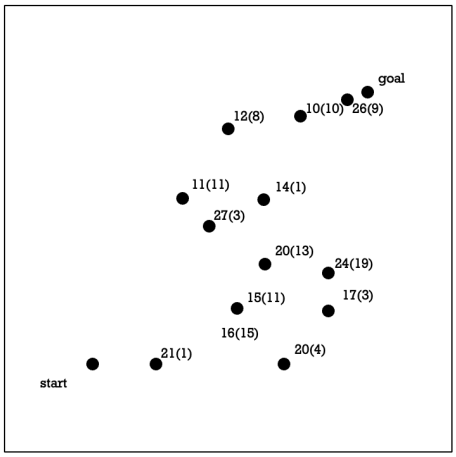

図 8　実験に使用した地形．

図 8 の黒点はノードであり建物を表す．また右上の 2 つの数字は

$$T_k(V_k)$$

という表記になっている．例えばグラフ中央の 14(1) のノードは 14 秒かけてアイテムが 1 つしか見つからない非効率的な建物であると言える．

各手法の実験結果を表 1 及び図 9, 10, 11 に示す．色分けされた線が各エージェントの移動経路である．

図 12 は 3 次元での実験結果のイメージである．色分けされた球体がエージェント，水色の直方体が出発点，ピンク色の直方体が集合地点，その他の直方体がアイテムを見つけられる建物である．出発点から集合地点に移動するまでに，制限時間の中でアイテムが得られる建物を探索している様子のシミュレーションである．エージェントがまだ探索していない建物は灰色で，探索した建物はエージェントの色に変わっている．図 13 は図 12 を上空から見たもので，それを二次元のグラフにプロットしたものが図 9, 10, 11 である．

表 1 の「発見数」は発見出来たアイテム数，「到着時

間」は始点から終点までにかかった時間，「効率」は発見数を到着時間で割った値である．

また，VRP および DCVRP での経路算出も提案手法と同様に Revised Simplex Method を用いて算出した．

表 1　実験 1 での各手法の比較．

	発見数	到着時間 (秒)	効率
提案手法	100	119.8	0.83
VRP	118	207.6	0.57
DCVRP	76	118.9	0.64

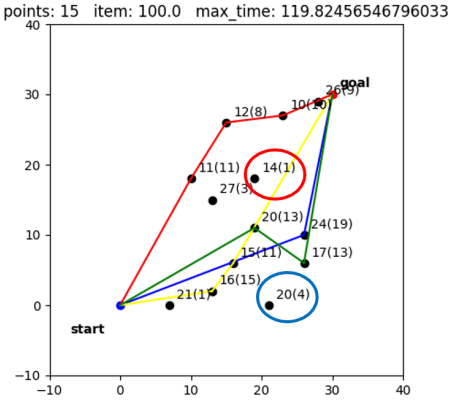

図 9　実験 1 の提案手法の結果．

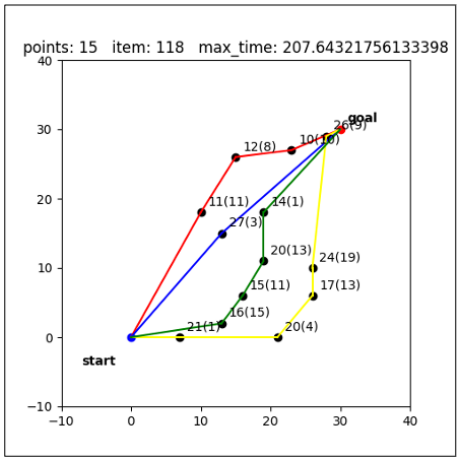

図 10　実験 1 の VRP の結果．

式 (2) を最大化する提案手法の方針で経路を探索した結果が図 9 である．グラフ中央の赤い円で囲んだ 14 秒かけて 1 つしかアイテムが見つからないような地点や，遠くで孤立していて移動に時間がかかるような非効率的な地点を避けることが出来ている．

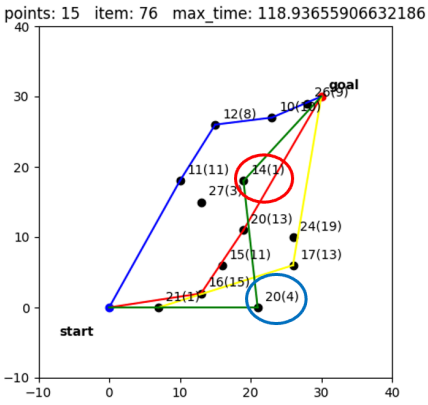

図 11　実験 1 の DCVRP の結果．

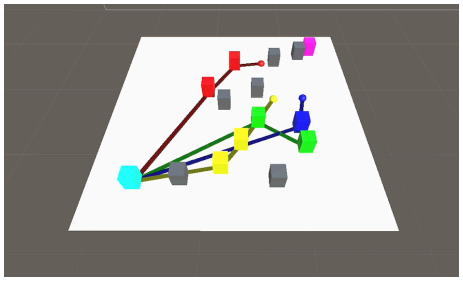

図 12　実験結果の 3 次元イメージ．

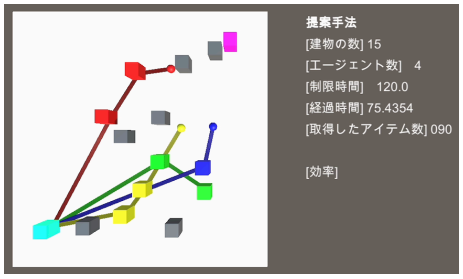

図 13　実験結果の 3 次元イメージ（上空）．

次に式 (4) のコスト制約がなく，全地点を分担して訪れる経路のうち $C_{\max}$ を最小化する VRP の解法で経路を探索した結果が図 10 である．VRP は制限時間が無い解法かつ全地点を訪れるため最も発見出来たアイテム数は多いが提案手法とさほど変わらず，制限時間を 87 秒も超過しているため冒頭で述べた通り実用性は低い．

次に DCVRP の解法で経路を探索した．式 (4) のコスト制約下では全地点を回れないことが分かっているため式 (6) を最大化する解法で経路を探索した結果が図 11 である．提案手法に比べ発見出来たアイテム数は少な

く，始点から終点までにかかった時間は変わらない．既存手法では避けられていた赤い円の非効率的な建物も探索してしまっている．

同様の条件で地形を 10 通りランダムに作成し，各手法を適用させ実験した結果の平均を表 2 に示す．

表 2　実験 1 での各手法の平均．

	発見数	到着時間 (秒)	効率
提案手法	95.7	119.7	0.79
VRP	107.2	207.6	0.52
DCVRP	72.2	119.3	0.61

エージェント数が 4 人での経路探索の場合，提案手法の定式を使い探索することが，汎用的に最も効率が良いことが分かった．

本手法および先行手法を用いて 10 回経路探索を実施し，処理時間の平均を表 3 に示す．

表 3　実験 1 における各手法の処理時間

	処理時間 (秒)
提案手法	8786.3
VRP	13.1
DCVRP	11709.1

提案手法と DCVRP の処理時間は非常に長く，VRP は比較して非常に短い処理時間となった．

6.2　実験 2

同様に近年流行している APEX LEGENDS[30] 等のタイトルを想定した実験環境でも実験した．こちらは今までのバトルロイヤルゲームとは違い 3 人パーティかつマップが比較的狭く作られているため，環境を

- 建物の棟数: 12
- 人数: 3
- 探索時間: 120(秒)

と設定した．各手法の実験結果を表 4 及び図 14, 15, 16 に示す．

提案手法の方針で経路を探索した結果が図 14 である．赤い円の建物は一つだけ離れているが 19 秒かけて 18 個もアイテムを得られる効率の良い建物であるため訪れている．一方で青い円の建物は始点からも他の建物からも近いが，29 秒かけて 6 個しかアイテムを得られず効率が悪いため避けている．

表 4　実験 2 での各手法の比較．

	発見数	到着時間 (秒)	効率
提案手法	84	119.9	0.70
VRP	122	286.1	0.43
DCVRP	65	119.9	0.54

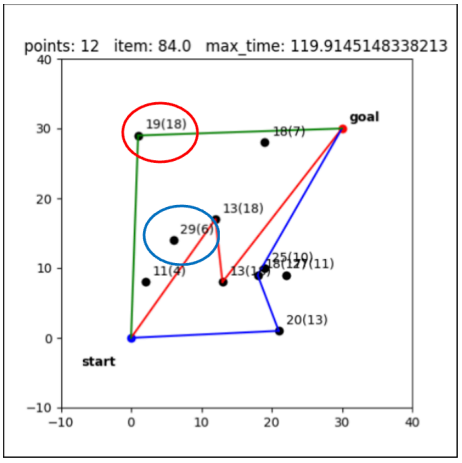

図 14　実験 2 での提案手法の結果．

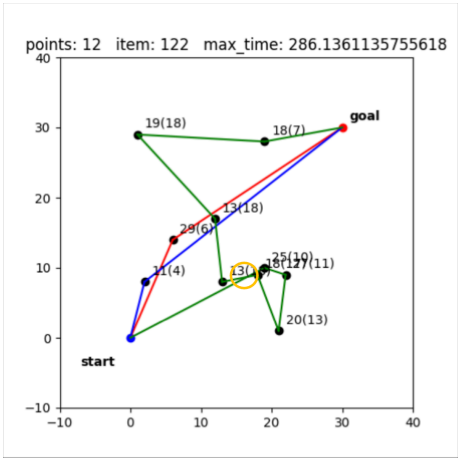

図 15　実験 2 での VRP の結果．

次に VRP の解法で経路を探索した結果が図 15 である．最短経路問題において経路が交差する場合は最適解ではないが，黄色の円で経路が交差してしまっている．これは他の手法にも同様に言えるが最適解ではなく近似解を求めるソルバー [28] を使用しているためである．

DCVRP の解法で経路を探索した結果が図 16 である．なるべく多くの地点を訪れるよう経路を選択するため効率が良い赤い円の建物を避けてしまっている．また効率

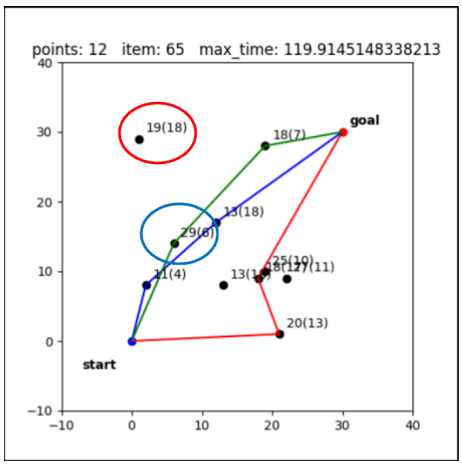

図 16　実験 2 での DCVRP の結果.

の悪い青い円の建物も訪れてしまっている．以上のことから得られたアイテム数及び効率が提案手法と比較し低くなった．

同様の条件で地形を 10 通りランダムに作成し，各手法を適用させた結果の平均を表 5 に示す．

表 5　実験 2 での各手法の平均.

	発見数	到着時間 (秒)	効率
提案手法	79.4	119.8	0.66
VRP	139.0	299.1	0.46
DCVRP	62.3	119.7	0.52

この結果から，エージェント数や地点数の変化に対してもある程度の汎用性が窺える．

本手法および先行手法を用いて 10 回経路探索を実施し，処理時間の平均を求めた結果を表 6 に示す．

表 6　実験 2 における各手法の処理時間

	処理時間 (秒)
提案手法	16.8
VRP	2.9
DCVRP	10.1

エージェント数を一人減らし，建物の数を 3 棟減らしただけで，実験 1 よりも大幅に時間を短縮する結果となった．

6.3　実験 3

実験 1 の条件に対して到着時間を 120 秒 から 300 秒にし，制限時間内にすべての建物を訪れることが出来る状況を設定した．今回は提案手法と VRP の解法を適用させた．表 7 および図 17, 18 に提案手法と VRP の結果を示す．なお DCVRP は制限時間内にすべての建物を訪れることが出来る場合に VRP と同じ経路を導くため割愛する．

表 7　実験 3 での各手法の比較.

	発見数	到着時間 (秒)	効率
提案手法	124	151.7	0.8
VRP	124	106.7	1.2

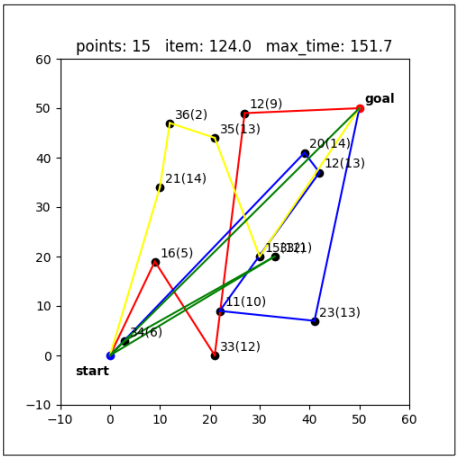

図 17　実験 3 での提案手法の結果.

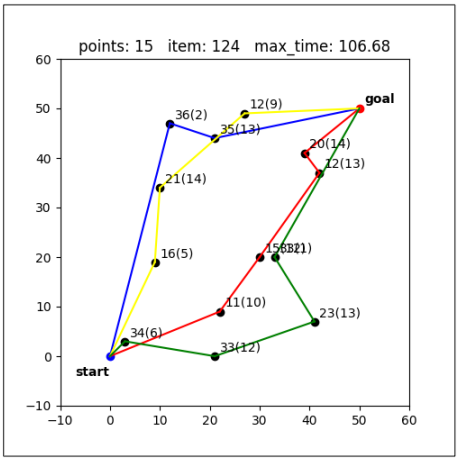

図 18　実験 3 での VRP の結果.

提案手法の方針で経路を探索した結果が図 17 である．経路を見ると，スタートから遠い建物を訪れた後にスタート付近の建物を訪れている様子が見られ非効率的であることが分かる．

VRP の解法で経路を探索した結果が図 18 である．始点から近い順に分担して建物を訪れている様子が見られる．提案手法においても制限時間内で全ての建物を訪れることが出来るため，発見できるアイテムは両手法変わらない．加えて到着時間を最小化する解法のため全員集合するまでの時間が提案手法より少なく，効率も高い結果となった．

同様の条件で地形を 10 通りランダムに作成し，各手法を適用させた結果の平均を表 8 に示す．

表 8　実験 3 での各手法の平均.

	発見数	到着時間 (秒)	効率
提案手法	152.1	201.7	0.75
VRP	152.1	135.4	1.12

この結果から，制限時間が余る場合は既存手法である VRP の方が適していることが分かる．

本手法および VRP を用いて 10 回経路探索を実施し，処理時間の平均を求めた結果を表 9 に示す．

表 9　実験 3 における各手法の処理時間

	処理時間 (秒)
提案手法	42.2
VRP	14.3

6.4　考察

6.1 節と 6.2 節に示した実験結果より，制限時間内に全ての建物を訪れることが出来ない状況下においては提案手法が有効であると考えられる．これはバトルロイヤルゲーム，ひいては制限時間などのコスト制約下でより利益の高い経路を求める際には，取捨選択の出来ない既存の配送計画問題の解法が適していないことを示している．

一方 6.3 節より，制限時間が余る状況下では提案手法は適していないことが分かる．原因は処理時間を見て分かる通り，提案手法はあくまで式 (2) の V を最大化する解法であるため，全建物を訪れることが出来た時点で経路のコストに関係なく最適解であると判断出来てしまうからだと考察する．

以上のことからバトルロイヤルゲームのアイテム探索など，想定されるコスト制約に幅がある環境においては特に，単一の移動経路算出手法ではなく複数を用いることが必要であると言える．

また処理時間に関しては，現状はどの手法もリアルタイムに活用できるスコアでは無い．しかし基本的にバトルロイヤルゲームのマップとアイテムが見つかる地点は変わらないため，事前に経路を算出して活用する方法でも，キャラクター AI のアイテム探索の効率化に寄与出来ると考える．

7　まとめと今後の展望

本研究ではバトルロイヤルゲームのアイテム探索に適した配送計画問題の定式を提案した．その上でアイテム探索に利用できる既存の理論として挙げた VRP, DCVRP の解法と経路の効率を比較をした．

その結果，コスト制約下で全地点を経由することが出来ない状況下では本手法が最も効率の高い経路を導くことが分かった．エージェント数やノード数を変えても有効な結果を出したためある程度の汎用性が確認出来た．

しかし，コスト制約下で全地点を経由することができる状況下では，既存の VRP, DCVRP の方が有効であることが分かった．

今回はあらかじめ実験に使用するグラフにエッジを設定しておらず，どのノード間でも直接移動できる環境下で実験した．そのためバトルロイヤルゲームにおいては建物が点在する市街地など屋外でのアイテム探索などで有効だと考えられる．今後の展望として，エッジを設定し移動できるノードを制限したグラフを使い，入り組んだマップや迂回する必要のあるマップでも有用性を示すことが出来れば，応用できるシチュエーションも増えると考えている．

参考文献

[1] KRAFTON INC., 2021. PUBG —KRAFTON. https://asia.battlegrounds.pubg.com/ja/. 参照: 2023.02.02.

[2] NetEase, Inc., 2017. 荒野行動 —NetEase Games. https://www.knivesout.jp/. 参照: 2023.02.03.

[3] 三宅陽一郎. ディジタルゲームにおける人工知能の応用の現在. 人工知能, Vol. 30, No. 1, pp. 45–64, 2015.

[4] Alan M. Gibbons. *Encyclopedia of Computer Sci-*

ence, pp. 755–759. John Wiley and Sons Ltd., 2003.

[5] Grantham KH Pang, K Takabashi, Takayoshi Yokota, and Hiroshi Takenaga. Adaptive route selection for dynamic route guidance system based on fuzzy-neural approaches. *IEEE Transactions on Vehicular Technology*, Vol. 48, No. 6, pp. 2028–2041, 1999.

[6] 林 良嗣; 槇谷 博光; 大島 邦彦; 中村英夫;. 大規模鉄道ネットワークにおける経路探索の簡略化手法に関する研究. 土木学会論文報告集, Vol. 338, , 1983.

[7] Zeyad Abd Algfoor, Mohd Shahrizal Sunar, and Hoshang Kolivand. A comprehensive study on pathfinding techniques for robotics and video games. *International Journal of Computer Games Technology*, pp. 7–18, 2015.

[8] Moh Zikky. Review of a*(a star) navigation mesh pathfinding as the alternative of artificial intelligent for ghosts agent on the pacman game. *EMITTER International journal of engineering technology*, Vol. 4, No. 1, pp. 141–149, 2016.

[9] Merrill M Flood. The traveling-salesman problem. *Operations Research*, Vol. 4, No. 1, pp. 61–75, December 1956.

[10] Ravindra K. Ahuja, Kurt Mehlhorn, James Orlin, and Robert E. Tarjan. Faster algorithms for the shortest path problem. *J. ACM*, Vol. 37, No. 2, pp. 213–223, apr 1990.

[11] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2002.

[12] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers and Industrial Engineering*, Vol. 99, pp. 300–313, 2016.

[13] G. Laporte, M. Desrochers, and Y. Nobert. Two exact algorithms for the distance-constrained vehicle routing problem. *Networks*, Vol. 14, pp. 161–172, 1984.

[14] Marcus Poggi, Henrique Viana1, and Eduardo Uchoa. The team orienteering problem: Formulations and branch-cut and price. In Thomas Erlebach and Marco Lübbecke, editors, *ATMOS*, Vol. 14, pp. 142–155, January 2010.

[15] C. Archetti, M.Grazia Speranza, and Daniele Vigo. *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. Society for Industrial and Applied Mathematics, November 2014.

[16] Zuzana Borcinova. Two models of the capacitated vehicle routing problem. *Croatian Operational Research Review*, Vol. 8, pp. 463–469, December 2017.

[17] C. Archetti, Dominique Feillet, Alain Hertz, and M.Grazia Speranza. The capacitated team orienteering and profitable tour problems. *JORS*, Vol. 60, pp. 831–842, June 2009.

[18] Claudia Archetti, Nicola Bianchessi, and M. Grazia Speranza. The capacitated team orienteering problem with incomplete service. *Optimization Letters*, Vol. 7, No. 7, pp. 1405–1417, 2013.

[19] 有熊 翼; 松冨 達夫; 木村有寿;. ファジィ性を考慮した時間枠付き多目的配送計画問題の解法. *Fuzzy System Symposium*, Vol. 27, , September 2011.

[20] Brian; Kallehauge. On the vehicle routing problem with time windows. Master's thesis, Technical University of Denmark, 2006.

[21] Malandraki; Chryssi and Daskin; Mark;. The maximum benefit chinese postman problem and the maximum benefit traveling salesman problem. *European Journal of Operational Research*, Vol. 65, pp. 218–234, 1993.

[22] Wang Hsiao-Fan; Wen Yu-Pin;. Time-constrained chinese postman problems. *Computers and Mathematics with applications*, Vol. 44, No. 3-4, pp. 375–387, 2002.

[23] Epic Games Inc., 2023. Fortnite —Epic Games. `https://www.fortnite.com/`. 参照: 2023.02.02.

[24] 柴山叶, 阿部雅樹, 渡辺大地. バトルロイヤルゲームのアイテム探索におけるマルチエージェント移動経路最適化. *NICOGRAPH2022*, No. F-2, pp. 1–8, 2022.

[25] William Ho, George TS Ho, Ping Ji, and Henry CW Lau. A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Engineering applications of artificial intelligence*, Vol. 21, No. 4, pp. 548–557, 2008.

[26] Stefan Hougardy. The floyd–warshall algorithm on graphs with negative cycles. *Information Processing Letters*, Vol. 110, No. 8-9, pp. 279–281, 2010.

[27] Alexander Schrijver; John Wiley and Sons;. *Theory of Linear and Integer Programming*. 1998.

[28] pulp documentation team., 2009-2023. PuLP Document. `https://coin-or.github.io/pulp/`. 参照: 2023.02.02.

[29] John Forrest and Robin Lougee-Heimer. CBC User's Guide. `https://www.coin-or.github.io/Cbc/`. 参照: 2023.05.06.

[30] Electronic Arts Inc., 2023. APEX LEGENDS —Electronic Arts. `https://www.ea.com/ja-jp/games/apex-legends`. 参照: 2023.02.02.

柴山　叶

2021 年会津大学コンピュータ理工学部卒業．2023 年東京工科大学大学院修士課程バイオ・情報メディア研究科メディアサイエンス専攻修了．芸術科学会会員．

阿部　雅樹

2008 年東京工科大学メディア学部卒業．2010 年同大学大学院修士課程バイオ・情報メディア研究科メディアサイエンス専攻修了．2016 年より同大学メディア学部実験助手，現在に至る．コンピュータグラフィックスやゲーム制作に関する研究に従事．芸術科学会会員．

渡辺　大地

1994 年慶応義塾大学環境情報学部卒業．1996 年慶応義塾大学政策・メディア研究科修士課程修了．2016 年岩手大学工学研究科デザイン・メディア工学専攻博士後期課程修了．博士 (工学)．1999 年より東京工科大学メディア学部講師．2017 年より同准教授，2020 年より同教授，現在に至る．コンピュータグラフィックスやゲーム制作に関する研究に従事．芸術科学会，情報処理学会，画像電子学会，人工知能学会会員．