筑波大学大学院博士課程

システム情報工学研究科修士論文

マルチエージェントモデルによる介護保険施設における火災時の避難誘導に関する研究

鎌田 智之

(リスク工学専攻)

指導教員 糸井川 栄一

２００８年３月

平成１９年度　システム情報工学研究科修士(　工学　)論文概要

マルチエージェントモデルによる介護保険施設における火災時の避難誘導に関する研究

専攻名　　リスク工学専攻
学籍番号 200620911 学生氏名　　鎌田 智之
指導教員名　　糸井川 栄一

本研究は，介護保険施設における火災時の避難を表現したシミュレーションモデルを構築し，入所者の避難完了時刻や施設内人数の時間経過，避難不能人数を比較することで，効果的な避難誘導の方法を導出し，介護保険施設における火災時の対応策や職員へのマニュアル作成等の一助となることを目的として行った．本研究では，２階建ての施設を対象とし，職員が避難誘導する際の誘導方法を様々に設定することや，入所者の居室設定が避難誘導へどのように影響するかについて，比較分析を行うことを主眼としていることが，先行研究と異なる特徴であると考える．

本研究で構築したシミュレーションモデルは，マルチエージェントモデルを基本としており，入所者と職員それぞれを主体的な行動をするエージェントとしてモデル化した．入所者の行動規範にはポテンシャルモデルによって進路決定を行い，どの試行においても同様の最短経路を辿るものと設定した．職員の行動規範にはネットワークを利用したダイクストラ法を用い，入所者に最短経路で接近し，避難誘導を行うものとした．

シミュレーション実験では，日中シナリオ，夜間シナリオを設定し，対象施設で実現可能な避難誘導方法をモデル化し，誘導方法や初期配置変更などによる避難誘導結果の比較分析を行った．また，夜間シナリオの一つとして，火災発生ケースを位置づけ，ある出火点の火災に対し，避難誘導の効果の評価を行った．

避難誘導シミュレーション実験を通じ，平等性の確保と避難不能人数の減少を達成するためには，移動能力が低い入所者を優先的に避難させることに一定の効果があることが明らかになった．一方で，避難完了時間の短縮のためには２階を優先した避難誘導，平均避難時間の短縮のためには１階を優先した避難誘導が効果的である結果が導出され，この２点についてはトレードオフの関係にあることが明らかになった．

目次

表リスト

図リスト

第1章　序論

1.1　研究の背景

総務省統計局[1)]によると，図 1.1.1 に示すとおり我が国における 65 歳以上の高齢者人口の比率は年々増加しており，全体の 20%を超える割合を占めている．また，厚生労働省[2)]によると，図 1.1.2 に示すとおり介護保険施設の定員数は増加傾向にあり，2025 年以降に団塊の世代が 75 歳以上の後期高齢者となることから，ますます高齢者を考慮した防災のあり方を模索していくことが必要になると予想される．なお，介護保険で利用できる施設をまとめたものを表 1.1.3 に示す．

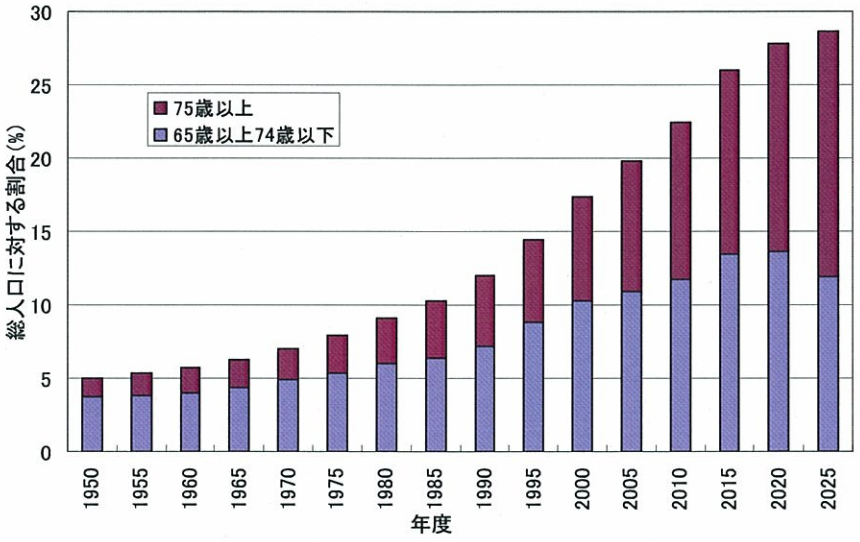


図 1.1.1　高齢者人口割合[1)]

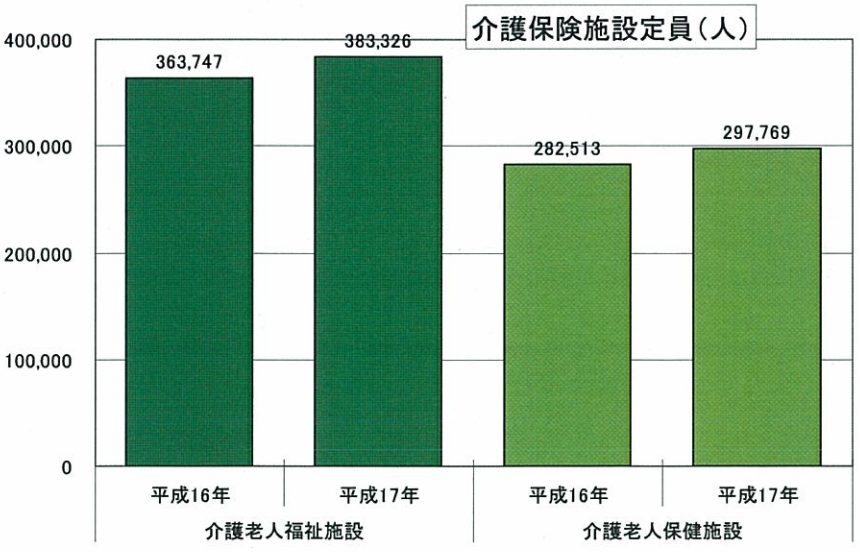


図 1.1.2　介護保険施設定員推移[2)]

表 1.1.3　介護保険で利用できる施設 [5,6)]

	認知症対応型共同生活介護（グループホーム）	介護老人福祉施設（特別養護老人ホーム）	介護老人保健施設
分類	地域密着型サービス	施設サービス	施設サービス
内容	認知症の利用者が，少人数で共同生活する場を提供．入浴，排泄，食事等の介護その他の日常生活上の世話，機能訓練などを行う．	常に介護が必要で自宅での介護が困難な人に対し，入所で介護や日常生活上の世話を提供する．	在宅復帰に向けてリハビリテーションを必要とする人に対し，入所で短期集中型の介護や看護，リハビリテーションを提供する．
施設数*[5)]	8,938	5,898	3,461

＊：施設数は 2007 年 4 月 30 日時点の全国計

そのような中，2006 年 1 月未明，長崎県大村市の認知症高齢者グループホーム「やすらぎの里さくら館」において火災時の在館者 10 名のうち死者 7 名，負傷者 3 名という全焼火災が発生した [3)]．死者 7 名は全て入所者であり，出火当時には施設職員は 1 名のみであった(軽傷)．この火災で夜間時の職員が少数であると，避難誘導，火災早期発見，消防機関への通報，初期消火などの対応への困難さが日中とは比較にならないものであることが明らかになった．また，入所者が認知症や災害時要援護者であることから上記の対応を期待することは難しく，さらには，施設周辺が住宅地ではないため救助等に近隣住民の助力を期待することが難しいケースもあると考えられ，事前対策によって被害のリスクを軽減することの重要性は非常に高いと言える．

一方で，文部科学省 [7)]の調査から加齢に伴う運動能力の低下は明らかであるため，健常者の運動能力を基礎とした試算だけで災害対策を全てカバーできるとは言い難い．しかし，現在の建築基準法をはじめとする諸法制度では高齢者や災害時要援護者に特化した基準等は見受けられない．介護保険施設は建築基準法の中では老人福祉施設として「児童福祉施設等」に含まれて位置づけられている [4)]が，例えば直通階段までの歩行距離については百貨店や料理店と比較して児童福祉施設等の方が長い距離を基準として設定されている．これは，百貨店等と比較して火災が起きにくいことを考慮しての値だと考えられるが，高齢者の中には，寝たきりで自力移動不可能，車椅子自走可能，自力歩行可能などと，歩行様態は様々であるため，一律に健常者の指標で設定したもの以上に避難に関する危険性は高いものであると予想される．

1.2 研究の目的

本研究では，介護保険施設における火災時の避難を表現したシミュレーションモデルを構築し，避難誘導方法や初期配置を変更する試行を行い，入所者の避難完了時刻や施設内人数の時間経過，避難不能人数を比較することで，効果的な避難誘導の方法を導出する．このことによって，介護保険施設における火災時の対応策や職員へのマニュアル作成等の一助となることを本研究の目的とする．

1.3 研究の構成

第2章では，本研究に関連する既往研究を整理し，本研究の位置づけを明らかにする．

(1) 介護保険施設における避難に関する報告書及び基礎的研究の整理
(2) 災害時要援護者を対象としたコンピュータシミュレーションを用いた避難誘導に関する既往研究の整理

第 3 章では，介護保険施設における火災時の効果的な避難誘導方法を求めるための，避難誘導シミュレーションを構築する．

(1) 避難誘導シミュレーションモデルの概要
(2) モデルの前提条件の設定及びデータの作成
(3) 避難誘導シミュレーションプログラムの構成
(4) 避難誘導シミュレーションの出力

第 4 章では，避難誘導シミュレーションを用いて，異なる避難誘導方針による試行や，入所者の配置を変更した試行などの比較分析を行う．

(1) シミュレーションを用いた比較分析の概要
(2) 入所者移動能力の考慮とリレー方式による避難誘導の影響分析
(3) 夜間シナリオにおける避難誘導方法の比較分析の概要
(4) 入所者移動能力と階別優先順位の考慮による避難誘導の影響分析
(5) 入所者の初期配置変更による避難誘導への影響の把握
(6) 火災発生シナリオにおける避難誘導方法の比較

第5章では，本研究の結論及び，本研究によって導かれる課題をまとめる．

(1) 本研究の結論
(2) 今後の課題

第2章　避難誘導に関する既往研究の整理

2.1 介護保険施設における避難に関する報告書及び基礎的研究の整理

2.1.1 認知症高齢者グループホーム等における防火安全対策検討会報告書：総務省消防庁，平成 18 年 3 月 29 日[3)]

2006 年 1 月未明，長崎県大村市の認知症高齢者グループホーム「やすらぎの里さくら館」で死者 7 名，負傷者 3 名の火災が発生した．火災の概要を以下に示す．

火災の概要

出火日時：平成 18 年 1 月 8 日（日）2 時 19 分頃（推定）
死傷者の状況：死者 7 名，負傷者 3 名
建物の用途：福祉施設（消防法施行令別表第一(6)項ロ）
初期消火状況：1 名の職員が消火器による消火を試みるが，消火できずに断念
避難・救出状況：避難誘導は特に行われていない

報告書より，認知症高齢者グループホームにおける防火上の問題点を次のように大きく 3 つの原因毎にまとめることができる．

夜間時職員が 1 名（もしくは少数）であることが原因で挙げられる問題

- 全入所者を短時間で避難させることが難しい
- 火災の発見が遅れる可能性が高い
- 消防機関に対する通報が遅れる可能性がある
- 初期消火ができない可能性がある
- 避難誘導方法が十分認識されていない場合がある

入所者が認知症や要介護者であることが原因で挙げられる問題

- 認知機能の低下によって火気等の適正な管理が行えない可能性がある
- 入所者が火災を覚知し，職員等に情報伝達を行うことを前提とする防火対策は検討しがたい
- 入所者が消防機関へ通報することを前提とする防火対策は検討しがたい
- 入所者が初期消火を行うことを前提とする防火対策は検討しがたい
- 入所者が屋外に出ることが難しい構造となっている場合がある（徘徊防止）

施設・設備上の問題

- 木造建築物としてグループホームは建設でき，防火区画がないことが一般的
- 自動火災報知設備・消防機関への通報設備・スプリンクラー設備の設置率を高める必要がある
- 周辺に民家や他の施設がない場合があり，近隣の援助が得られない可能性がある

2.1.2　介護保険施設における非常時の避難誘導に関する基礎的研究：山下恵，糸井川栄一：地域安全学会梗概集 No.21，P81-86，2007 年 11 月 [8)]

①概要

施設職員らへのヒアリングから介護保険施設における避難に関する問題点を整理し，実測調査により各避難行動にかかる所要時間や歩行速度原単位の観測を行っている．さらに，対象施設について避難安全性評価のケーススタディを行った．評価指標として避難時間のほかに，避難リスクを出火点と避難者の距離に反比例するものとして定義し比較を行っている．

②ケーススタディについて

出火箇所を 5 箇所，施設職員の誘導方法を 3 種類設定し，それぞれについて試行を行っている．特に誘導方法については，2 階を優先して避難誘導する場合において，1 階まで下ろした後に入居者を一旦待機させ別の職員にリレーする方法に取り組んでいる．

③研究の結論

- ベッド以外での避難が可能な入居者は，ベッドを使用しない避難をさせたほうがよい
- 1 階出火の場合は，1 階を優先して避難させたほうがよい
- 避難経路やエレベータに近い居室及び 1 階居室からの出火は特に危険である

④課題

特定の施設を用いたケーススタディのため，この研究での結果が他の施設での避難に必ずしも当てはまるわけではなく，より一般的な避難特性と効果的な避難方法を明らかにする必要がある点が課題として挙げられる．

また，シミュレーションモデルの構築によって条件を様々に変化させることができれば，より精度の高い結論を導くことができるであろうと考えられる．

2.2 災害時要援護者を対象としたコンピュータシミュレーションを用いた避難誘導に関する既往研究の整理

2.2.1 道路閉塞・火災延焼被害を考慮した地震時における災害時要援護者救助シミュレーション：上田遼，瀬尾和大，元木健太郎：地域安全学会論文集，No.8，P341-348，2006 年 11 月 [9)]

①概要

道路閉塞，火災延焼，及び人間行動の各モデルを統合して，要援護者救助のシミュレーションを行う手法を提案し，それを利用したケーススタディを通じて有用性を示している．人間行動の記述にはマルチエージェントモデルを採用している．

②道路閉塞と火災延焼被害の考慮

- 道路閉塞および出火点はシミュレーション 1 回ごとに設定している
- 全てのエージェントは道路閉塞を通過できない
- 火災が隣接したら巻き込まれたものとして死亡とみなす

③要援護者救助モデルの構築

防災リーダーエージェント，救助エージェント，要援護者エージェントの 3 種類のエージェントを用意している．以下にそれぞれのエージェントの役割を示す．

[防災リーダーエージェント]

- 要援護者エージェントの居場所情報を保有
- 救助エージェントに要援護者救助を指示

[救助エージェント]

- 防災リーダーの指示を受け，要援護者エージェントを救助・搬送

[要援護者エージェント]

- 自力で移動できずに救助エージェントの救助を受ける

④要援護者救助戦略のケーススタディ

2種類の救助優先順位と2種類の情報伝達方法の組み合わせで4種類のケーススタディを行っている．それぞれの組み合わせを表 2.2.1 に示す．

表 2.2.1 ケーススタディによる救助戦略の組み合わせ

救助対象の要援護者の選択方法(優先順位) ／ 防災リーダーの移動ルール(情報伝達方法)	救助優先順位 A 近隣優先型 救助エージェントの居場所から近い要援護者を優先	救助優先順位 B 火災地域優先型 火災現場に近い要援護者を優先
情報伝達方法 A　移動型 ランダムウォーク	移動＋近隣優先型	移動＋火災地域優先型
情報伝達方法 B　待機型 避難所に向かい待機	待機＋近隣優先型	待機＋火災地域優先型

2.2.2　避難シミュレーションに基づく高齢者施設の避難安全性の確保に関する考察：海老原学，掛川秀史：日本建築学会計画系論文集，第521号，P1-8，1999年7月[10)]

①概要

それまでの高齢者施設での火災事例を概観し，高齢者施設の避難安全性の確保に対して影響を及ぼしそうな要因を整理し，それらの要因に関して避難及び煙流動シミュレーションモデルを用いての検討を加えている．シミュレーションには海老原ら(1995)（2.2.3 において後述）でのモデルを用いており，また，介助者や被介助者に関する避難行動特性を実測などからパラメータとして導出している．

②シミュレーションモデルについて

海老原ら(1995)によるシミュレーションモデルを用いている．この研究で用いた介助者に関する避難行動特性として設定したパラメータを表2.2.2に示す．

表2.2.2　避難行動特性に関するパラメータ

	水平歩行速度	介助器具への移し替えに要する時間	開口部の通過に要する時間	占有面積
自力避難可能な入所者	0.5m／秒			0.5 ㎡
車椅子での介助を要する入所者	1.5m／秒	15秒	9秒	0.7 ㎡
ストレッチャーでの介助を要する入所者	1.5m／秒	25秒	9秒	1.0 ㎡
介助者	2.0m／秒			0.5 ㎡

③ケーススタディの条件

ケーススタディでは中廊下型と回廊型の仮想の2平面の居室での出火を仮定したシナリオの下で，介助者数，初期配置，防災設備対策，熱伝播の影響を変化させて比較分析を行っている．避難者数は自力避難可能者20名，車椅子やストレッチャーが必要な避難者36名とし，介護者はケースに応じて3，6，9名と変化させている．以上をまとめたものを表2.2.3に示す．

表2.2.3　ケーススタディ条件まとめ

平面形態		熱伝播の影響	防火設備対策		防火管理対策	
中廊下型	回廊型		スプリンクラー	水平避難区画	介助者数	初期配置
Case A1	Case B1	—	—	—	3	グループ
Case A2	Case B2	—	—	—	6	グループ
Case A3	Case B3	—	—	—	9	グループ
Case A4	Case B4	○	—	—	3	グループ
Case A5	Case B5	○	—	—	6	グループ
Case A6	Case B6	○	—	—	9	グループ
Case A7	Case B7	○	—	—	3	ランダム
Case A8	Case B8	○	○	—	3	グループ
Case A9	Case B9	○	—	○	3	グループ

2.2.3　オブジェクト指向に基づく避難・介助行動シミュレーションモデル：海老原学，掛川秀史：日本建築学会計画系論文集，第 467 号，P1-12，1995 年 1 月 [11)]

①概要

高齢者施設を対象に，建築物利用者の避難行動上の条件の違いを考慮でき，個人単位の局所的な状況変化に応じた避難・介助行動を予測できるシミュレーションモデルの開発を行っている．

②モデル化の基本的な考え方及び各モデルの概要

オブジェクト指向に基づいて，空間・人間・煙流動の 3 要素をモデル化している．以下にそれぞれのモデルについての説明を示す．

[空間モデル]

建築物は複数の階層の集合体，階層は複数の部屋の集合体として定義している．階層と階層を接続する情報は階段に，部屋と部屋の接続情報をドアに持たせている．以上をモデル化した概念図を図 2.2.4 に示す．

[人間モデル]

- 自力避難者クラス・要介助者クラス・介助者クラスと，3 つのタイプを定義している
- 各々の占有面積は等価の円形としている
- 歩行速度は群集密度の影響を受けない

移動目標の決定について

長期移動目標を満足するために短期移動目標を選択しながら避難行動を行う

長期移動目標：最終的に避難（移動）したい場所
短期移動目標：長期移動目標を満足するために当面到達（通過）したい場所

移動方向及び移動位置の決定について

移動目標となるドアや誘導灯からは引力が，避難を阻害する物体からは斥力が作用するものとして，ベクトルを合成して決定している移動方向及び移動位置の決定に関する概念図を図 2.2.5 に示す．

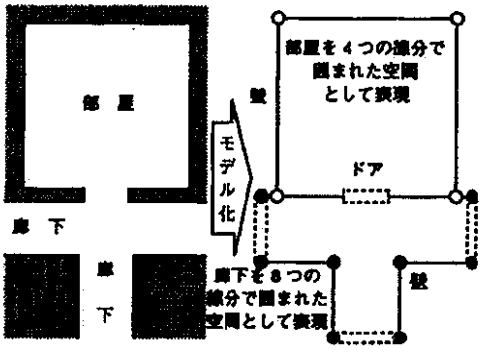


図 2.2.4　空間のモデル化の概念図

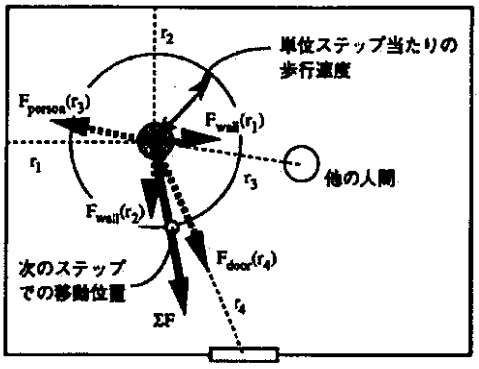


図 2.2.5　移動方向及び移動位置の決定

[煙モデル]

独立の煙流動シミュレーションの結果から，空間ごとに避難限界時間 t_e を以下の式として算定している．

$$\int_{t_s}^{t_e}(\Delta T)^2 dt > 4000 \qquad \Delta T：煙層の温度上昇(K) \qquad [2.2.6]$$

を満たす t_e を避難限界時間とする．

ただし，煙曝露開始時間 t_s は次式をはじめて満たした時間とする

$$s < 1.6 + 0.1H \qquad s：煙層境界高さ(m)，H：天井高(m) \qquad [2.2.7]$$

2.3 本研究と先行研究の相違点

山下(2007)[8]では，避難誘導方針の比較を主眼としているが，1 階または 2 階のどちらを優先するかまでの言及にとどまっており，入所者の特性に応じた避難誘導方針についてのケーススタディはされていない．

上田ら(2006)[9]では，市街地での要援護者援助のシミュレーションの開発に主眼が置かれている．従って，複数階での誘導戦術の考慮はなされていない．また，要援護者の移動レベルは自力移動不可として一律に設定されている．

海老原ら(1999,1995)[10,11]では，高齢者施設内の避難や介助行動をモデル化しケーススタディを行っているが，初期配置や空間が仮想のものであり，上下階の移動は考慮していない．また，介助者の誘導方針による比較には取り組んでいない．

本研究では，2 階建ての施設を対象とし，職員が避難誘導する際の誘導方法を様々に設定することや，入所者の居室設定が避難誘導へどのように影響するかについて，比較分析を行うことを主眼としているため，先行研究と大きな重複はないものと考える．

第3章　避難誘導シミュレーションモデルの構築

3.1 避難誘導シミュレーションモデルの概要

本章では，効率的な避難誘導方策を策定するための避難誘導シミュレーションモデルについて説明する．シミュレーションの基本的なモデルとしてマルチエージェントモデルを用いる．

3.1.1 マルチエージェントモデル及び artisoc について

マルチエージェントモデルは，自らの価値基準に従って自分の行為を自由に選択するような主体性を持ち自律的に行動するエージェントが多数共存するモデルを指す[12]．

なお，マルチエージェントモデルによるシミュレータの作成には株式会社構造科学研究所の artisoc をプラットフォームとして用いる．

3.1.2 対象施設の設定

本研究の対象施設の平面図を図3.1.1に示す．対象とする施設はA市のある介護老人福祉施設であり，昭和60年7月に開設され，鉄筋コンクリート造2階建ての構造とである（但し，平成5年に鉄骨造を一部増築）．介護老人福祉施設は，常に介護が必要で自宅での介護が困難な人に対し，入所で介護や日常生活上の世話を提供する施設[6]とされている．

平常時の上下階の移動は，平面図中央の階段とエレベータを用いている．平面図右の階段は外階段であり，平常時には利用されていない．外部空間への退出には玄関・非常口の他に，全ての 1 階居室の扉からも可能な空間構成となっている．ある時点での入居状況は表 3.2.6 で後述する．

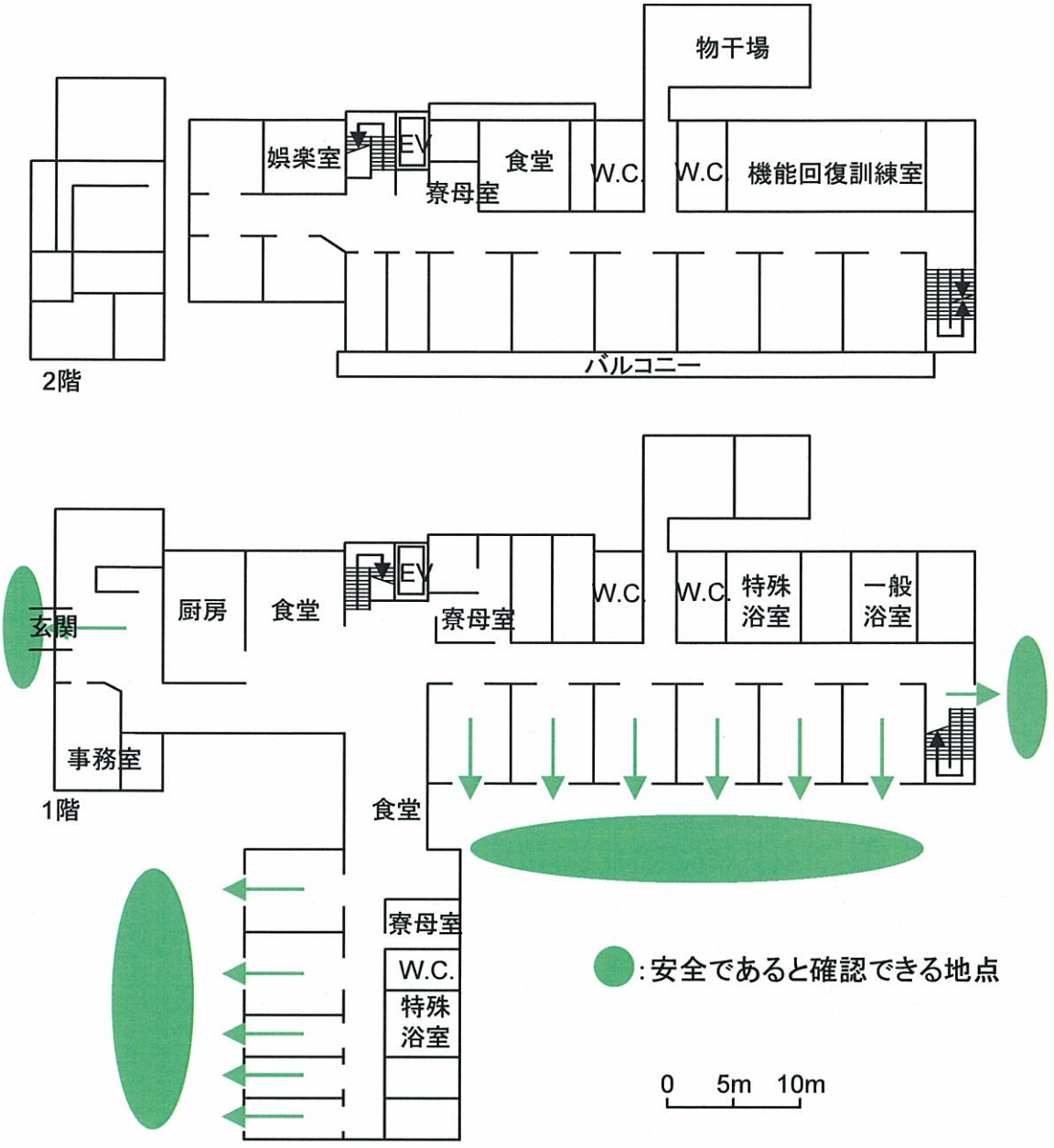


図 3.1.1　対象施設平面図

3.1.3　空間と各エージェントの構成について

作成する避難誘導シミュレーションにおける，空間とエージェントの名称及び構成は以下の通りである．

各エージェントは X,Y 座標による位置情報を持っており，エージェント同士でその位置情報を参照することで職員エージェントが入所者エージェントに接近する動作などを表現している．

対象施設が 2 階建てであることを表現するために，シミュレーションにおける空間は二つ用意し，それぞれの空間内に各エージェントが所属しているツリー構造である．それぞれの機能については 3.2 以降で説明する．なお，本避難誘導シミュレーションにおける 1 グリッドは 25cm×25cm と設定した．

1 階フロア
- ├─壁エージェント
- ├─床エージェント
- ├─ノードエージェント
- ├─入所者エージェント
- └─職員エージェント

2 階フロア
- ├─壁エージェント
- ├─床エージェント
- ├─ノードエージェント
- ├─入所者エージェント
- └─職員エージェント

3.2 モデルの前提条件の設定及びデータの作成

3.2.1 対象施設のモデル化及びポテンシャル・ネットワークの設定

対象施設の平面構成を避難誘導シミュレーションで表現するために，空間を構成するエージェントとして壁エージェントと床エージェントを設定する．また，職員エージェントが入所者エージェントに接近する際に参照するネットワークを設定するためにノードエージェントを設定する．

(1) 壁エージェント

平面図の空間構成を参考に壁や扉として通過できない箇所に設置する．後述するポテンシャルには大数を与えて通過不能としている．

本シミュレーションにおける壁エージェントの設置数は 1 階．これらのエージェントに与えるポテンシャルは初期値として予め算出したものであり，シミュレーション試行中に計算を行うものではない．

(2) 床エージェント

施設内から出口への経路選択をする際に，各グリッドにポテンシャルと呼ばれる数値を与え，それを参照して移動方向を決定するものとする．その際に，ポテンシャルを格納するエージェントとして床エージェントを設置する．

本シミュレーションにおける床エージェントの設置数は 1 階 20,099 個，2 階 10,326 個である．これらのエージェントに与えるポテンシャルは初期値として予め算出したものであり，シミュレーション試行中に計算を行うものではない．

a) 対象施設空間構成の設定

居室や廊下の空間構成を参考に，対象施設の廊下や居室を区分したものを図 3.2.1 に示す．色付きの部分は床エージェントが設定され通行可能である．また，色が切り替わっている箇所に目的地エージェント（後述の目的地エージェント j）を設定する．

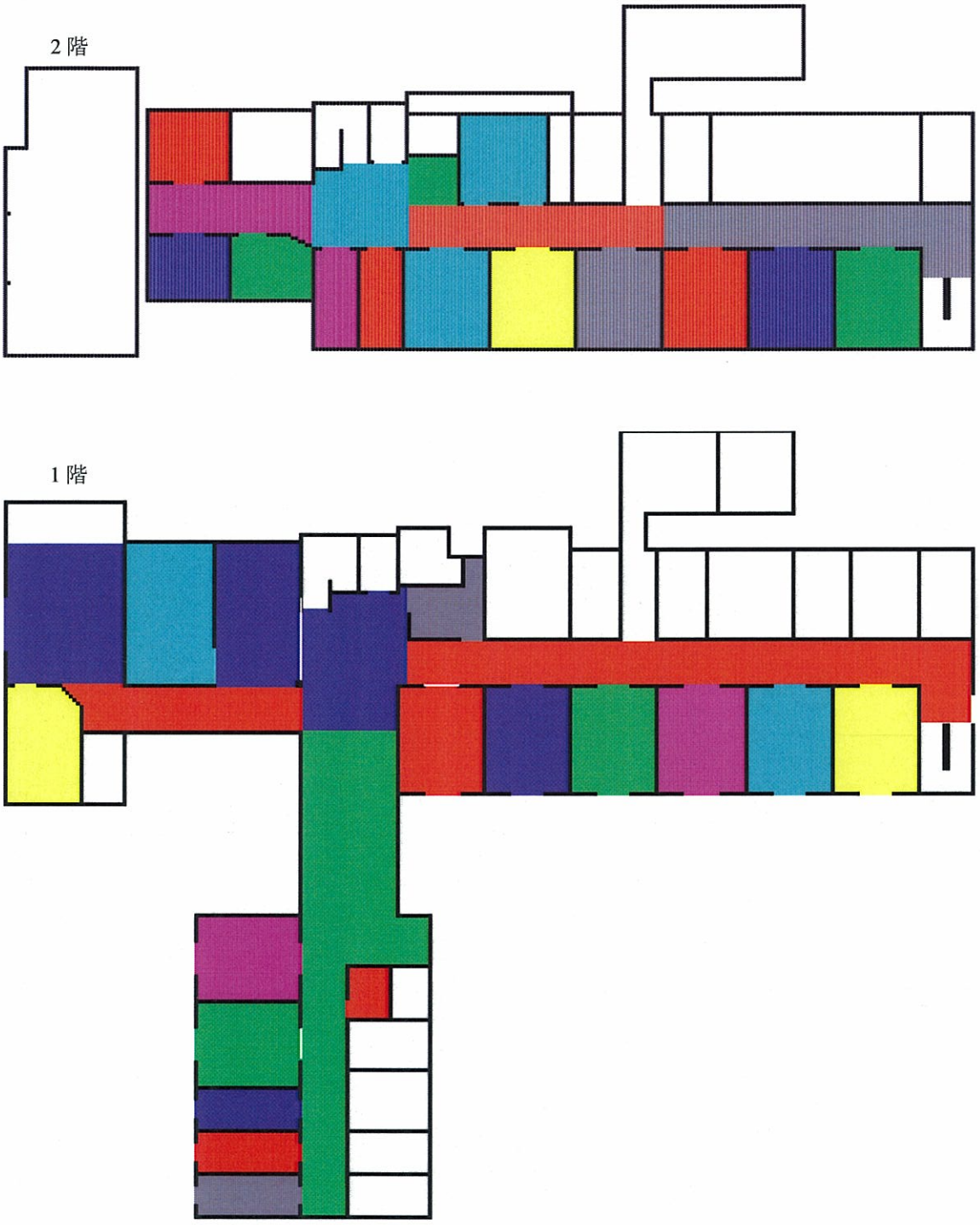


図 3.2.1　対象施設空間構成設定

b) ポテンシャルの設定

居室と廊下それぞれで目的となる床エージェント j までの距離 d_{ij} を算出し，エージェント j のポテンシャルを足し合わせたものを i のポテンシャルとする[式 3.2.2]. 出口から遠いほどポテンシャルの値は大きく，出口に近いほど値は小さくなるため，この値を小さいところへ辿ることで，各床エージェントから出口への最短経路がそれぞれ一通り求められることになる.

さらに，壁エージェントの周辺のエージェントには Pw_i のポテンシャルを与え，過剰に壁に接近しないことを表現している. 図 3.2.3 にポテンシャル設定の模式図を示す.

$$P_i = P_j + d_{ij} + Pw_i \qquad [3.2.2]$$

P_i：対象床 i のポテンシャル

P_j：目的地 j のポテンシャル

d_{ij}：床 i から床 j までの距離

Pw_i：壁付近の床に加えるポテンシャルの増分

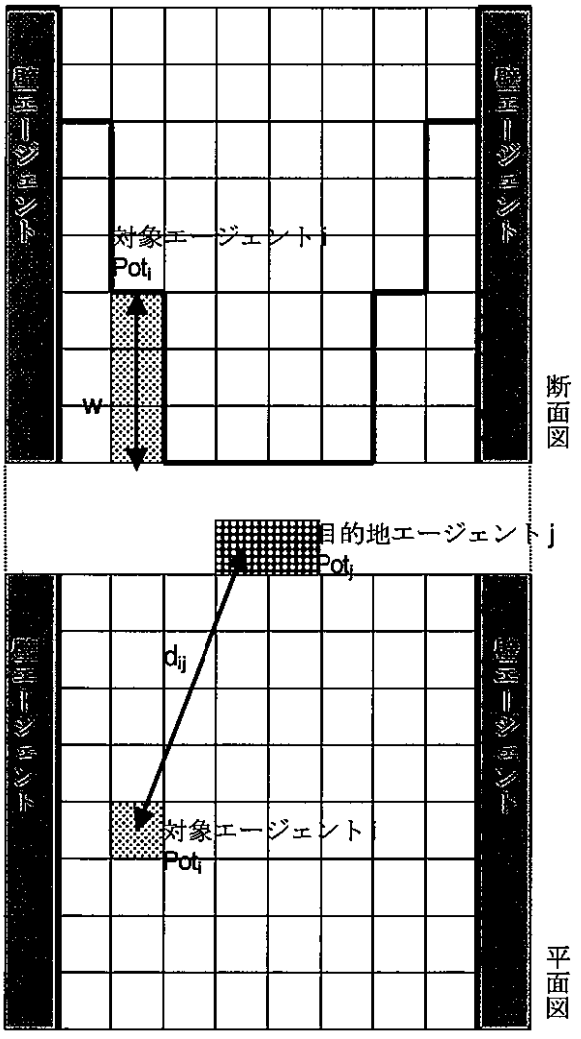


図 3.2.3　ポテンシャル設定概念図

(3) ノードエージェント

職員が対象施設内を移動する際に参照するネットワークを以下の図 3.2.4 に示す．数字は設置したノードエージェントの ID を示し，実線で接続されているノード間に接続関係を与えている．

避難誘導シミュレーションでは，職員エージェントと入所者エージェントのそれぞれについて，最も近いノードエージェントを算出し，2 点間の最短経路をダイクストラ法 [13]によって決定することで職員エージェントが入所者エージェントに接近することを表現する．

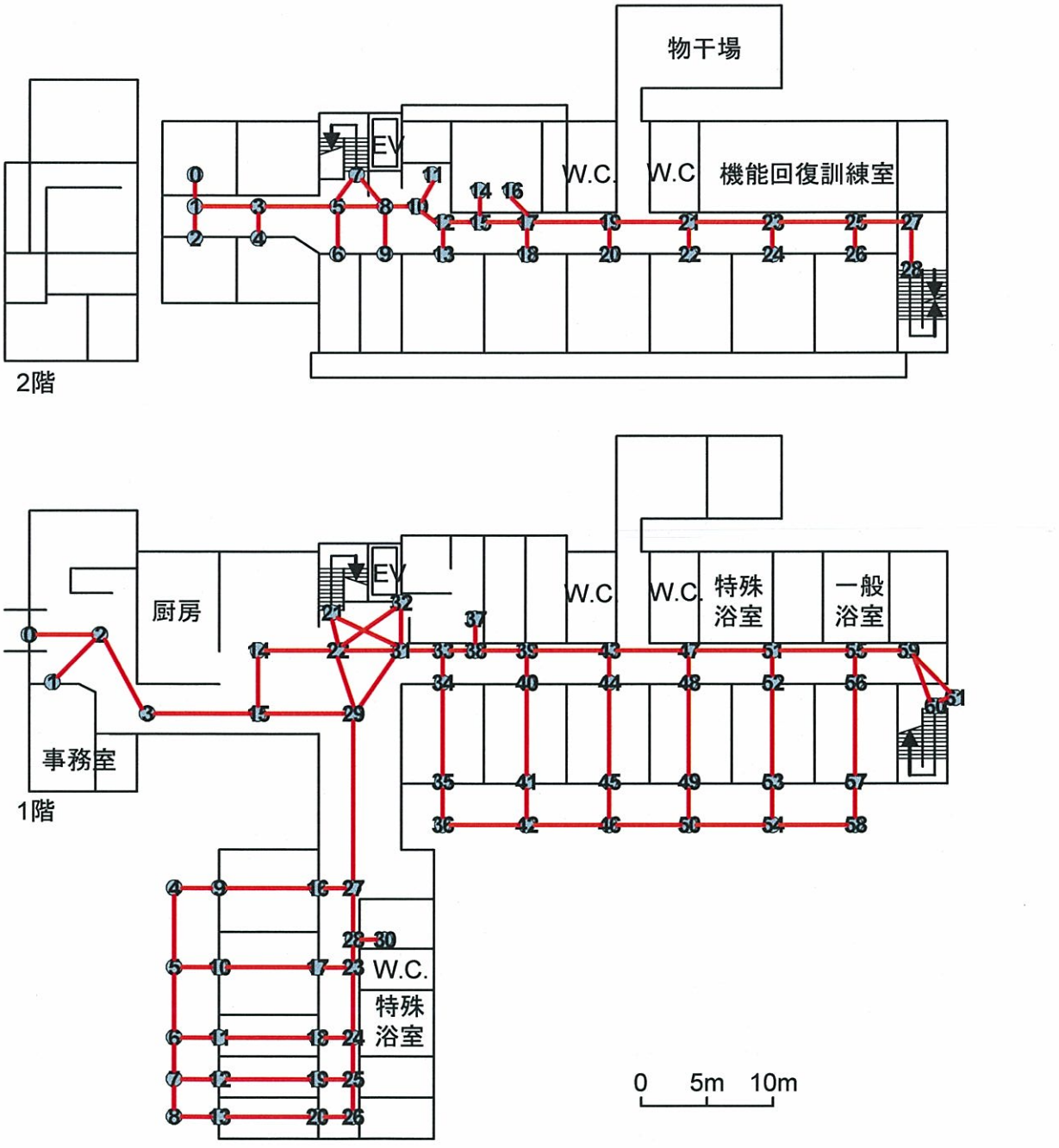


図 3.2.4　ネットワーク設定模式図

3.2.2　避難誘導シナリオの設定

避難誘導シミュレーションモデルを構築するにあたって，対象施設において入所者や職員の初期位置を設定するが，それらを日中・夜間それぞれシナリオとして設定して与えることとする．

日中・夜間ともに入所者は 1 階 26 名，2 階 26 名である．職員人数は日中については 1 階 20 名，2 階 18 名である．夜間は避難開始時には 3 名の職員が事務室にいるものとし，山下(2007)[8]を参考に職員が参集するものとする．ある時点での実際の居住状況を参考にした，日中・夜間の基本的な初期配置を図 3.2.5，表 3.2.6 に示す．

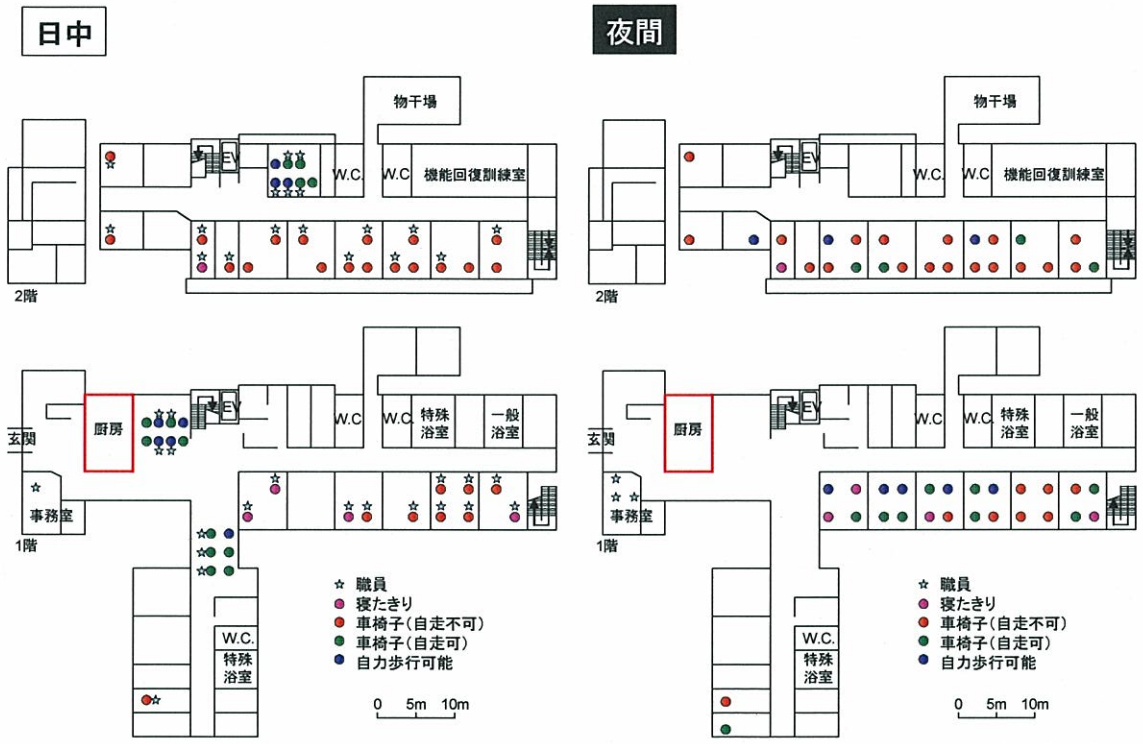


図 3.2.5　入所者・職員エージェント初期配置（四角の居室は火災設定対象）

表 3.2.6　入所者エージェント人数・移動能力

	寝たきり	車椅子（自走不可）	車椅子（自走可能）	自力歩行可能	合計
2 階	1 人	18 人	4 人	3 人	26 人
1 階	4 人	8 人	9 人	5 人	26 人
合計	5 人	26 人	13 人	8 人	52 人

(1) 日中シナリオ

日中でのシナリオとして，食事中の施設内の状況を取り上げて各エージェントの設定を行う．

a) 初期配置

寝たきりで食堂で食事を取ることができない入所者はそれぞれの居室におり，職員が対応していることとする．

自力移動可能な入所者は，食堂で食事していることとし，居室対応していない職員数名は食堂対応しているものとする．このとき，車椅子自走可能な入所者は，車椅子に乗っている状態で食事をとっているものとする．

また，館内放送によって全館同時に火災を覚知することを表現するため，1名の職員は事務室にいるものとする．このことによって避難誘導開始時刻は全てのエージェントについて同時である．

b) 避難開始時刻

館内放送で全館に知らせるものとし，全てのエージェントについて同時に開始とする．

c) 2階から1階への移動方法

対象施設にはエレベータが 1 箇所に設置されているが，火災による避難を想定とするため，使用できないものとし，全ての上下階移動は2箇所の階段を利用するものとする．

(2) 夜間シナリオ

夜間のシナリオとして，少数の職員が当直として事務室におり，全ての入所者は睡眠中である状況を取り上げ，各エージェントの設定を行う．

a) 初期配置

夜間は全ての入所者は居室ベッドに，職員は事務室にいるものとする．

b) 出火箇所

出火箇所は厨房とする．

c) 避難開始時刻

館内放送で全館に知らせるものとし，全てのエージェントについて同時に開始とする．

d) 誘導人員数の変化

職員の参集を考慮し，時間経過ごとに職員エージェントを追加する．山下(2007)[8]を参考とした職員の到着時刻の設定を表 3.2.7 に示す．また，到着した職員エージェントが避難誘導を開始する地点を図 3.2.8 に示す．

e) 2階から1階への移動方法

対象施設にはエレベータが 1 箇所に設置されているが，火災による避難を想定とするため，使用できないものとし，全ての上下階移動は2箇所の階段を利用するものとする．

表 3.2.7　職員到着時刻・人数の設定 [8)]

避難開始から経過時間（秒）	到着職員数（人）	累積職員数（人）
0	3	3（避難開始時の職員数）
300	1	4
420	1	5
480	3	8
540	2	10
660	1	11
780	1	12
840	1	13
900	3	16
1020	1	17
1140	1	18
1200	1	19
1320	1	20

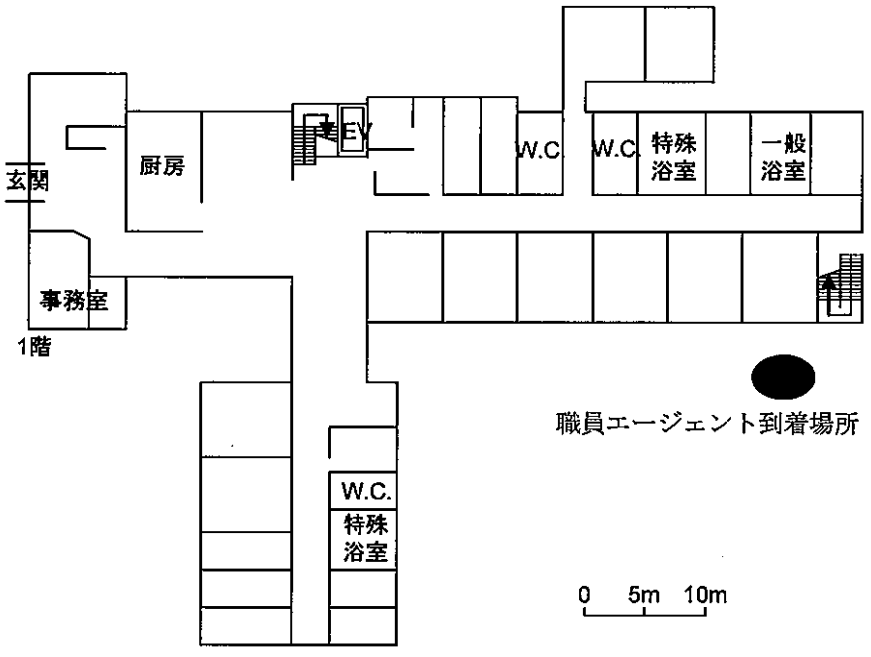


図 3.2.8　職員エージェント到着場所（赤丸部分）

3.2.3　入所者エージェントの設定

(1) 初期配置

日中シナリオ及び夜間シナリオに則して，それぞれ図 3.2.5（前掲）のように配置する．入所者エージェントの数は 1 階，2 階ともに 26 名ずつである．

(2) 移動能力及び移動手段

入所者の移動能力を示すデータとして，自力移動レベルを初期設定として与える．自力移動レベルは，歩行可能・車椅子自走可能・車椅子自走不可・寝たきりの 4 種類とする．職員エージェントによる避難準備作業によって移動手段が変化するものとし，その際の移動手段とは，自力歩行，背負い，車椅子自走，車椅子介助付移動，ベッド介助付移動の 5 種類とする．

車椅子自走可能な入所者は，日中シナリオでは車椅子に乗って食事をしていることを想定しているため自力移動による避難を開始するが，夜間シナリオではベッドでの睡眠中を想定し自力でベッドから車椅子に乗ることは困難だと考え，自力移動による避難は行わない．

なお，避難誘導シミュレーションにおいて，入所者エージェント同士が重ならないように，入所者エージェント自身のポテンシャルとして大数を与える．

(3) 歩行速度

歩行速度を移動手段別に設定したものを表 3.2.9 に示す．

表 3.2.9　移動手段別歩行速度及び必要リソース [8)]

移動手段	歩行速度	必要介助者数
自力歩行	0.63m/秒	0
背負い	1.50m/秒	1
車椅子自走	0.35m/秒	0
車椅子介助付移動	1.72m/秒	1
ベッド介助付移動	1.29m/秒	1

(4) 行動規範

避難開始と同時に，最も近い退出口に向かう．自力移動不能な入所者はその場で待機し，職員の到着に応じて移動手段が定まる．

進行方向は，自エージェントの周囲のエージェント（床・壁・人を問わず）の中から，最も低いポテンシャルを持つ床エージェントに向かって移動をするものとする．

ポテンシャルが 0 の床エージェントまで到達すると，避難完了とみなす．

ポテンシャルを用いた移動モジュールについては 3.3.1 で，階段降下については 3.3.3 で後述する．

3.2.4　職員エージェントの設定

(1) 初期配置

日中シナリオ及び夜間シナリオに則して，それぞれ図 3.2.5（前掲）のように配置する．職員エージェントの数は日中シナリオでは 1 階 20 名，2 階 18 名であり，夜間シナリオでは 1 階事務室に 3 名いる状況から開始し，表 3.2.7（前掲）に則り職員が到着するものとする．

(2) 属性及び移動方法

全ての職員エージェントは同様の移動能力とし，入所者を避難誘導していない状態の移動速度は 3.11m/秒[8]とする．

なお，避難誘導シミュレーションにおいて，入所者エージェントと重ならないようにポテンシャルとして大数を与える．

(3) 避難誘導方針

避難誘導する入所者エージェントを決定するために，職員エージェントは避難誘導方針に則り行動する．設定可能な避難誘導方針を以下に示す．

a) 入所者エージェントの移動能力を考慮した優先順位決定

対象施設内の入所者の移動能力は多様である．そのため，入所者の移動能力を考慮した避難誘導方針をシミュレーションモデルに組み込む．図 3.2.10 に，入所者の移動能力を考慮しない決定フローを示す．図 3.2.11 に，移動能力が低い入所者を優先的に避難誘導するための決定フローを示す．

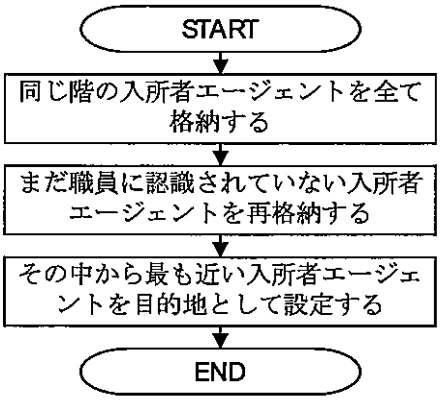


図 3.2.10　属性を考慮しない誘導入所者エージェント決定フロー

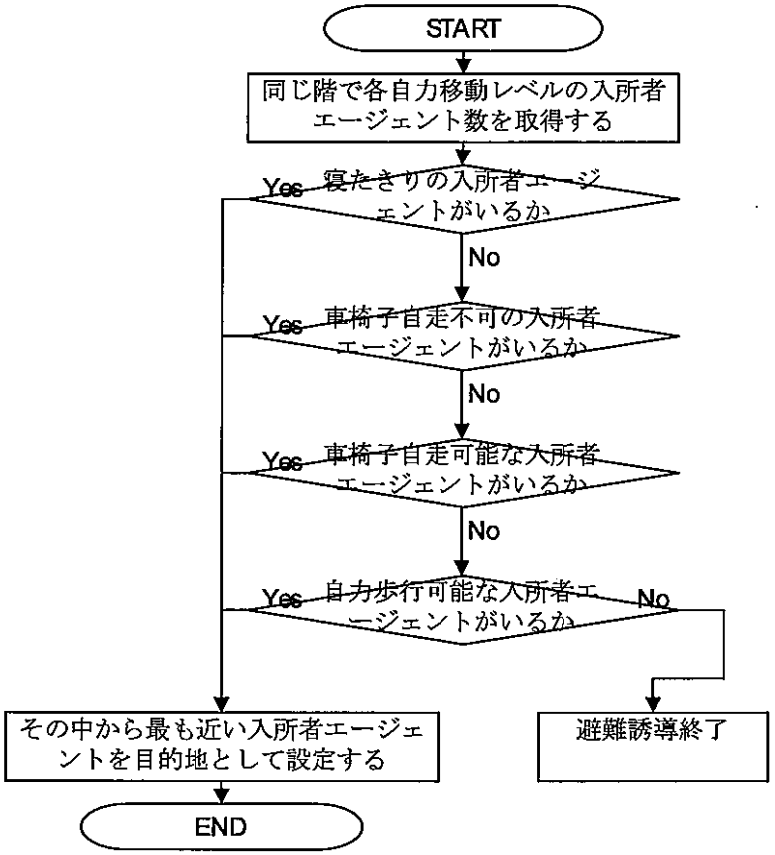


図 3.2.11　属性を考慮した誘導入所者エージェント決定フロー

b) リレーによる2階から1階への受け渡し（日中シナリオ）

対象施設の特徴として，2階建てであることが挙げられる．そこで，2階から1階に避難誘導する際に，上下階で連携したリレーによる避難誘導をシミュレーションモデルに組み込む．図3.2.12にリレーを行わない避難誘導を，図3.2.13にリレーを行う際の避難誘導のフローを示す．

なお，リレーによる避難誘導は，職員数が充分でないと効果的ではないと考え，日中シナリオでの試行のみ行う．

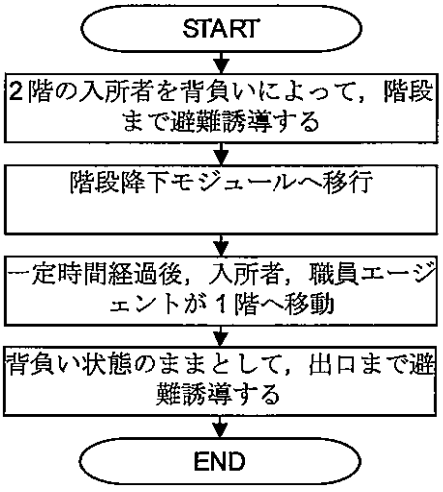


図3.2.12　リレー未実行時誘導フロー

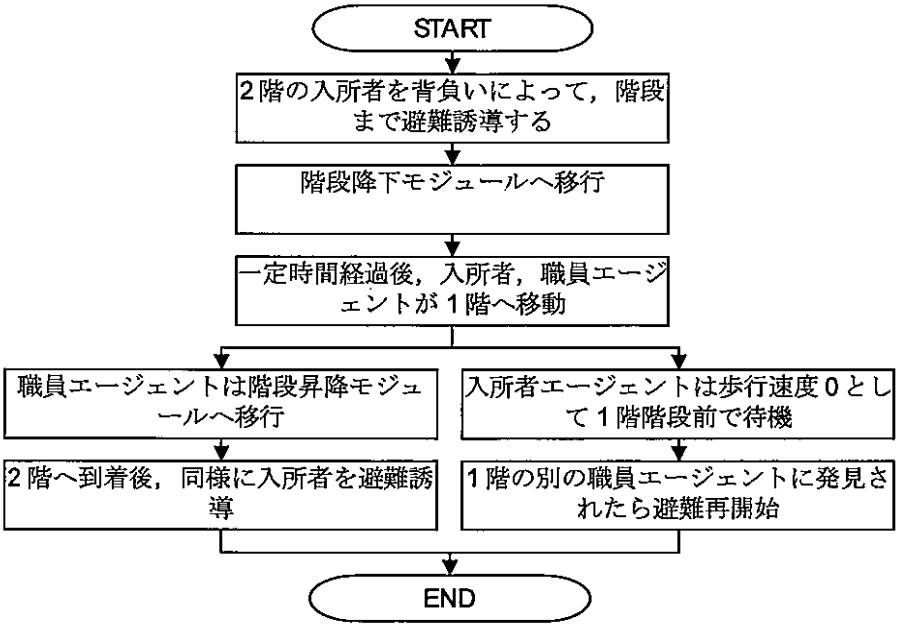


図3.2.13　リレーによる誘導フロー

c) 階毎の避難誘導方針（夜間シナリオ）

1階，2階のどちらを先に避難誘導するかをシミュレーションモデルで設定する．それぞれの階で，職員エージェントに避難誘導されていない，あるいは職員エージェントに認識されていない入所者エージェントが0になるまで，その階を優先した避難誘導を継続する．

なお，階毎の優先順位を考慮した避難誘導は，職員数が少ない場合で無いと効果的ではないと考え，夜間シナリオでの試行のみ行う．

・1階から先に避難誘導を行う

・2階から先に避難誘導を行う

(4) 行動規範

避難誘導方針に従って避難誘導に向かう入所者エージェントを決定すると，その入所者エージェントと自エージェントそれぞれについて最も近いノードエージェントが定まる．2つのノードエージェントについてダイクストラ法によって最短経路を算出し，導出した通過順序に従って入所者エージェントに接近するものとする．

入所者エージェントに到着後は，避難準備作業を行う．その後，入所者エージェントとともに出口まで床エージェントのポテンシャルを参照し，外部空間まで到達する．以上を本モデルにおける避難誘導とする．

なお，ネットワークを用いた移動モジュールについては 3.3.2 で，階段移動については 3.3.3 で後述する．

(5) 避難準備作業

入所者エージェントが「ベッド→背負い」などと移動手段が変化する際に，入所者を背負い上げるなどの準備作業が必要である．それらにかかる時間を表 3.2.14 のように設定する．表 3.2.14 に記した所要時間を経過すると，入所者エージェントの歩行速度が表 3.2.9 に再設定されることで避難準備作業完了を表現している．なお，避難準備作業中は職員エージェントの歩行速度を 0 としている．

表 3.2.14　避難準備作業 [8)]

作業内容	所要時間	その後の移動手段
ベッドのロック解除	5.7秒	ベッド
車椅子手押し	0秒	車椅子介助付移動
ベッド・車椅子・自力歩行→背負い	16.1秒	背負い

3.3 避難誘導シミュレーションプログラムの構成

本節では 3.2 で作成した各エージェントの，避難誘導シミュレーションにおける行動を，モジュール毎に詳細に説明する．

入所者エージェントや職員エージェントはそれぞれの移動モジュールを基本として行動をし，階段に到着した場合や，避難準備作業を行う場合に，階段移動モジュールや避難準備作業モジュールでの計算に移行するプログラム構成となっている．

3.3.1 ポテンシャルを用いた入所者エージェント移動モジュール

入所者エージェントが避難をする際には，床エージェントのポテンシャルを参照して出口への最短経路を辿り移動する．このポテンシャルを参照することで，対象施設の任意の地点からの出口への最短経路が一意に決定される．

なお，エージェントと他エージェントとの距離を計算し，一定距離以下の場合にポテンシャルが高い位置にいるエージェントの速度を 0 にする（＝遠いエージェントが立ち止まる）ことで，人が重ならないことを表現した．判断基準としてのエージェント同士の距離は 1m 以内に近づいたかどうかと設定している．

入所者計算プロセスのフローを図 3.3.1 に示す．

移動中に階段に到着した際の移動方法は 3.3.3 で述べる．

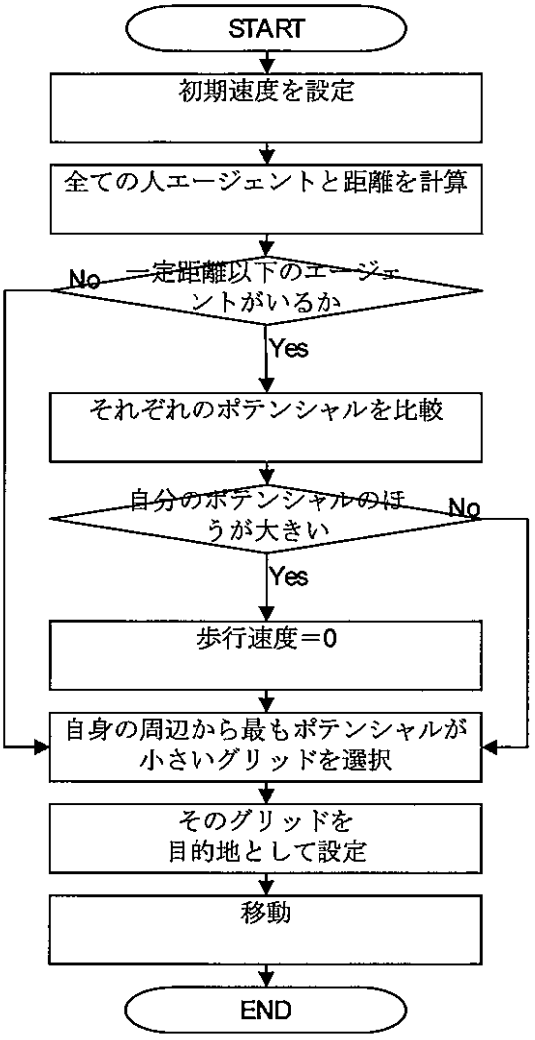


図 3.3.1　入所者エージェント移動モジュールフロー

3.3.2　ネットワークを用いた職員エージェント移動モジュール

職員エージェントが施設内を移動する際の計算方法について説明する．

試行毎に設定した避難誘導方針に従い，誘導する入所者を設定する．ただし，施設全体において「入所者エージェント＜職員エージェント」となれば，それ以上の避難誘導は行わずに職員エージェントは出口へ避難する．

次に，誘導する入所者エージェントと自エージェントについて，最も近いノードエージェントをそれぞれ求める．そのノードエージェント 2 点についてダイクストラ法 [13]で最短経路を計算する．このとき，誘導する入所者エージェントと自エージェントについて最も近いノードエージェントが一致している場合，同じ居室にいると判断されるため，目的地はノードエージェントではなく誘導する入所者エージェントに再設定される．

以上の手順で目的となるノードエージェントを設定し，そのノードエージェントの方向に移動するモジュールが職員エージェント移動モジュールである．

計算プロセスのフローを図 3.3.2 に示す．

移動中に階段に到着した際の移動方法は，階段移動モジュールとして 3.3.3 で述べる．

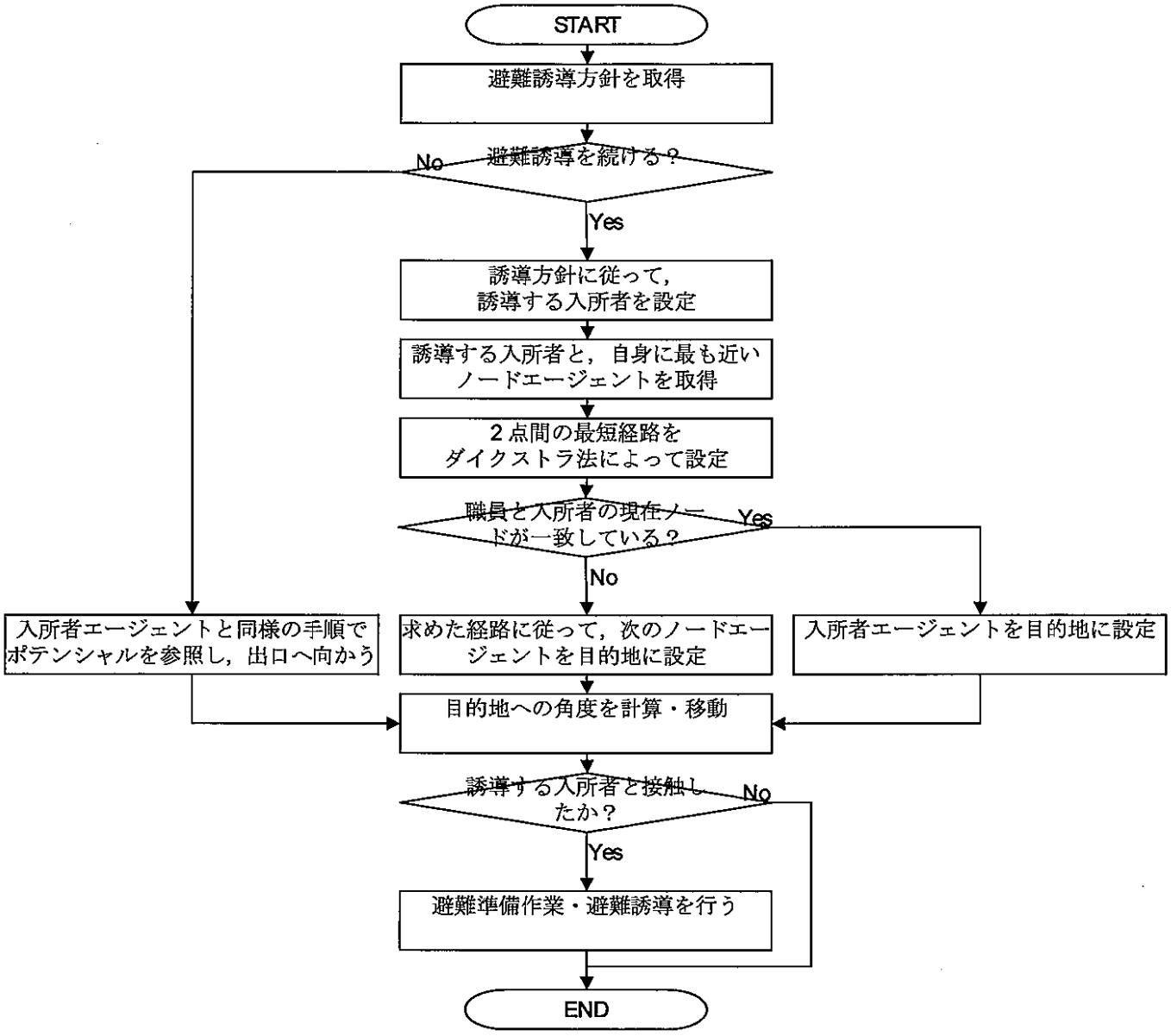


図 3.3.2　職員エージェント移動モジュールフロー

3.3.3　階段移動モジュール

入所者エージェント，職員エージェントについて，階段に到着した際の行動フローについて説明する．入所者エージェントは 2 階から 1 階に降りるケースしかないが，職員エージェントは 1 階から 2 階に上るケースがあるため，それぞれについて区分して説明する．

(1) 2 階から 1 階へ降りる際の処理

エージェントが階段に到着すると，階段移動モジュールでのプログラム処理に移行する．階段内の状況に応じて，すぐに階段の降下を開始するか，待機するかの判断を行う．入所者エージェントが自力で階段を降りることが可能かどうかは，試行毎に設定する．自力で降りない場合は，全て職員エージェントによる背負いで階段を降りるように設定している．

なお，判断には，職員エージェントが入所者エージェントを背負って降りている場合は「1 人」としている．

図 3.3.3 を用いてケースごとに説明をする．AGENT は入所者・職員を問わない人エージェントを表す．なお，避難誘導シミュレーション実験では，既往研究[8]を参考にした上で，背負いの歩行速度やすれ違い時の速度低下を考慮し，2 階から 1 階に降りる際にかかる時間 t を 20 秒としている．入所者単独での階段降下にかかる時間 t も同様に 20 秒と設定した．

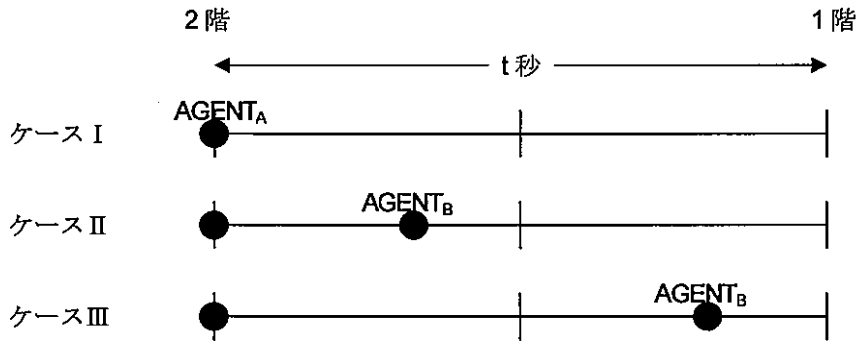


図 3.3.3　階段降下モデル模式図

a)　ケースⅠ：階段内に誰もいない場合

階段に到着し次第，階段移動モードに移行し，階段移動モジュールによる計算を行う．その後，AGENT$_A$が階段を通過した時刻から t 秒後に，AGENT$_A$ として入所者エージェントが 1 階に生成される．

b)　ケースⅡ：先のエージェントが通過してから t／2 秒より早く到着した場合

AGENT$_B$ が階段を通過した時刻から t／2 秒後に，階段降下モードに移行する．その後，AGENT$_A$が階段を通過した時刻から t 秒後に 1 階に生成される．

c) ケースⅢ：先のエージェントが通過してからt／2秒より後に到着した場合

ケースⅠと同様の処理を行う．

(2) 1階から2階へ上る際の処理

1階から2階へ上るのは職員エージェントのみである．職員エージェントが1階から2階へ上る際に，すぐに階段を上るか待機するかの判断には，上っている職員エージェントの数を指標とする．従って，降りている人エージェントの数や状況は，階段を上る際の判断には用いない．

以上より，ケースとしては前述と同様の 3 ケースが考えられ，同様の判断フローによって階段移動を行う．

図 3.3.4 を用いてケースごとに説明をする．なお，避難誘導シミュレーション実験では，既往研究[8)]を参考にした上で，すれ違い時の速度低下を考慮し，1階から2階に上る際にかかる時間tを10秒としている．

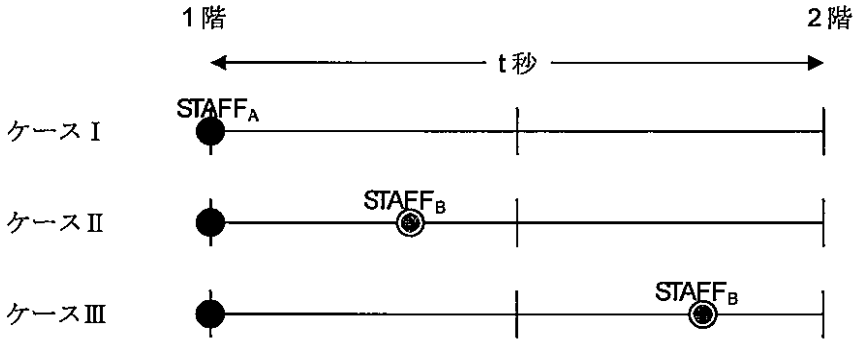


図 3.3.4　階段上昇モデル模式図

a) ケースⅠ：階段内に誰もいない場合

階段に到着し次第，階段移動モードに移行し，階段移動モジュールによる計算を行う．その後，STAFF$_A$が階段を通過した時刻からt秒後に，STAFF$_A$として入所者エージェントが2階に生成される．

b) ケースⅡ：先のエージェントが通過してからt／2秒より早く到着した場合

STAFF$_B$が階段を通過した時刻からt／2秒後に，階段降下モードに移行する．その後，STAFF$_A$が階段を通過した時刻からt秒後に2階に生成される．

c) ケースⅢ：先のエージェントが通過してからt／2秒より後に到着した場合

ケースⅠと同様の処理を行う．

3.3.4　火災発生モジュール

「認知症高齢者グループホーム等の社会福祉施設における防火安全対策のための消防法施行令等の一部改正」（平成19年6月13日，総務省消防庁）[14]を参考に，居室毎の避難限界時間を定めることによって火災及び煙拡散の影響をシミュレーション上で表現する．

(1) 参考文献[14]より「避難限界時間」について

「避難限界時間」は火災により各居室や避難経路が危険な状況となるまでの時間であり，「基準時間」と「延長時間」の和により算定するものとされている．

＜基準時間＞

基準時間は火災室が盛期火災に至る算定上の時間である．起点は自動火災報知設備の作動時を想定している．

表 3.3.5　基準時間算定表[14]

算定項目			基準時間
共通			2分
加算条件	壁及び天井の室内に面する部分の仕上げ	不燃材料	3分
		準不燃材料	2分
		難燃材料	1分
	寝具・布張り家具の防炎性能の確保		1分
	初期消火（屋内消火栓設備によるもの）		1分

＜延長時間＞

延長時間は，盛期火災に至った火災室からの煙・熱の影響によって，他の居室や避難経路が危険な状況となるまでの算定上の時間である．

表 3.3.6　延長時間算定表[14]

算定項目		延長時間
火災室からの区画の形成	防火区画	3分
	不燃化区画	2分
	上記以外の区画	1分
当該室等の床面積×（床面から天井までの高さ－1.8m）≧200m^3		1分

(2) 避難不能時刻の設定

表 3.3.5, 表 3.3.6 を元に避難不能時刻を設定した．厨房を出火点とし，避難開始時刻から煙に巻き込まれて避難不能に至るまでの時間を算出したものを図 3.3.7 に示す．避難開始時刻は，火災報知器が作動した時刻を想定し，避難不能時刻は煙に巻き込まれた時刻として設定した．

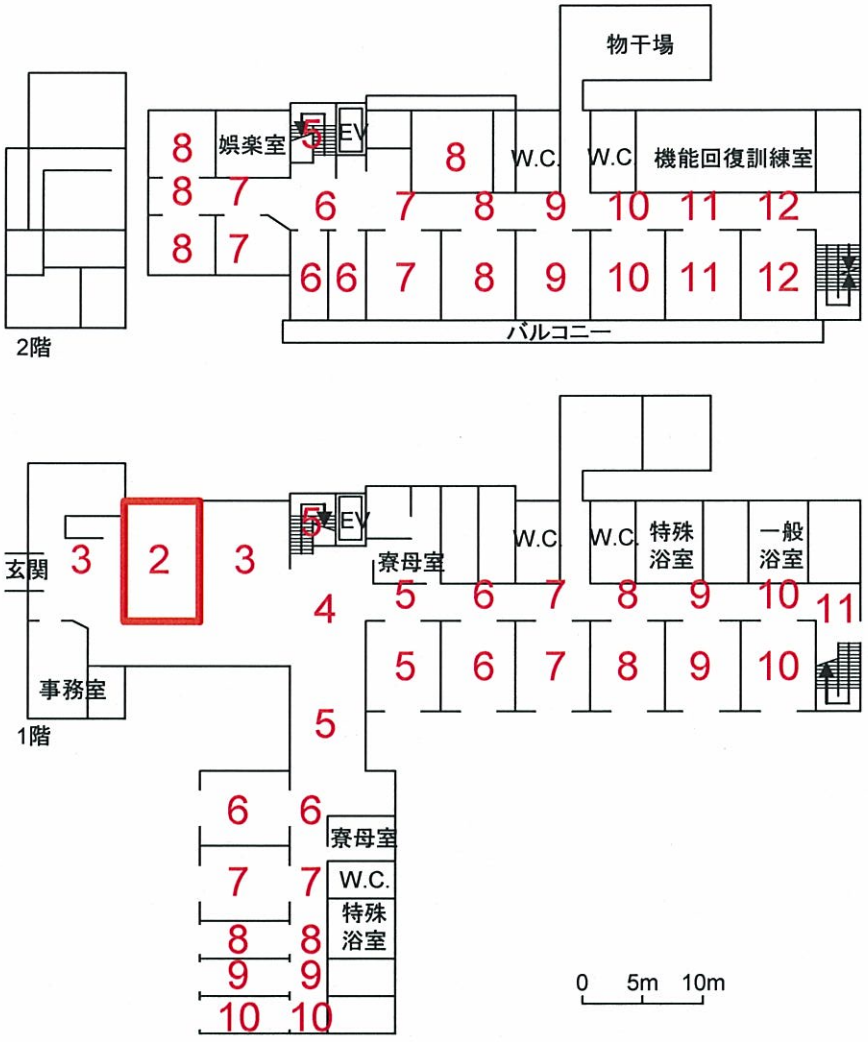


図 3.3.7 避難不能時刻設定（単位：分）

3.4 避難誘導シミュレーションの出力

以上のモデルを試行し，1回の試行が終わると以下の数値データをcsvデータとして出力するものとする．また，シミュレーション実行画面のスクリーンショットを図3.4.1に示す．

(1) 試行終了時

・入所者エージェント避難完了時刻
・入所者エージェント2階階段前待機総時間
・入所者エージェント施設内人数積算
・職員エージェント移動距離総和
・入所者エージェント避難不能人数
・職員エージェント避難不能人数

(2) ステップ毎

・入所者エージェント1階人数
・入所者エージェント2階人数
・入所者エージェント総人数
・職員エージェント1階人数
・職員エージェント2階人数
・職員エージェント総人数

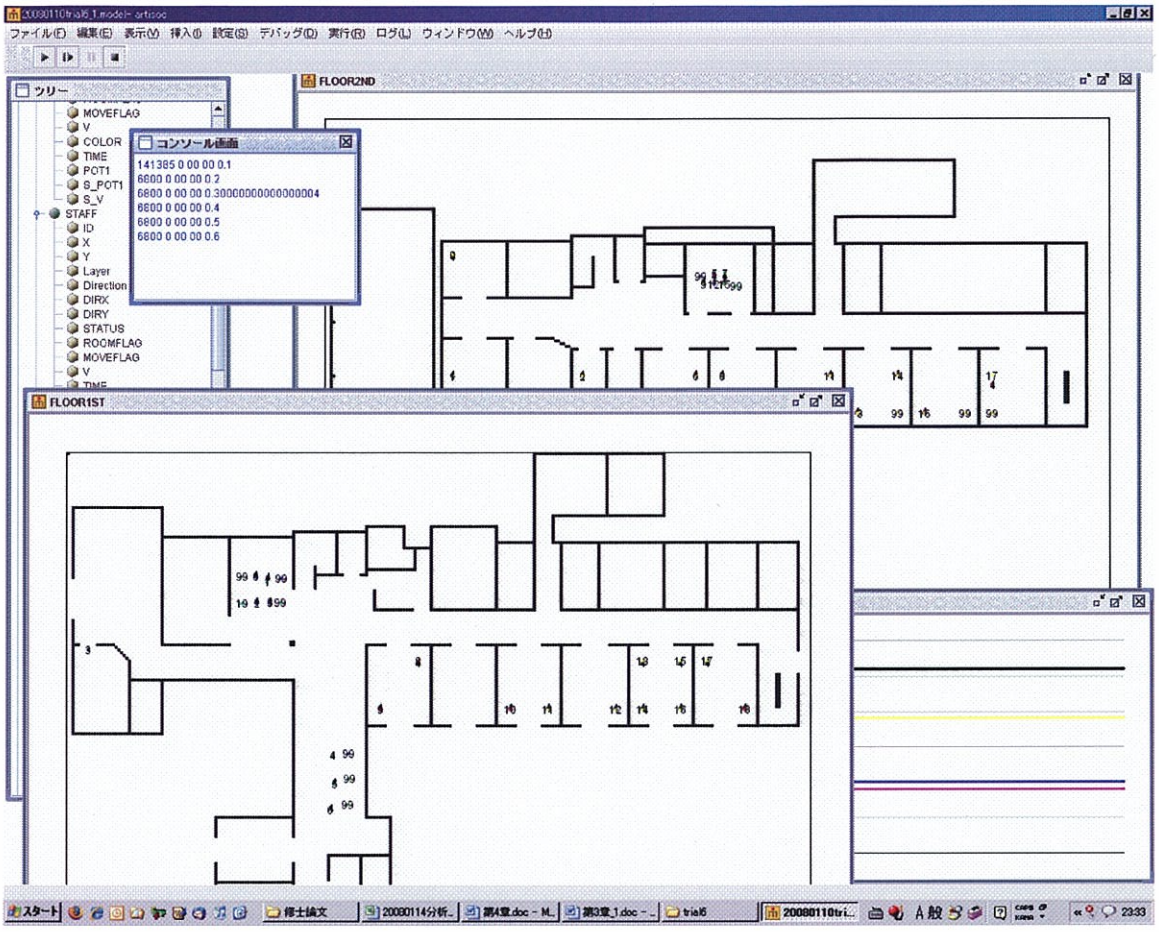


図 3.4.1　避難誘導シミュレーションスクリーンショット

第4章　避難誘導シミュレーション実験の結果と分析

4.1 シミュレーションを用いた比較分析の概要

本章では，第 3 章で構築した避難誘導シミュレーションを用いて，効果的な避難誘導方法を導出することを目的として，入所者の避難誘導方針の設定や初期配置の設定を変化させ，各試行間の比較分析を行う．

4.1.1 分析計画

本研究の目的として掲げている，効果的な避難誘導方法の導出を達成するために，対象施設のような介護保険施設で実現可能と考えられる避難誘導方法をシミュレーションで表現し，その効果の比較分析を行う．

入所者エージェント及び職員エージェントの初期配置は，前述の日中・夜間シナリオに則りそれぞれ設定する．また，夜間シナリオにおいて，初期配置を自力移動レベルに応じて再設定した試行を行い，その効果を分析する．さらに，夜間シナリオについては，厨房での火災を想定したモジュールを組み込み，避難不能人数の算定を行う．

避難誘導方針の比較については，入所者エージェントの移動能力での優先順位付けの効果を把握する．具体的には，寝たきりの入所者エージェントを優先的に避難誘導する試行や，移動能力を問わずに職員エージェントに近い入所者エージェントから避難誘導する試行の比較を行う．

避難誘導戦術として，日中シナリオでは 2 階から 1 階へのリレーをシミュレーションに組み込み，その効果を分析する．夜間シナリオでは，1 階，または 2 階のどちらの階の入所者エージェントを先に全て避難誘導するかを，それぞれ試行を行い比較分析をする．

以上より，表 4.1.1 に本章で行うシミュレーション実験の試行番号とその試行に取り入れる要素をまとめる．試行番号について，D は日中シナリオ（Daytime），N は夜間シナリオ（Night），F は火災発生シナリオ（Fire）を表している．なお，試行回数は各 1 回，単位時間 0.1 秒として行った．

表 4.1.1　比較分析整理表

試行番号	初期配置				火災の影響を考慮		避難誘導方針策定			避難誘導戦術の考慮				入所者運動能力	
	日中シナリオ	夜間シナリオ			火災の影響		移動能力での優先順位		距離による優先順位	階毎の避難誘導方針		2階から1階へのリレー		階段単独降下	
	通常	通常	移動レベルが低いエージェントを1階へ	移動レベルが低いエージェントを2階へ	考慮しない	考慮する	考慮しない	自力移動レベルが低い	職員に近い	1階から先に誘導	2階から先に誘導	行わない	行う	不可能	可能
D1_1	○	○			○		○		○			○		○	
D1_2	○	○			○		○		○				○	○	
D1_3	○	○			○			○	○			○		○	
D1_4	○	○			○			○	○				○	○	
N1_1		○			○		○		○	○		○			○
N1_2		○			○		○		○		○	○			○
N1_3		○			○			○	○	○		○			○
N1_4		○			○			○	○		○	○			○
N2_1			○		○		○		○	○		○			○
N2_2			○		○		○		○		○	○			○
N2_3			○		○			○	○	○		○			○
N2_4			○		○			○	○		○	○			○
N3_1				○	○		○		○	○		○			(○)
N3_2				○	○		○		○		○	○			(○)
N3_3				○	○			○	○	○		○			(○)
N3_4				○	○			○	○		○	○			(○)
F1_1		○				○	○		○	○		○			○
F1_2		○				○	○		○		○	○			○
F1_3		○				○		○	○	○		○			○
F1_4		○				○		○	○		○	○			○

4.1.2 比較指標

前述 3.4 のシミュレーションモデルの出力として得られた数値を用いて，以下を比較指標として各試行結果の比較分析を行う．

(1) 避難完了時刻

全ての入所者エージェント及び職員エージェントが，屋外に退出した時刻を避難完了時刻とする．

(2) 平均避難時間

時間経過ごとに施設内の入所者エージェントの数は変化するが，その人数を積算し効率性を判断する比較指標として用いる．避難完了時刻が同値であっても，この値が小さければより早く多くの入所者が避難を終えていると判断できる．この値を全ての入所者で除し，平均避難時間として以下の式 4.1.2 として算出した．

$$T_{ave} = \sum_{t=0}^{T} n(t) \cdot \Delta t \Big/ N = \left(\sum_{t=0}^{T} n(t) \Big/ N \right) \cdot \Delta t \qquad [4.1.2]$$

T_{ave}：平均避難時間（秒）

T：避難完了時間（秒）

n(t)：時刻 t における施設内入所者人数（人）

N：入所者総人数（人）

(3) 職員エージェント総移動距離

職員エージェントが移動した距離の総和を，以下の式 4.1.3 として算出した．

$$D = \sum_{t=0}^{T} \sum_{i=1}^{M} v_i(t) \cdot \Delta t \qquad [4.1.3]$$

D：職員総移動距離（m）

i：職員 ID

M：職員総人数（人）

$v_i(t)$：時刻 t における職員 i の歩行速度（m／秒）

(4) 入所者エージェント階段前総待機時間

2 階から 1 階への階段の前で階段待ちの滞留が生じるが，入所者エージェントが階段待ちのために歩行を停止している時間の総和を，入所者エージェント総待機時間として求めた．

(5) 入所者エージェント施設内人数

ステップ毎の施設内の人数を 1 階，2 階，1 階＋2 階をそれぞれ記録した．この人数変化を用いて避難誘導方法の特徴の比較を行う．なお，施設内人数を積分した値は，前述(2)の入所者エージェント施設内人数を積算したものと同値である．

(6) 避難不能人数

火災発生ケースにおいて，避難不能となった人数を，入所者エージェント，職員エージェントそれぞれを比較に用いる．

(7) ジニ係数

ジニ係数は，経済分野で頻繁に用いられる，不平等性を表す指標である．

本研究での用い方としては，まず，横軸を累積避難人数，縦軸を累積避難時間とし，避難時間の短い入所者から並べることでローレンツ曲線を作成する．その後，均等分布線とローレンツ曲線で囲まれる面積を求め，均等分布線と軸が描く三角形の面積を 1 とした面積比をジニ係数として用いている．

4.2　入所者移動能力の考慮とリレー方式による避難誘導の影響分析

日中シナリオの下で，避難誘導方針の比較分析を行う．ここでの避難誘導方針として，職員エージェントが入所者エージェントの移動能力を考慮するか，つまり自力移動レベルを参照するかしないかの比較，及び2階から1階に入所者エージェントを誘導する際にリレーを行うかの比較を行う．

4.2.1　試行内容

本節で行った試行内容を表 4.2.1 に示す．移動能力での優先順位の有無及び2階から1階へのリレーの有無について2×2通りの試行による比較を行う．

表 4.2.1　入所者属性×リレー方式 試行内容

試行番号	初期配置				火災の影響を考慮		避難誘導方針策定			避難誘導戦術の考慮				入所者運動能力	
	日中シナリオ	夜間シナリオ			火災の影響		移動能力での優先順位		距離による優先順位	階毎の避難誘導方針		2階から1階へのリレー		階段単独降下	
	通常	通常	移動レベルが低いエージェントを1階へ	移動レベルが低いエージェントを2階へ	考慮しない	考慮する	考慮しない	自力移動レベルが低い	職員に近い	1階から先に誘導	2階から先に誘導	行わない	行う	不可能	可能
D1_1	○	○			○		○		○			○		○	
D1_2	○	○			○		○		○				○	○	
D1_3	○	○			○			○	○			○		○	
D1_4	○	○			○			○	○				○	○	

4.2.2　実験結果

それぞれの避難誘導方針毎の結果を表 4.2.2 に，比較のための図を図 4.2.3，図 4.2.4 に示す．また，図 4.2.5，図 4.2.6 に施行毎の施設内入所者エージェント数の時間経過を記す．

各試行について，避難完了時刻や施設内総人数の時間経過については大きな差は見られなかった．その中で，避難誘導戦術のリレーについて，リレーを行うことで避難完了時刻や入所者の 2 階階段付近待機時間は減少しており，一定の効果があると言える．

4.2.3　分析・考察

避難完了時刻については大きな差が見られなかったのは，施設内の職員数が充分であるためと考えられる．特に 1 階居室については全ての入所者と職員が 1 対 1 で対応している状態から避難誘導を開始するため，避難開始 30 秒前後まで施設内人数の時間経過が同様の数値変化を示しており，このことによって試行毎に差が生じなかったと考えられる．

リレーを行うことがそれほど効果的な結果とならなかったのは，リレーを行うことによって 1 階階

段付近で，改めて別の職員エージェントによる避難準備作業（＝背負い）が必要になり，避難準備作業の回数が増加することが原因であると考えられる．

入所者に移動能力を考慮しての誘導方針が効率的である結果となったのは，理にかなった結果であり，自力移動できる入所者には自力で移動してもらい，職員のマンパワーは寝たきり等で動けない入所者に向けるべきであることを示唆している．

表 4.2.2　移動能力考慮×リレー方式 実験結果

移動能力考慮×リレー		リレーしない	リレーする
入所者の移動能力を考慮しない	避難完了時刻(秒)	259.2	239.2
	職員総移動距離(m)	14374.7	14310.1
	入所者 2 階階段待機時間(秒)	2087.6	1845.1
	平均避難時間	77.7	82.3
	平均入所者数(人／ステップ)	15.6	17.9
自力移動レベルが低い入所者を優先	避難完了時刻(秒)	259.3	239.1
	職員総移動距離(m)	13826.9	14540.1
	入所者 2 階階段待機時間(秒)	2103.9	1860.5
	平均避難時間	75.9	80.7
	平均入所者数(人／ステップ)	15.2	17.6

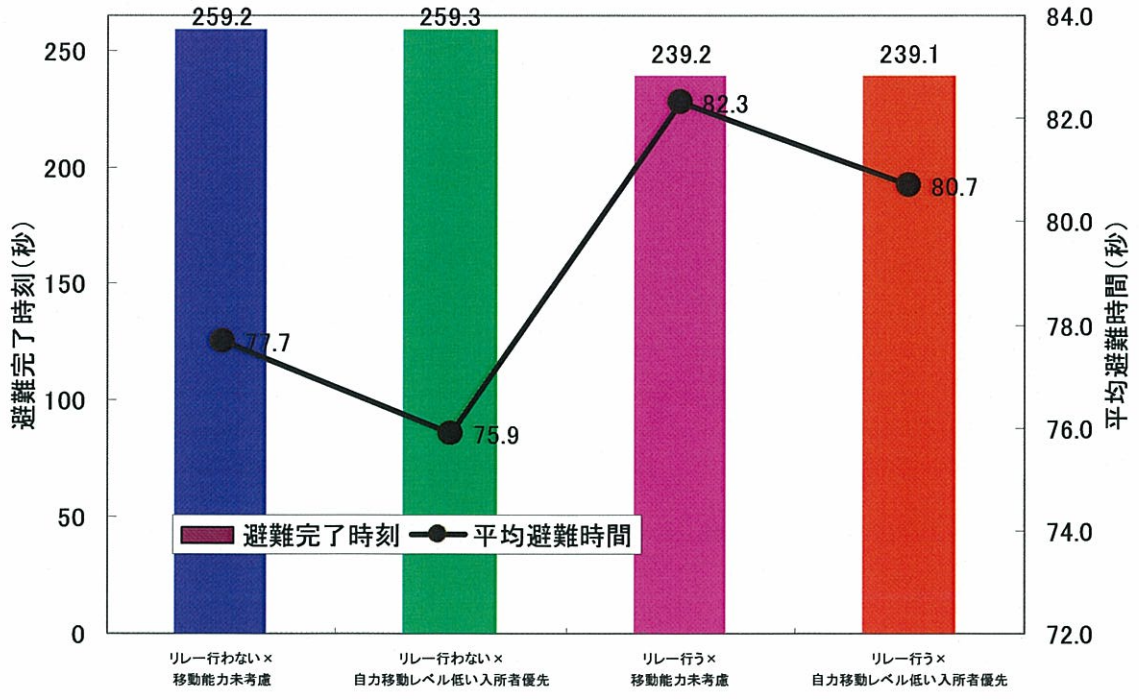


図 4.2.3　移動能力考慮×リレー方式 避難完了時刻・平均避難時間比較

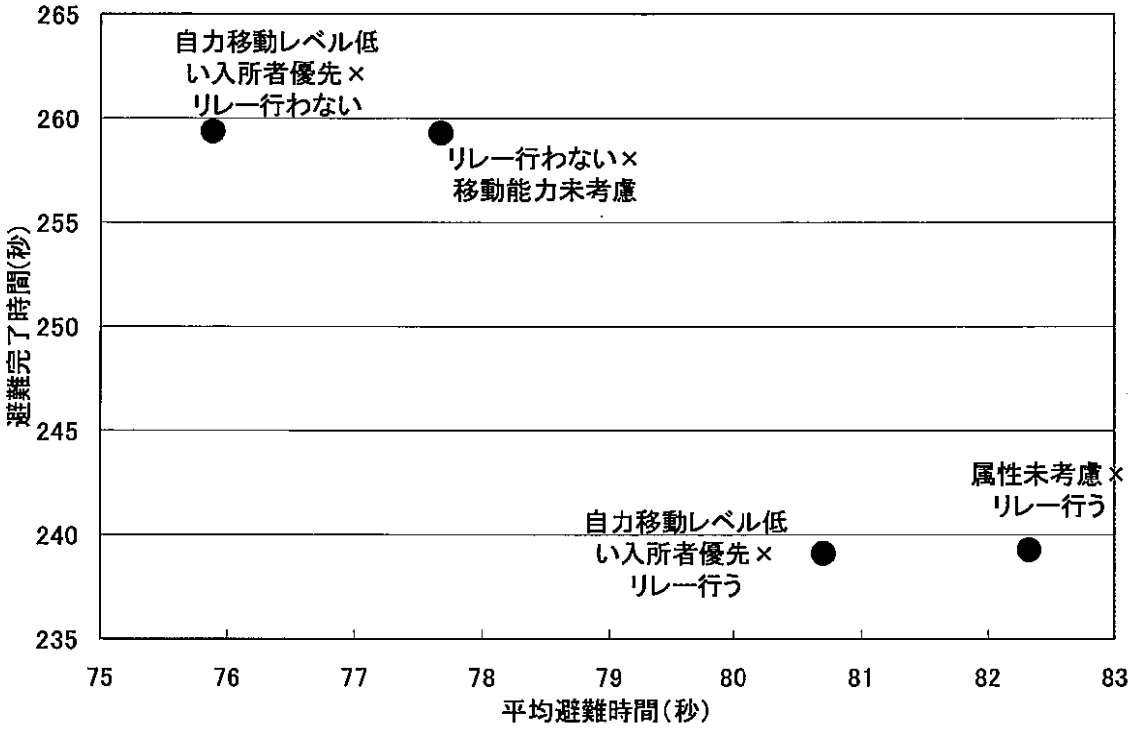


図 4.2.4　移動能力考慮×リレー方式 避難完了時刻・平均避難時間プロット

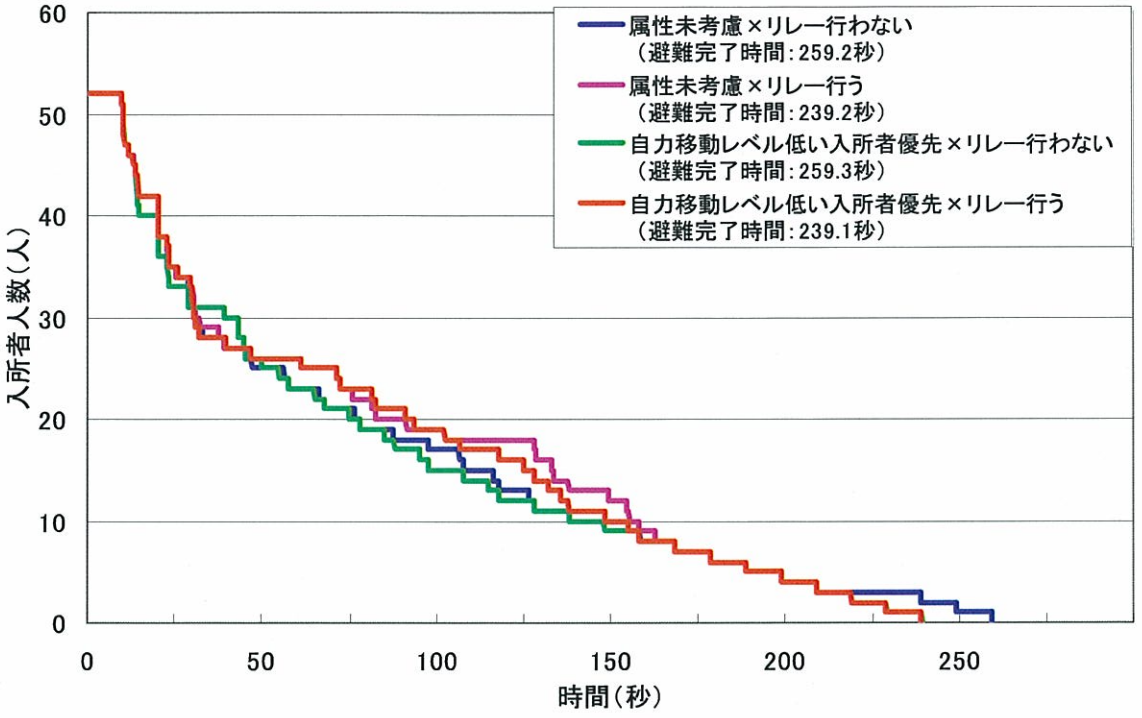


図 4.2.5　移動能力考慮×リレー方式 入所者人数時間経過

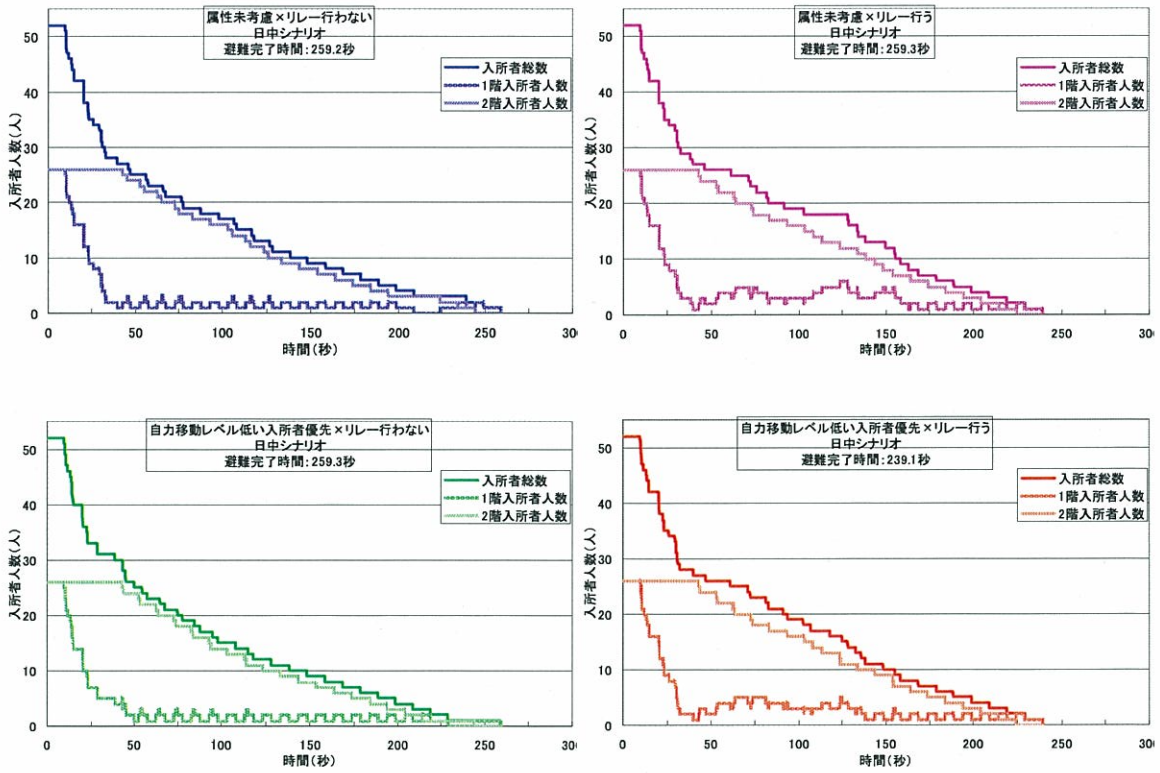


図 4.2.6　移動能力考慮×リレー方式 入所者人数時間経過（試行毎）

4.3 夜間シナリオにおける避難誘導方法の比較分析の概要

夜間シナリオの下で，まず，通常配置での試行をベースケースとして，避難誘導方針の比較分析を行う．次に，初期配置を変更し，その効果の分析を行う．その後，ベースケースに火災発生モジュールを実装し，火災発生ケースとして分析を行う．

夜間シナリオとしての試行内容を表 4.3.1 に示す．なお，同色の試行は同じアルゴリズムを使用している（ただし，火災発生を組み込んだ試行については，その部分を加筆している）．

表 4.3.1　夜間シナリオ 試行内容

試行番号	初期配置				火災の影響を考慮		避難誘導方針策定			避難誘導戦術の考慮				入所者運動能力	
	日中シナリオ	夜間シナリオ			火災の影響		移動能力での優先順位		距離による優先順位	階毎の避難誘導方針		2階から1階へのリレー		階段単独降下	
	通常	通常	移動レベルが低いエージェントを1階へ	移動レベルが低いエージェントを2階へ	考慮しない	考慮する	考慮しない	自力移動レベルが低い	職員に近い	1階から先に誘導	2階から先に誘導	行わない	行う	不可能	可能
N1_1		○			○		○		○	○		○			○
N1_2		○			○		○		○		○	○			○
N1_3		○			○			○	○	○		○			○
N1_4		○			○			○	○		○	○			○
N2_1			○		○		○		○	○		○			○
N2_2			○		○		○		○		○	○			○
N2_3			○		○			○	○	○		○			○
N2_4			○		○			○	○		○	○			○
N3_1				○	○		○		○	○		○			(○)
N3_2				○	○		○		○		○	○			(○)
N3_3				○	○			○	○	○		○			(○)
N3_4				○	○			○	○		○	○			(○)
F1_1		○				○	○		○	○		○			○
F1_2		○				○	○		○		○	○			○
F1_3		○				○		○	○	○		○			○
F1_4		○				○		○	○		○	○			○

4.4　入所者移動能力と階別優先順位の考慮による避難誘導の影響分析

夜間シナリオの下で，避難誘導方針の比較分析を行う．ここでの避難誘導方針として，職員エージェントが入所者エージェントの移動能力を参照するかの比較，及び 1 階の入所者から避難誘導を行うか，2 階の入所者から避難誘導を行うかの比較を行う．

4.4.1　試行内容

属性での優先順位の有無及び 1 階の入所者から避難誘導を行うか，2 階の入所者から避難誘導を行うかについて 2×2 通りの試行による比較を行う．表 4.4.1 に試行内容を示す．

表 4.4.1　入所者移動能力×階別優先順位 試行内容

試行番号	初期配置				火災の影響を考慮		避難誘導方針策定			避難誘導戦術の考慮				入所者運動能力	
	日中シナリオ	夜間シナリオ			火災の影響		移動能力での優先順位		距離による優先順位	階毎の避難誘導方針		2階から1階へのリレー		階段単独降下	
	通常	通常	移動レベルが低いエージェントを1階へ	移動レベルが低いエージェントを2階へ	考慮しない	考慮する	考慮しない	自力移動レベルが低い	職員に近い	1階から先に誘導	2階から先に誘導	行わない	行う	不可能	可能
N1_1		○			○		○		○	○		○			○
N1_2		○			○		○		○		○	○			○
N1_3		○			○			○	○	○		○			○
N1_4		○			○			○	○		○	○			○

4.4.2　実験結果

それぞれの避難誘導方針毎の結果を表 4.4.2 に，比較のための図を図 4.4.3，図 4.4.4 に示す．また，図 4.4.5，図 4.4.6 に試行毎の施設内入所者エージェント数の時間経過を記す．

避難完了時刻については，最も早い試行と最も遅い試行の差は約 40 秒となった．図 4.4.5 の入所者人数時間経過に注目すると，1 階を優先するか，2 階を優先するかで，人数変化の形状が類似している．1 階を優先する避難誘導では，150 秒前後までは速いペースで避難誘導を進めているが，その後は 2 階との往復が繰り返されるため，ゆっくりとしたペースとなっている．一方，2 階を優先する避難誘導では，避難開始直後は 1 階の自力歩行可能な入所者のみが避難しているのみで，なかなか避難誘導がはかどらないが，500 秒前後には避難誘導のペースが上がり，避難完了時刻についても 1 階を優先する避難誘導に追いつく結果となった．

平均避難時間については，1 階の入所者を優先する避難誘導が低い値を示している．

4.4.3　分析・考察

1階の入所者を優先しての避難誘導が，入所者人数の積算の値が低くなり効率的である，というのは理にかなった結果と言える．これは，施設全体の避難の効率という観点で言えば，出口に近い入所者から避難誘導することが望ましいことを示唆している．

図 4.4.5 において，傾きが大きい時間帯は，1階の避難誘導に取り組んでいる時間であると考えられる．1階優先と比較して2階優先の傾きが大きいのは，2階の避難誘導を予め完了させ，一定時間経過ごとに到着する職員に，効率よく入所者に接近させることができた結果であると考えられる．その結果として，避難完了時刻に関しても，2階を優先的に避難誘導する試行が最も早く避難を終えたと考えられる．

図 4.4.4 から読み取れることとして，1 階を優先する際には自力移動できない入所者を優先的に，2階を優先する際には，近くの入所者を優先的にそれぞれ避難することが望ましいと言える．1階の各居室から出口へは非常に近距離であり，一度避難誘導を終えた職員が次に避難誘導する入所者を，自分の近くにいる入所者に決定して避難誘導を行っても，すぐに出口まで到達できることが予想される．そのため，1階を優先して避難誘導する際に自力移動できない入所者を優先的に誘導するのは，あらゆる入所者に対して限られた職員が対応しようとすることはが非効率的であることの裏づけであり理にかなっていると言える．

表 4.4.2　入所者移動能力×階別優先順位 実験結果

入所者移動能力×階別優先順位		1 階優先	2 階優先
入所者の属性を考慮しない	避難完了時刻(秒)	627.6	587.4
	職員総移動距離(m)	8816.2	8888.4
	入所者 2 階階段待機時間(秒)	607.0	597.7
	平均避難時間	228.0	393.1
	平均入所者数(人／ステップ)	18.9	34.8
自力移動レベルが低い入所者を優先	避難完了時刻(秒)	609.1	612.1
	職員総移動距離(m)	8833.2	9634.6
	入所者 2 階階段待機時間(秒)	621.7	629.7
	平均避難時間	239.3	417.9
	平均入所者数(人／ステップ)	20.4	35.5

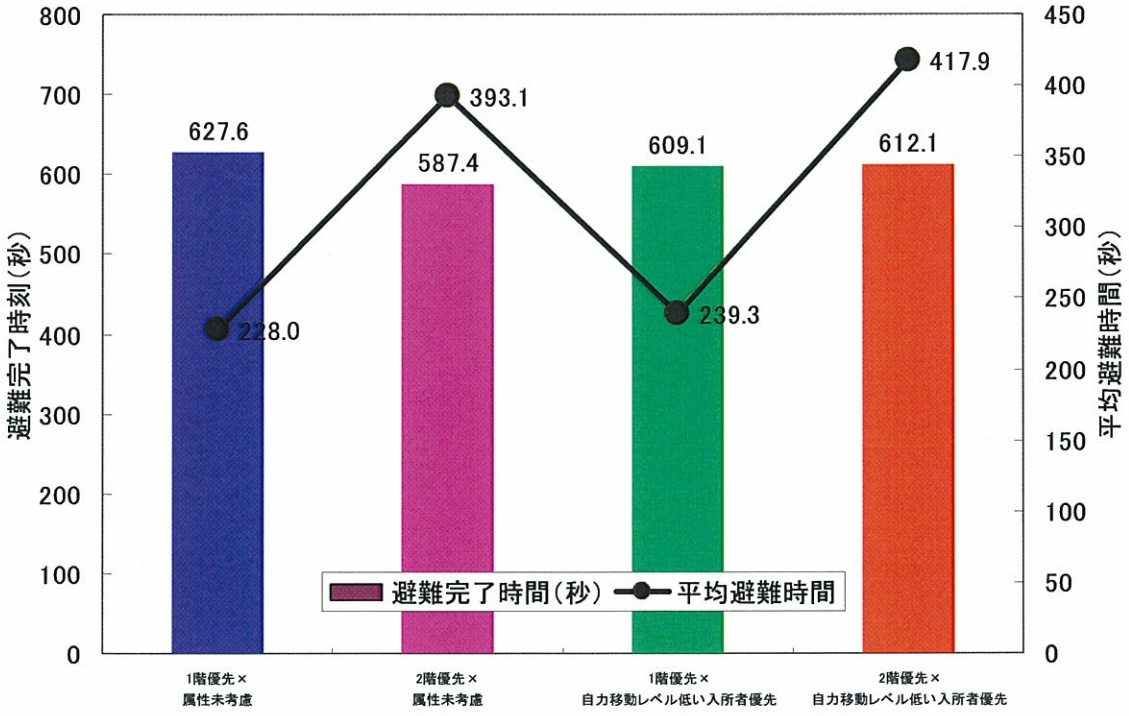


図 4.4.3　入所者移動能力×階別優先順位 避難完了時刻・平均避難時間比較

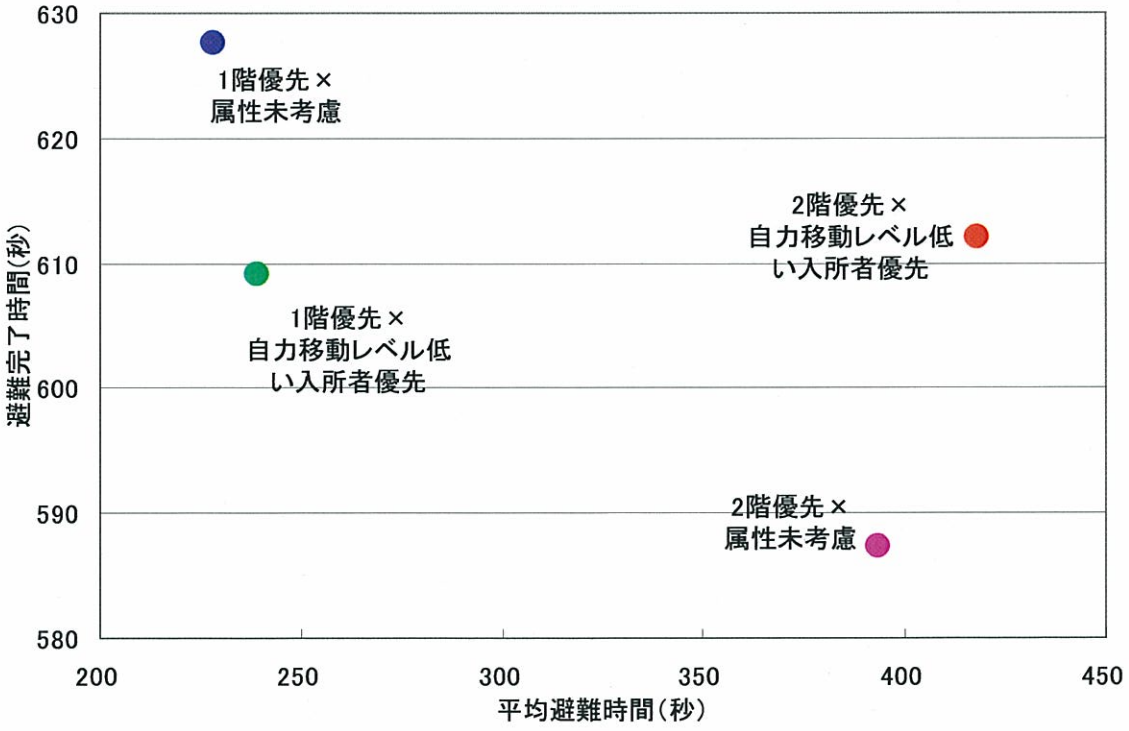


図 4.4.4　入所者移動能力×階別優先順位 避難完了時刻・平均避難時間プロット

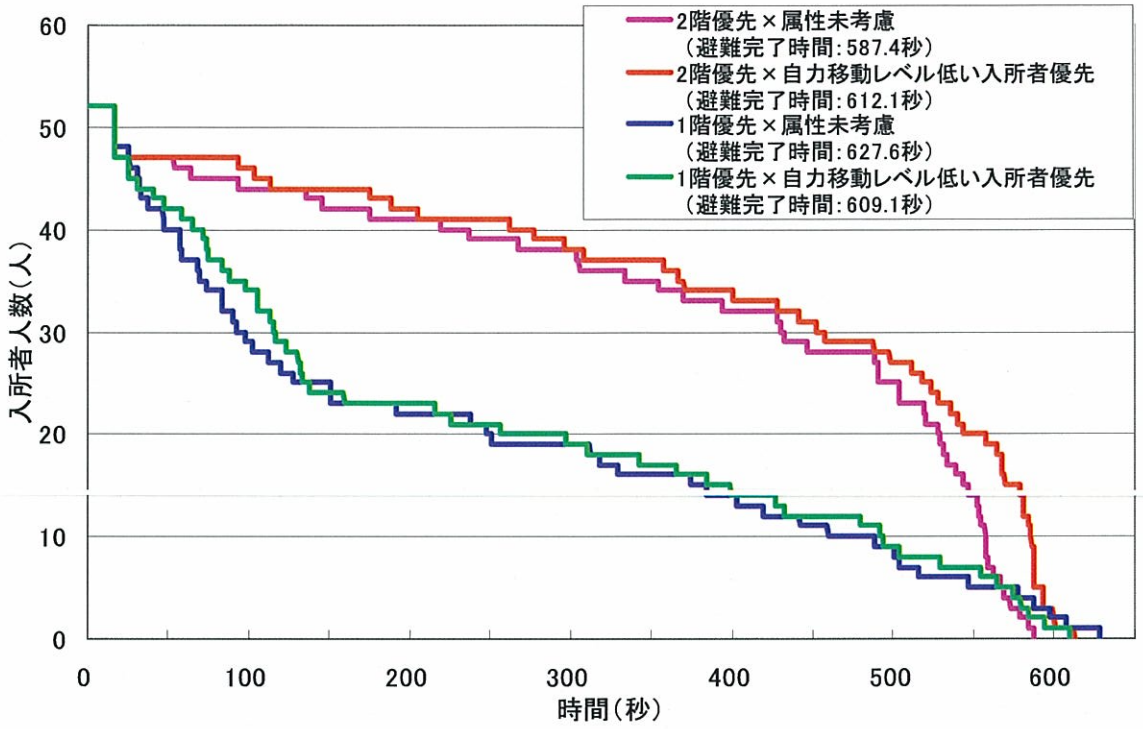


図 4.4.5　入所者移動能力×階別優先順位　入所者人数時間経過

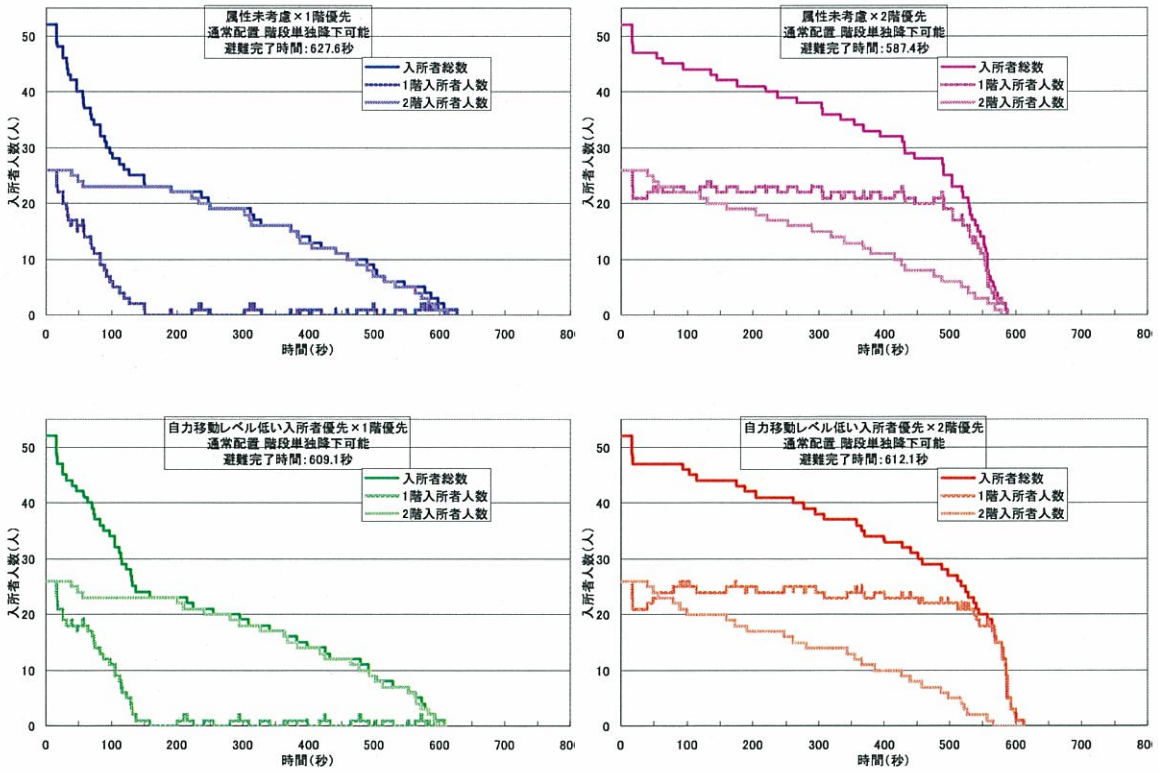


図 4.4.6　入所者移動能力×階別優先順位　入所者人数時間経過（試行毎）

4.5　入所者の初期配置変更による避難誘導への影響の把握

避難開始時の初期配置設定を変更しての試行によって，居室計画による避難の影響について分析を行う．初期配置設定の方針として，入所者の移動能力に注目し，自力移動レベルが低い入所者を 1 階の居室に割り当てるか，2 階の居室に割り当てるかを方針として掲げ，それぞれの避難の比較を行う．

4.5.1　試行内容

初期配置を 4.4 と同様の試行の他に，図 4.5.1 に示す初期配置での試行を行い，比較分析を行う．入所者の移動能力に着目し，2 つの配置ケースを追加する．表 4.5.2.に試行内容を示す．表において同色の試行は同様のアルゴリズムによる試行である．

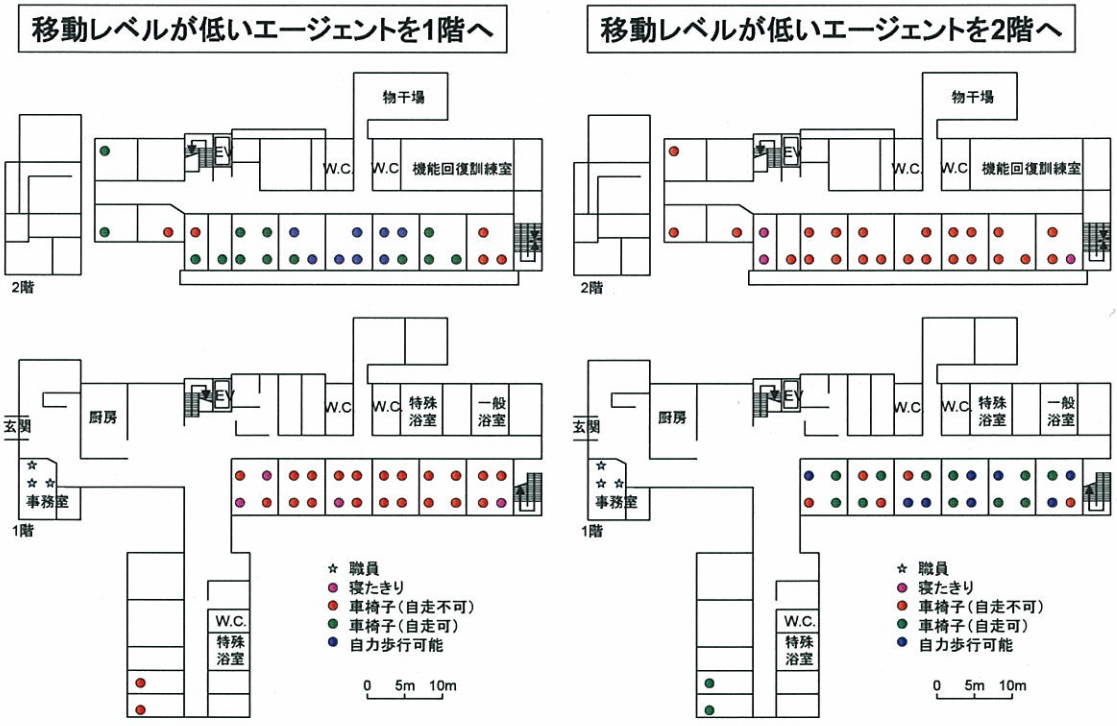


図 4.5.1　初期配置変更ケース設定

表 4.5.2　初期配置変更ケース 試行内容

試行番号	初期配置				火災の影響を考慮		避難誘導方針策定			避難誘導戦術の考慮				入所者運動能力	
	日中シナリオ	夜間シナリオ			火災の影響		移動能力での優先順位		距離による優先順位	階毎の避難誘導方針		2階から1階へのリレー		階段単独降下	
	通常	通常	移動レベルが低いエージェントを1階へ	移動レベルが低いエージェントを2階へ	考慮しない	考慮する	考慮しない	自力移動レベルが低い	職員に近い	1階から先に誘導	2階から先に誘導	行わない	行う	不可能	可能
N1_1		○			○		○		○	○		○			○
N1_2		○			○		○		○		○	○			○
N1_3		○			○			○	○	○		○			○
N1_4		○			○			○	○		○	○			○
N2_1			○		○		○		○	○		○			○
N2_2			○		○		○		○		○	○			○
N2_3			○		○			○	○	○		○			○
N2_4			○		○			○	○		○	○			○
N3_1				○	○		○		○	○		○			(○)
N3_2				○	○		○		○		○	○			(○)
N3_3				○	○			○	○	○		○			(○)
N3_4				○	○			○	○		○	○			(○)

4.5.2　実験結果

それぞれの避難誘導方針毎の結果を表 4.5.3 に，比較のための図を図 4.5.4，図 4.5.5 に示す．また，図 4.2.6 に施行毎の施設内入所者エージェント数の時間経過を記す．

最も早く避難完了した試行と最も時間がかかった試行で約 90 秒の差が生じている．図 4.5.5 から，移動レベルが低い入所者を 1 階に配置している場合は，入所者の移動能力を考慮した避難誘導を行うほうが効果的であることが読み取れる．また，入所者の移動能力を考慮せずに，近い入所者から避難誘導する方法は，他の誘導方針と比較して避難完了時刻が遅くなる傾向があることがわかる．

移動レベルが低い入所者を 2 階に配置することは，入所者の移動能力を考慮した避難誘導を行う試行において，平均避難時間が減少しており，施設全体の避難誘導の効率性が高まる結果となった．

4.5.3　分析・考察

自力移動できる入所者を 1 階に配置することで，早い時間帯で何人かの入所者が避難完了し，効率的である結果が導かれると考えられるが，2 階を優先しての避難誘導では平均避難時間の値は減少し，避難完了時刻は増加している．これは，自力移動できる入所者を 1 階に配置すると，背負いで避難誘導しなくてはならない入所者が 2 階に集中するというトレードオフを示した結果と言える．

1 階を優先して避難誘導をする際には，入所者の居室状況にかかわらず，自力移動できない入所者を優先的に避難誘導することが望ましいことを示している．

しかし，避難完了時刻を比較すると，2 階を優先しての避難誘導のほうが短時間で終了することができる傾向にある．これは，2 階から避難誘導をしておけば，到着した職員が大きく移動することなく，1 階の入所者を発見できることで，最終的に避難完了時刻を短く済ますことができるためだと考えられる．

表 4.5.3　初期配置変更ケース 実験結果

初期配置変更		現行の配置	移動レベルが低い入所者を 1 階に配置	移動レベルが低い入所者を 2 階に配置
試行番号		N1_○	N2_○	N3_○
入所者の属性を考慮しない × 1 階優先	避難完了時刻(秒)	627.6	613.6	651.1
	職員総移動距離(m)	8816.2	8747.7	9220.1
	入所者 2 階階段待機時間(秒)	607.0	662.6	643.0
	平均避難時間	228.0	243.5	244.0
	平均入所者数(人／ステップ)	18.9	20.6	19.5
入所者の属性を考慮しない × 2 階優先	避難完了時刻(秒)	587.4	605.6	604.9
	職員総移動距離(m)	8888.4	9668.6	9103.5
	入所者 2 階階段待機時間(秒)	597.7	685.6	605.4
	平均避難時間	393.1	416.9	371.3
	平均入所者数(人／ステップ))	34.8	35.8	31.9
自力移動レベルが低い入所者を優先 × 1 階優先	避難完了時刻(秒)	609.1	606.2	637.6
	職員総移動距離(m)	8833.2	8452.7	9309.6
	入所者 2 階階段待機時間(秒)	621.7	642.7	577.0
	平均避難時間	239.3	232.0	245.2
	平均入所者数(人／ステップ)	20.4	19.9	20.0
自力移動レベルが低い入所者を優先 × 2 階優先	避難完了時刻(秒)	612.1	566.2	616.7
	職員総移動距離(m)	9634.6	8804.9	8701.6
	入所者 2 階階段待機時間(秒)	629.7	746.1	546.3
	平均避難時間	417.9	368.6	365.0
	平均入所者数(人／ステップ)	35.5	33.9	30.8

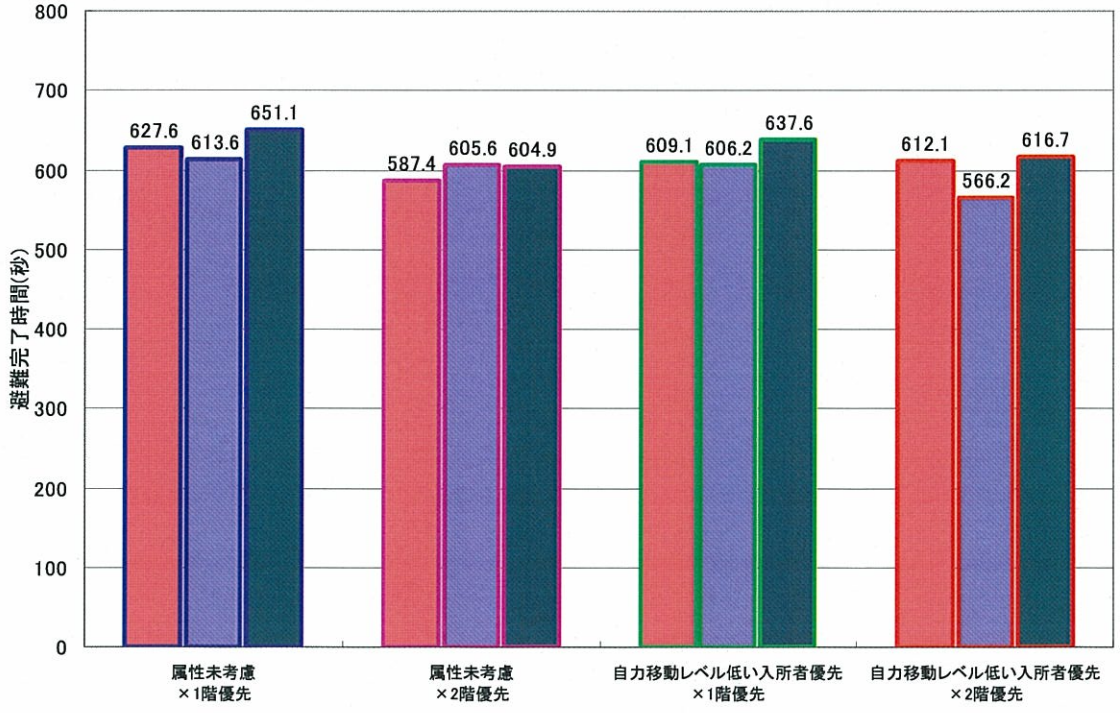


図 4.5.4　初期配置変更ケース 避難完了時刻比較

左：現行配置，中：移動レベル低い入所者 1 階配置，右：移動レベル低い入所者 2 階配置

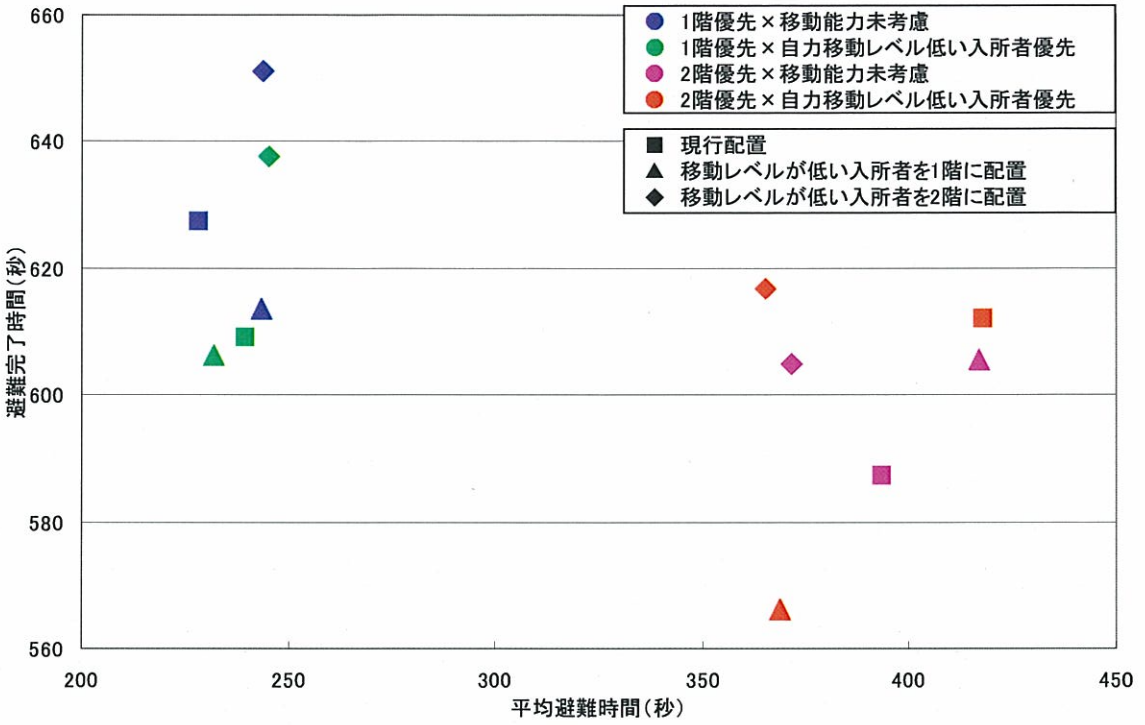


図 4.5.5　初期配置変更ケース 避難完了時刻・平均避難時間プロット

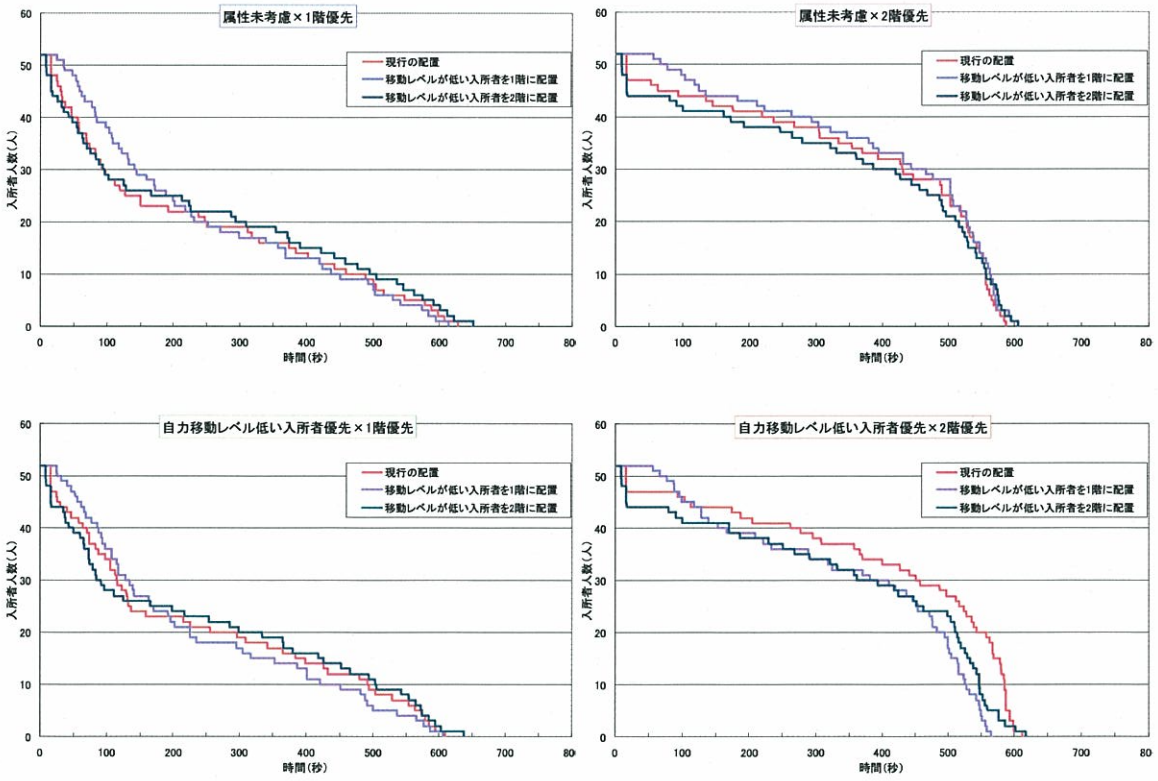


図 4.5.6　初期配置変更ケース 入所者人数時間経過（試行毎）

4.5.4　ジニ係数を用いた平等性に関する分析

次に，ジニ係数を求め，各試行の平等性について分析を行う．ジニ係数は 0 に近づくほど不平等性が小さいことを表す．表 4.5.7，図 4.5.8 にジニ係数を求めた結果を示す．

入所者の移動能力を考慮する方針と，考慮しない方針でそれぞれ平等性が大きく異なる結果となった．一方，避難優先階や初期配置は平等性に大きくは影響しない結果となった．

移動能力が低い入所者を先に避難誘導させることで，最初に避難が完了する入所者と最後に避難が完了する入所者の時間差が小さくなり，施設全体の避難に関する平等性が高まることは理にかなった結果であると言える．

表 4.5.7　初期配置変更ケース ジニ係数

ジニ係数	現行の配置	移動レベルが低い入所者を 1 階に配置	移動レベルが低い入所者を 2 階に配置
入所者の属性を考慮しない × 1 階優先	0.64	0.60	0.63
入所者の属性を考慮しない × 2 階優先	0.61	0.62	0.62
自力移動レベルが低い入所者を優先 × 1 階優先	0.33	0.31	0.39
自力移動レベルが低い入所者を優先 × 2 階優先	0.32	0.35	0.41

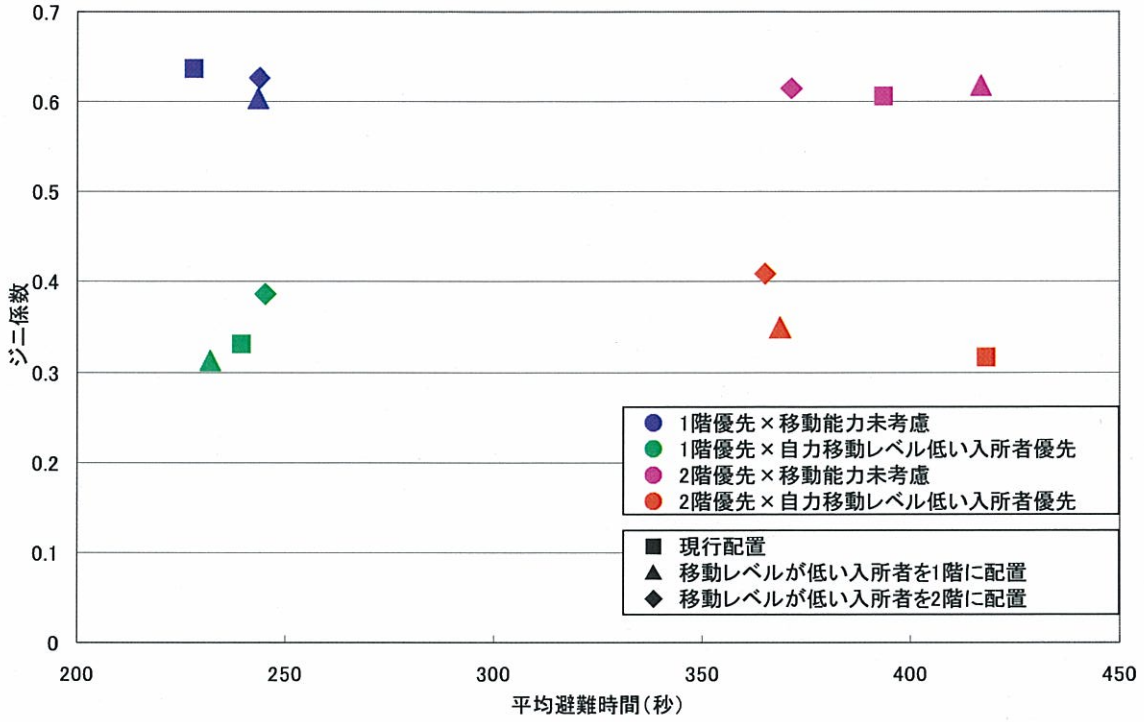


図 4.5.8　初期配置変更ケース避難完了時刻・ジニ係数プロット

4.6　火災発生シナリオにおける避難誘導方法の比較

対象施設において，シミュレーション上で火災を表現し，その際の避難誘導方法について比較分析を行う．前述 3.3.4 火災発生モジュールでの設定と日中シナリオでの避難完了時刻を比較すると，避難不能になる入所者や職員は発生しないと考えられるため，夜間シナリオを基本としてシミュレーション実験を行う．

なお，本モデルでは階移動を行う際に，自エージェントから近い階段を選択するアルゴリズムになっているが，火災発生シナリオでは 240 秒経過後に階移動の必要がある場合は，全て東側の階段に向かうものとして火災から遠ざかる行動を表現した．

4.6.1　試行内容

初期配置は夜間の通常ケースを用い，自力歩行可能な入所者は階段単独歩行可能とする．図 4.6.1 に避難不能時間の設定を示す．各区画において一定時間を経過すると，その区画にいるエージェントは避難不能とみなす．

表 4.6.1　火災発生ケース 試行内容

試行番号	初期配置				火災の影響を考慮		避難誘導方針策定			避難誘導戦術の考慮				入所者運動能力	
	日中シナリオ	夜間シナリオ			火災の影響		移動能力での優先順位		距離による優先順位	階毎の避難誘導方針		2階から1階へのリレー		階段単独降下	
	通常	通常	移動レベルが低いエージェントを1階へ	移動レベルが低いエージェントを2階へ	考慮しない	考慮する	考慮しない	自力移動レベルが低い	職員に近い	1階から先に誘導	2階から先に誘導	行わない	行う	不可能	可能
F1_1		○				○	○		○	○		○			○
F1_2		○				○	○		○		○	○			○
F1_3		○				○		○	○	○		○			○
F1_4		○				○		○	○		○	○			○

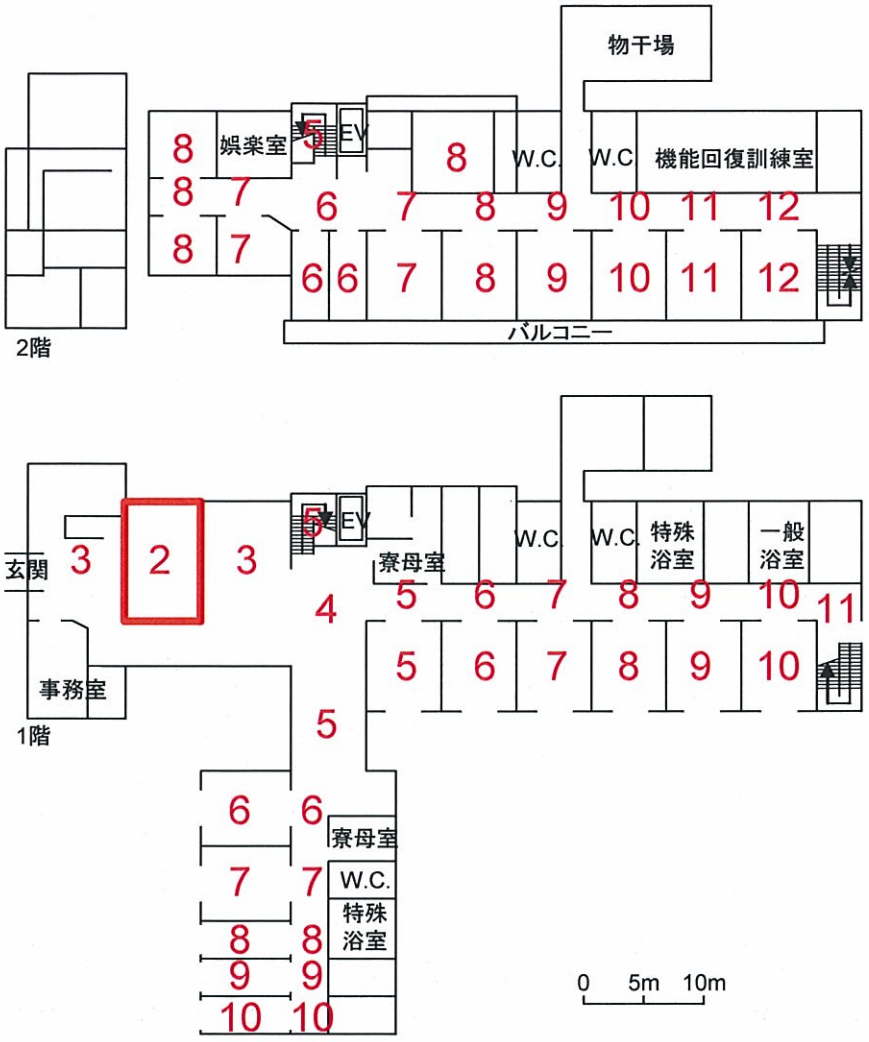


図 4.6.2　避難不能時刻設定（単位：分）（再掲）

4.6.2　実験結果

実験結果を表 4.6.3，図 4.6.4 に示す．避難活動終了時間は，全ての入所者が避難完了または避難不能になった時間を表している．避難不能職員数は，避難活動終了時間での人数を示している．

1 階優先での避難誘導では入所者の，2 階優先での避難では職員の避難不能人数が増加している．

4.6.3　分析・考察

2 階を優先する避難誘導が，多くの入所者の避難誘導に成功する可能性が高いことを示す結果となった．特に，自力移動レベルが低い入所者を優先しての避難誘導において，避難不能の入所者が少なくなったのは，自力移動できない入所者を可能な限り早い時間帯に避難させることが，煙や火災に巻き込まれる人数を減らすことができることを示しており，これは理にかなった結果であると言える．

表 4.6.3　火災発生ケース 実験結果

火災発生ケース		1 階優先	2 階優先
入所者の属性を考慮しない	避難活動終了時間(秒)	529.9	600.1
	避難不能入所者数(人)	9	7
	避難不能職員数(人)	2	7
自力移動レベルが低い入所者を優先	避難活動終了時間(秒)	603.4	600.0
	避難不能入所者数(人)	7	3
	避難不能職員数(人)	2	5

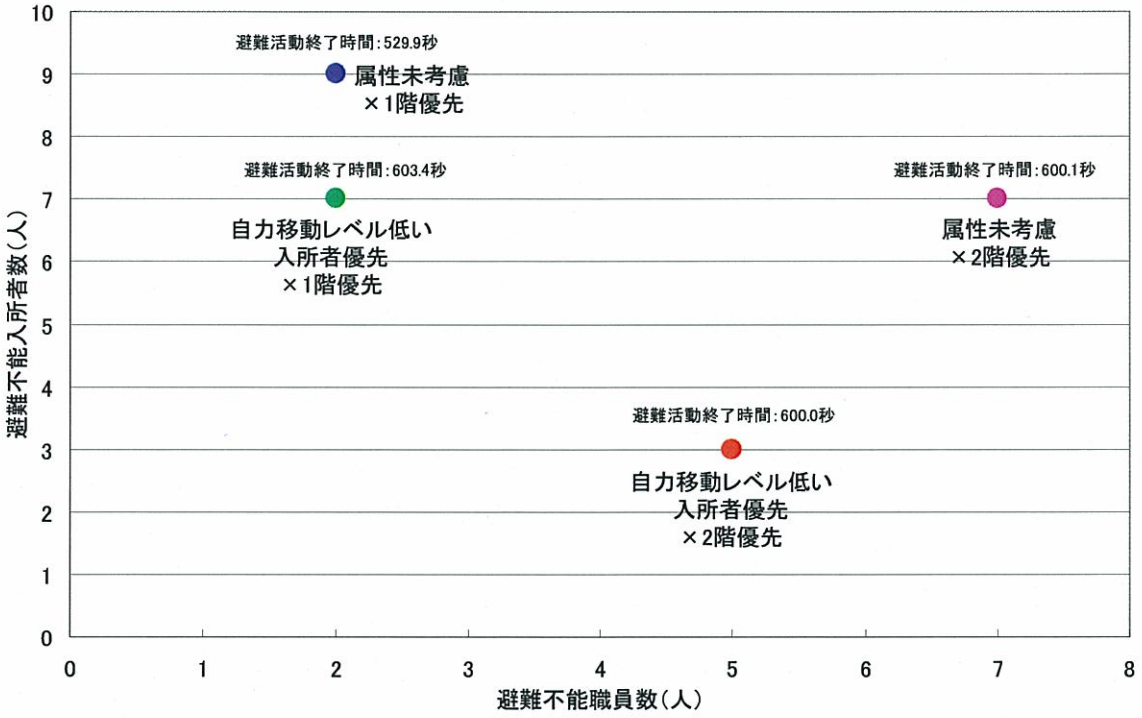


図 4.6.4　火災発生ケース 避難不能入所者・職員プロット

第5章　結論

5.1 本研究の結論

本研究では避難誘導シミュレーションを構築し，シナリオや誘導方法を変化させながらシミュレーション実験を行った．本章では本研究で明らかになった点をまとめる．

5.1.1 複数階建物であることを考慮した避難誘導戦術の効果

日中シナリオにおいて，2階から1階へリレーする避難誘導の試行を行ったが，多少避難完了時間が短縮した程度で，大きな効果は認められなかった．これは，リレーを行うことによって，1階階段付近での避難準備作業回数の増加が原因となり，予想していたほど効率的な方法ではなかったと言える．

5.1.2 入所者の移動能力を考慮した避難誘導の効果

日中・夜間のシナリオに関わらず，入所者の移動能力を考慮し，自力移動が困難な入所者から順次避難誘導する方法に一定の効果があることが得られた．自力で動ける入所者は自力で，動けない入所者は職員が，という合理的な考えの下での避難誘導をすることによって，避難完了時間を短縮することができる．

5.1.3 階別優先順位を考慮した避難誘導の効果

1階を優先して避難誘導を行うと，避難開始から早い時間である程度の入所者を避難させることができ，比較指標からも1階を優先した避難誘導が効率的である結果を導くことができた．しかし，避難完了時間については，2階を優先しての避難誘導が早く完了する傾向にある．これは，2階を優先しての避難誘導であれば，夜間シナリオにおいて到着する職員が，すぐに入所者にアプローチできるためであると考えられる．

5.1.4 初期配置変更による避難誘導の効果

自力移動できない入所者を2階に配置することは，入所者の移動能力を考慮した避難誘導を行う試行において，施設全体の避難誘導の効率性が高まる結果となった．入所者の移動能力を考慮して居室計画を練る場合は，避難誘導方針でも，移動能力を考慮した避難誘導を行わないことには，その効果が発揮されないことが明らかになった．

自力移動できない入所者が2階に多くなることは，自力移動可能な入所者が1階に多くいることになり，避難開始から早い時間帯で自力避難を済ませることができ，施設全体で見ると効率的である結果を導き出すことができた．

平均避難時間は避難優先階に依存し，避難誘導の平等性は誘導方針に依存することから，避難完了時間の短縮を目標にするだけでは避難安全性を確保しきれない示唆を得た．

5.1.5　火災発生ケースによる避難誘導方法の評価

2階を優先しての避難誘導が効果的である結果を導き出すことができた．さらに，自力移動ができない入所者を優先的に避難させることで，より避難成功する確率を高めることができると言える．

5.2　介護保険施設での避難誘導に関するまとめと提言

本研究では効果的な避難誘導を達成するために，

- 避難完了時間の短縮
- 平均避難時間の短縮
- 平等性の確保
- 避難不能人数の減少

の 4 つについてアプローチを試みた．この内，平等性の確保と避難不能人数の減少を達成するためには，移動能力が低い入所者を優先的に避難させることに一定の効果があることが明らかになった．一方で，避難完了時間の短縮のためには 2 階を優先した避難誘導，平均避難時間の短縮のためには 1 階を優先した避難誘導が効果的である結果が導出され，この 2 点についてはトレードオフの関係にあることが明らかになった．

以上より，入所者の移動能力と位置を把握し，非合理的な行動が起きない避難誘導方法を選択することが必要であると言える．

5.3 今後の課題

5.3.1 完全情報下でない試行の必要性

本研究で構築したモデルは，職員エージェントが避難誘導を継続するか終了するかや，2階へ移動するかなどの判断材料として，施設内の人数を用いている．また，職員エージェントが対応していない入所者エージェントの中で最近接のエージェントを捜索することなく発見できる．つまり，本研究で構築したシミュレーションモデルは，エージェントの在階に関わりなく，施設内全ての情報を各エージェントが有している完全情報下によるシミュレーションモデルであるといえる．

実際の避難においては，全ての情報を有していることはありえないため，施設内の人数，入所者の位置などを未知とし，職員エージェントが施設内を捜索するフェーズと，エージェント同士で避難完了情報を同期することなどを実装することで，本モデル以上に実際の避難で起こりうる現象を表現できるのではないかと考えられる．

5.3.2 シミュレーションモデルの他施設への適用可能性の検討

本研究で構築したシミュレーションモデルは，今回取り上げた対象施設でのシミュレーション実験のみを行った．本モデルは，対象施設に特化したものとして構築したわけではないが，このモデルが一般性を持った提言に繋げるためには，同様の介護保険施設での試行が必要であると考えられる．

特に，今回の対象施設の特徴として，1階から外部空間へ通ずる出口が非常に多いため，比較的避難するのは容易な施設である可能性がある．外部空間への出口が少ない施設などの試行を行うことで，今回は発見できなかった避難誘導に関する別の問題点が導出されることが考えられる．

5.3.3 シミュレーションモデルの精緻化

(1) 火災発生モジュール

本モデルで用いた避難不能時間の算定は，各居室や避難経路が危険になる状況までの時間を足し合わせたものである．実際の火災では，特に煙の拡散・伝播によって巻き込まれて避難不能になる場合が多いと考えられるが，本モデルの火災発生モジュールでは拡散・伝播を詳細に表現しているわけではなく，火災発生時点から火災もしくは煙によって避難できなくなるであろう時刻を設定したものである．健常者と比較して運動能力が低い入所者が，火災時にどの程度の状況までなら避難可能かを考慮した上で，火災発生を組み込む必要があると考えられる．

(2) 職員エージェント移動モジュール

職員エージェントの歩行速度を常に一定として設定したが，実際は避難誘導を続けていることで疲労により歩行速度が低下することが予想される．本モデルで用いた 3.11m/秒という歩行速度は小走り程度であり，夜間シナリオでの避難に 600 秒前後要していることを考慮すると，実際の避難には本研究の結果以上の時間を要すると予想される．

また，本モデルで入所者エージェントを発見する際に，方向の概念を取り入れておらず，一定距離に近づくことで，発見や接触を全て表現した．前述の全ての情報を有していない情報下（＝非完全情報下）での試行において，捜索というフェーズは欠かせず，その際には職員の視野・視界を考慮して入所者発見の判断フローを組み込む必要があると考えられる．

(3) 入所者エージェント移動モジュール

入所者エージェントはどの試行においても同じ行動を行うことを前提としてモデル化したが，実際の避難では転倒することや，職員に代わって避難誘導を補助することなどが考えられる．それぞれ確率的に，エージェントの転倒や役割変更を発生させ，避難にどの程度影響するかを分析する必要があると思われる．

(4) ソフトウェアに依存した乱数性への配慮

本モデルでは，乱数を発生させず，全ての試行が同じ結果を導出するものとして構築したが，シミュレーション実行の際に，各エージェントの計算順序は試行毎に異なることが発見された．これは，本研究における夜間シナリオであれば，エージェント間の距離に余裕があるため影響はほとんどないが，日中シナリオでエージェントの関係が密であると，計算順序によってエージェントの挙動が異なることが懸念される．このような乱数性の影響を受けないような配慮をする必要があると言える．

参考文献・資料

1) 統計局ホームページ，http://www.stat.go.jp/

2) 平成 17 年介護サービス施設・事業所調査結果の概況：厚生労働省

3) 認知症高齢者グループホーム等における防火安全対策検討会報告書：総務省消防庁，平成 18 年 3 月 29 日

4) 改訂版イラストレーション建築基準法：高木任之：学芸出版社，2003 年 11 月

5) WAM NET（独立行政法人福祉医療機構ホームページ），http://www.wam.go.jp/

6) 福祉住環境コーディネータ検定試験 3 級公式テキスト：東京商工会議所 編，2007 年 2 月

7) 平成 17 年度体力・運動能力調査：文部科学省，平成 18 年 10 月 9 日

8) 介護保険施設における非常時の避難誘導に関する基礎的研究：山下恵，糸井川栄一：地域安全学会梗概集 No.21，P81-86，2007 年 11 月

9) 道路閉塞・火災延焼被害を考慮した地震時における災害時要援護者救助シミュレーション：上田遼，瀬尾和大，元木健太郎：地域安全学会論文集，No.8，P341-348，2006 年 11 月

10) 避難シミュレーションに基づく高齢者施設の避難安全性の確保に関する考察：海老原学，掛川秀史：日本建築学会計画系論文集，第 521 号，P1-8，1999 年 7 月

11) オブジェクト指向に基づく避難・介助行動シミュレーションモデル海老原学，掛川秀史：日本建築学会計画系論文集，第 467 号，P1-12，1995 年 1 月

12) MAS コミュニティ，http://mas.kke.co.jp/

13) 建築・都市計画のためのモデル分析の手法：日本建築学会：井上書院，1992 年

14) 認知症高齢者グループホーム等の社会福祉施設における防火安全対策のための消防法施行令等の一部改正：総務省消防庁，平成 19 年 6 月 13 日

謝　辞

指導教員である筑波大学システム情報工学研究科リスク工学専攻糸井川栄一教授には，大学４年次から大学院２年次の３年間もの間，公私に渡って大変お世話になりました．研究に関する手厚いご指導のみならず，リスク工学専攻での講義，災害調査，研究室運営など多岐に渡りご指導いただきました．ここに深く感謝の意を表します．先生からご指導いただいたことを忘れずに，今後の人生も精進してまいりたいと思います．

熊谷良雄筑波大学名誉教授には，ご多忙の中でゼミにご出席いただき，幾度となく的確なご指導をいただきました．また，毎週のゼミ以外でも，学生部屋にいらして気にかけていただきました．研究を進めるにあたって，私が何をするべきか，何を考えなければならないかの指針を示していただきました．ここに深く感謝の意を表します．

システム情報工学研究科リスク工学専攻村尾修准教授には，リスク工学専攻での講義を中心に，都市リスク，都市防災の考え方について，幅広くアドバイスいただきました．感謝申し上げます．

システム情報工学研究科リスク工学専攻梅本通孝講師には，毎週のゼミでたくさんのアドバイスをいただきました．また，個別にご指導いただいた本研究のモデルのアルゴリズムについては，梅本先生のご指導なくしては完成することはできませんでした．ここに深く感謝の意を表します．

都市防災研究室 OG の山下恵さんには，貴重な研究資料を提供していただき，研究を進めるにあたってのアドバイスをいただきました．感謝申し上げます．

対象とした施設の職員の皆様には，施設の見学などで貴重なお時間を割いていただきました．心より感謝申し上げます．

都市防災研究室の M1，４年生の皆さんには，１年間通じて研究に集中できる環境を提供していただきました．私のわがままで食事に連れ出したりしても，嫌な顔せずについてきてくださり，良い気分転換となりました．ありがとうございました．

リスク工学専攻の２年間をともに過ごした田口元寿さん，山浦浩太さん，そして大学４年次からともに都市防災研究室にお世話になった子田大雄さんとは，研究室での多くの時間をすごしました．時に苛々し，時に手いっぱいになりながらも，この２年間を良き思い出として振り返ることができるのは皆様のおかげです．ありがとうございました．

最後に，大学４年間と合わせて６年間も学業に励ませてくれた両親をはじめとする家族の支えや，社会工学類都市計画主専攻同期生，システム情報工学研究科リスク工学専攻同期生との切磋琢磨があって，本研究を完成させることができました．ここに，心より感謝申し上げます．

2008 年 3 月 21 日

鎌田 智之

付録

```
Univ_Init{

}

Univ_Step_Begin{
            //評価指標計算モジュール
            @count()

            //火災発生モジュール
            @fire()

            //staff 到着モジュール
            @staffarrive()

}

Univ_Step_End{

}

Univ_Finish{

}


//火災発生モジュール
Sub fire()
{
            Dim xa as Integer
            Dim xb as Integer
            Dim ya as Integer
            Dim yb as Integer
            Dim xa2nd as Integer
            Dim xb2nd as Integer
            Dim ya2nd as Integer
            Dim yb2nd as Integer

            //火災発生時刻
            if 120.0 < Universe.TIME then
                        //火災エリア(xa, xb) (ya, yb)
                        xa = 36
                        xb = 60
                        ya = 32
                        yb = 70
                        @burnlst(xa, xb, ya, yb)                        //人エージェントの処理フロー
                        @cutnetworklst(xa, xb, ya, yb)          //ネットワーク切断フロー
            end if
            if 180.0 < Universe.TIME then
                        //火災エリア(xa, xb) (ya, yb)
                        xa = 2
                        xb = 83
                        ya = 32
                        yb = 83
                        @burnlst(xa, xb, ya, yb)                        //人エージェントの処理フロー
                        @cutnetworklst(xa, xb, ya, yb)          //ネットワーク切断フロー
            end if
            if 240.0 < Universe.TIME then
                        //火災エリア(xa, xb) (ya, yb)
                        xa = 84
                        xb = 110
                        ya = 45
                        yb = 83
                        @burnlst(xa, xb, ya, yb)                        //人エージェントの処理フロー
                        @cutnetworklst(xa, xb, ya, yb)          //ネットワーク切断フロー
            end if
            if 300.0 < Universe.TIME then
                        //火災エリア(xa, xb) (ya, yb)
                        xa = 84
                        xb = 110
                        ya = 84
                        yb = 135
                        @burnlst(xa, xb, ya, yb)                        //人エージェントの処理フロー
                        @cutnetworklst(xa, xb, ya, yb)          //ネットワーク切断フロー
                        //火災エリア(xa, xb) (ya, yb)
                        xa = 111
                        xb = 135
                        ya = 59
                        yb = 100
                        @burnlst(xa, xb, ya, yb)                        //人エージェントの処理フロー
                        @cutnetworklst(xa, xb, ya, yb)          //ネットワーク切断フロー
                        //火災エリア(xa, xb) (ya, yb)
                        xa = 91
                        xb = 117
                        ya = 80
                        yb = 105
                        @burn2nd(xa, xb, ya, yb)                        //人エージェントの処理フロー
                        @cutnetwork2nd(xa, xb, ya, yb)          //ネットワーク切断フロー
            end if
            if 360.0 < Universe.TIME then
                        //火災エリア(xa, xb) (ya, yb)
                        xa = 56
```

```
            xb = 119
            ya = 136
            yb = 158
            @burn1st(xa, xb, ya, yb)                    //人エージェントの処理フロー
            @cutnetwork1st(xa, xb, ya, yb)              //ネットワーク切断フロー
            //火災エリア(xa, xb) (ya, yb)
            xa = 136
            xb = 159
            ya = 59
            yb = 100
            @burn1st(xa, xb, ya, yb)                    //人エージェントの処理フロー
            @cutnetwork1st(xa, xb, ya, yb)              //ネットワーク切断フロー
            //火災エリア(xa, xb) (ya, yb)
            xa = 67
            xb = 117
            ya = 80
            yb = 134
            @burn2nd(xa, xb, ya, yb)                    //人エージェントの処理フロー
            @cutnetwork2nd(xa, xb, ya, yb)              //ネットワーク切断フロー
end if
if 420.0 < Universe.TIME then
            //火災エリア(xa, xb) (ya, yb)
            xa = 56
            xb = 108
            ya = 160
            yb = 182
            @burn1st(xa, xb, ya, yb)                    //人エージェントの処理フロー
            @cutnetwork1st(xa, xb, ya, yb)              //ネットワーク切断フロー
            //火災エリア(xa, xb) (ya, yb)
            xa = 160
            xb = 183
            ya = 59
            yb = 100
            @burn1st(xa, xb, ya, yb)                    //人エージェントの処理フロー
            @cutnetwork1st(xa, xb, ya, yb)              //ネットワーク切断フロー
            //火災エリア(xa, xb) (ya, yb)
            xa = 44
            xb = 66
            ya = 45
            yb = 120
            @burn2nd(xa, xb, ya, yb)                    //人エージェントの処理フロー
            @cutnetwork2nd(xa, xb, ya, yb)              //ネットワーク切断フロー
            //火災エリア(xa, xb) (ya, yb)
            xa = 118
            xb = 142
            ya = 94
            yb = 134
            @burn2nd(xa, xb, ya, yb)                    //人エージェントの処理フロー
            @cutnetwork2nd(xa, xb, ya, yb)              //ネットワーク切断フロー
end if
if 480.0 < Universe.TIME then
            //火災エリア(xa, xb) (ya, yb)
            xa = 56
            xb = 96
            ya = 184
            yb = 194
            @burn1st(xa, xb, ya, yb)                    //人エージェントの処理フロー
            @cutnetwork1st(xa, xb, ya, yb)              //ネットワーク切断フロー
            //火災エリア(xa, xb) (ya, yb)
            xa = 184
            xb = 208
            ya = 59
            yb = 100
            @burn1st(xa, xb, ya, yb)                    //人エージェントの処理フロー
            @cutnetwork1st(xa, xb, ya, yb)              //ネットワーク切断フロー
            //火災エリア(xa, xb) (ya, yb)
            xa = 134
            xb = 167
            ya = 68
            yb = 134
            @burn2nd(xa, xb, ya, yb)                    //人エージェントの処理フロー
            @cutnetwork2nd(xa, xb, ya, yb)              //ネットワーク切断フロー
end if
if 540.0 < Universe.TIME then
            //火災エリア(xa, xb) (ya, yb)
            xa = 56
            xb = 96
            ya = 196
            yb = 206
            @burn1st(xa, xb, ya, yb)                    //人エージェントの処理フロー
            @cutnetwork1st(xa, xb, ya, yb)              //ネットワーク切断フロー
            //火災エリア(xa, xb) (ya, yb)
            xa = 209
            xb = 232
            ya = 59
            yb = 100
            @burn1st(xa, xb, ya, yb)                    //人エージェントの処理フロー
            @cutnetwork1st(xa, xb, ya, yb)              //ネットワーク切断フロー
            //火災エリア(xa, xb) (ya, yb)
            xa = 168
            xb = 192
```

```
                ya = 94
                yb = 134
                @burn2nd(xa, xb, ya, yb)                    //人エージェントの処理フロー
                @cutnetwork2nd(xa, xb, ya, yb)              //ネットワーク切断フロー
        end if
        if 600.0 < Universe.TIME then
                //火災エリア(xa, xb) (ya, yb)
                xa = 56
                xb = 96
                ya = 208
                yb = 218
                @burn1st(xa, xb, ya, yb)                    //人エージェントの処理フロー
                @cutnetwork1st(xa, xb, ya, yb)              //ネットワーク切断フロー
                //火災エリア(xa, xb) (ya, yb)
                xa = 234
                xb = 255
                ya = 59
                yb = 100
                @burn1st(xa, xb, ya, yb)                    //人エージェントの処理フロー
                @cutnetwork1st(xa, xb, ya, yb)              //ネットワーク切断フロー
                //火災エリア(xa, xb) (ya, yb)
                xa = 193
                xb = 217
                ya = 94
                yb = 134
                @burn2nd(xa, xb, ya, yb)                    //人エージェントの処理フロー
                @cutnetwork2nd(xa, xb, ya, yb)              //ネットワーク切断フロー
        end if
        if 660.0 < Universe.TIME then
                //火災エリア(xa, xb) (ya, yb)
                xa = 256
                xb = 270
                ya = 59
                yb = 81
                @burn1st(xa, xb, ya, yb)                    //人エージェントの処理フロー
                @cutnetwork1st(xa, xb, ya, yb)              //ネットワーク切断フロー
                //火災エリア(xa, xb) (ya, yb)
                xa = 218
                xb = 242
                ya = 94
                yb = 134
                @burn2nd(xa, xb, ya, yb)                    //人エージェントの処理フロー
                @cutnetwork2nd(xa, xb, ya, yb)              //ネットワーク切断フロー
                //火災エリア(xa, xb) (ya, yb)
                xa = 267
                xb = 281
                ya = 94
                yb = 114
                @burn2nd(xa, xb, ya, yb)                    //人エージェントの処理フロー
                @cutnetwork2nd(xa, xb, ya, yb)              //ネットワーク切断フロー
        end if
        if 720.0 < Universe.TIME then
                //火災エリア(xa, xb) (ya, yb)
                xa = 218
                xb = 266
                ya = 94
                yb = 134
                @burn2nd(xa, xb, ya, yb)                    //人エージェントの処理フロー
                @cutnetwork2nd(xa, xb, ya, yb)              //ネットワーク切断フロー
        end if

}

//1階での火災巻き込まれ処理
Sub burn1st( xa as Integer, xb as Integer, ya as Integer, yb as Integer )
{
        Dim human1st as AgtSet
        Dim staff1st as AgtSet
        MakeAgtSet(human1st, Universe.FLOOR1ST.HUMAN)
        MakeAgtSet(staff1st, Universe.FLOOR1ST.STAFF)
        //人エージェントの処理フロー
        for each obj in human1st
                if xa <= obj.X AND obj.X <= xb then
                        if ya <= obj.Y AND obj.Y <= yb then
                                //巻き込まれ処理
                                KillAgt(obj)
                        end if
                end if
        next obj
        for each obj in staff1st
                if xa <= obj.X AND obj.X <= xb then
                        if ya <= obj.Y AND obj.Y <= yb then
                                //巻き込まれ処理
                                Universe.IMPOSSIBLESTAFF = Universe.IMPOSSIBLESTAFF + 1
                                KillAgt(obj)
                        end if
                end if
        next obj
}
```

```
//2階での火災巻き込まれ処理
Sub burn2nd( xa as Integer, xb as Integer, ya as Integer, yb as Integer )
{
        Dim human2nd as AgtSet
        Dim staff2nd as AgtSet
        MakeAgtSet(human2nd, Universe.FLOOR2ND.HUMAN)
        MakeAgtSet(staff2nd, Universe.FLOOR2ND.STAFF)
        //人エージェントの処理フロー
        for each obj in human2nd
            if xa <= obj.X AND obj.X <= xb then
                if ya <= obj.Y AND obj.Y <= yb then
                    //巻き込まれ処理
                    Universe.IMPOSSIBLEHUMAN = Universe.IMPOSSIBLEHUMAN + 1
                    KillAgt(obj)
                end if
            end if
        next obj
        for each obj in staff2nd
            if xa <= obj.X AND obj.X <= xb then
                if ya <= obj.Y AND obj.Y <= yb then
                    //巻き込まれ処理
                    Universe.IMPOSSIBLESTAFF = Universe.IMPOSSIBLESTAFF + 1
                    KillAgt(obj)
                end if
            end if
        next obj
}

//職員到着モジュール
Sub staffarrive()
{
        Dim newstaff1 as Agt
        Dim newstaff2 as Agt
        Dim newstaff3 as Agt

        //初期化
        newstaff1 = Universe.FLOOR1ST.WALL(0)
        newstaff2 = Universe.FLOOR1ST.WALL(0)
        newstaff3 = Universe.FLOOR1ST.WALL(0)

        //職員到着モジュール
        if 300.0 <= Universe.TIME AND Universe.TIME < 300.1 then
            newstaff1 = CreateAgt(Universe.FLOOR1ST.STAFF)
        elseif  420.0 <= Universe.TIME AND Universe.TIME < 420.1 then
            newstaff1 = CreateAgt(Universe.FLOOR1ST.STAFF)
        elseif  480.0 <= Universe.TIME AND Universe.TIME < 480.1 then
            newstaff1 = CreateAgt(Universe.FLOOR1ST.STAFF)
            newstaff2 = CreateAgt(Universe.FLOOR1ST.STAFF)
            newstaff3 = CreateAgt(Universe.FLOOR1ST.STAFF)
        elseif  540.0 <= Universe.TIME AND Universe.TIME < 540.1 then
            newstaff1 = CreateAgt(Universe.FLOOR1ST.STAFF)
            newstaff2 = CreateAgt(Universe.FLOOR1ST.STAFF)
        elseif  660.0 <= Universe.TIME AND Universe.TIME < 660.1 then
            newstaff1 = CreateAgt(Universe.FLOOR1ST.STAFF)
        elseif  780.0 <= Universe.TIME AND Universe.TIME < 780.1 then
            newstaff1 = CreateAgt(Universe.FLOOR1ST.STAFF)
        elseif  840.0 <= Universe.TIME AND Universe.TIME < 840.1 then
            newstaff1 = CreateAgt(Universe.FLOOR1ST.STAFF)
        elseif  900.0 <= Universe.TIME AND Universe.TIME < 900.1 then
            newstaff1 = CreateAgt(Universe.FLOOR1ST.STAFF)
            newstaff2 = CreateAgt(Universe.FLOOR1ST.STAFF)
            newstaff3 = CreateAgt(Universe.FLOOR1ST.STAFF)
        elseif  1020.0 <= Universe.TIME AND Universe.TIME < 1020.1 then
            newstaff1 = CreateAgt(Universe.FLOOR1ST.STAFF)
        elseif  1140.0 <= Universe.TIME AND Universe.TIME < 1140.1 then
            newstaff1 = CreateAgt(Universe.FLOOR1ST.STAFF)
        elseif  1200.0 <= Universe.TIME AND Universe.TIME < 1200.1 then
            newstaff1 = CreateAgt(Universe.FLOOR1ST.STAFF)
        elseif  1320.0 <= Universe.TIME AND Universe.TIME < 1320.1 then
            newstaff1 = CreateAgt(Universe.FLOOR1ST.STAFF)
        end if

        //生成座標設定
        newstaff1.X = 244
        newstaff1.Y = 113
        newstaff2.X = 240
        newstaff2.Y = 113
        newstaff3.X = 236
        newstaff3.Y = 113

}

//評価指標計算モジュール
Sub count()
{
        Dim i as Integer
        Dim obj as Agt
        Dim staff1st as AgtSet
        Dim staff2nd as AgtSet

        //初期化
```

```
            Universe.FREEHUMAN1ST = 0
            Universe.FREEHUMAN2ND = 0
            Universe.FREESTAFF1ST = 0
            Universe.FREESTAFF2ND = 0
            Universe.TOSTAIRSSTAFF = 0

            //時間カウント
            Universe.TIME = Universe.TIME +  1.0 * Universe.DELTATIME

            //人数カウント
            Universe.STAFF1STCOUNT = CountAgt(Universe.FLOOR1ST.STAFF)
            Universe.STAFF2NDCOUNT = CountAgt(Universe.FLOOR2ND.STAFF)
            Universe.HUMAN1STCOUNT = CountAgt(Universe.FLOOR1ST.HUMAN)
            Universe.HUMAN2NDCOUNT = CountAgt(Universe.FLOOR2ND.HUMAN)

            //フリーの入所者人数カウント
            For i = 0 to (CountAgt(Universe.FLOOR1ST.HUMAN) -1)
                        if Universe.FLOOR1ST.HUMAN(i).STATUS == 99 then
                                    Universe.FREEHUMAN1ST = Universe.FREEHUMAN1ST + 1
                        end if
            next i
            For i = 0 to (CountAgt(Universe.FLOOR2ND.HUMAN) - 1)
                        if Universe.FLOOR2ND.HUMAN.STATUS(i) == 99 then
                                    Universe.FREEHUMAN2ND = Universe.FREEHUMAN2ND + 1
                        end if
            next i

            //フリーの職員人数カウント
            For i = 0 to (CountAgt(Universe.FLOOR1ST.STAFF) -1)
                        if Universe.FLOOR1ST.STAFF(i).STATUS == 1  then
                                    Universe.FREESTAFF1ST = Universe.FREESTAFF1ST + 1
                        end if
            next i
            For i = 0 to (CountAgt(Universe.FLOOR2ND.STAFF) - 1)
                        if Universe.FLOOR2ND.STAFF.STATUS(i) == 1  then
                                    Universe.FREESTAFF2ND = Universe.FREESTAFF2ND + 1
                        end if
            next i

            //2階へ行く職員人数カウント
            For i = 0 to (CountAgt(Universe.FLOOR1ST.STAFF) - 1)
                        if Universe.FLOOR1ST.STAFF.STATUS(i) == 6  OR Universe.FLOOR1ST.STAFF.STATUS(i) == 88 then
                                    Universe.TOSTAIRSSTAFF = Universe.TOSTAIRSSTAFF + 1
                        end if
            next i

            //STAFF総移動距離
            MakeAgtSet(staff1st, Universe.FLOOR1ST.STAFF)
            MakeAgtSet(staff2nd, Universe.FLOOR2ND.STAFF)
            For each obj in staff1st
                        Universe.STAFFDIST = Universe.STAFFDIST + obj.V / Universe.GLID * Universe.DELTATIME
            next i
            For each obj in staff2nd
                        Universe.STAFFDIST = Universe.STAFFDIST + obj.V / Universe.GLID * Universe.DELTATIME
            next i
}

//1階に関するダイクストラ法計算
Sub routedecide1st (n as integer, m as integer)
{
            Dim i As Integer
            Dim j As Integer
            Dim u As Integer
            Dim k As Integer
            Dim s As Integer                                    'スタート地点のノードid
            Dim g As Integer                                    'ゴール地点のノードid
            Dim A(65, 16) As Double
            Dim dij(62, 3) As Double
            Dim rev(30) As Integer                  'ゴールからスタートへの順番を格納
            Dim fwd(30) As Integer                  'revを反転させて最短経路を格納
            Dim fwdsize As integer                  'fwd配列の大きさとして使用
            Dim tmp As Double
            Dim mindist As Double
            Dim labelcount As Integer

            'A()にデータを格納
            For i = 0 To 64
                        A(i, 1) = Universe.FLOOR1ST.POINT.ID(i)
                        A(i, 2) = Universe.FLOOR1ST.POINT.X(i)
                        A(i, 3) = Universe.FLOOR1ST.POINT.Y(i)
                        A(i, 4) = Universe.FLOOR1ST.POINT.Layer(i)
                        A(i, 5) = Universe.FLOOR1ST.POINT.Direction(i)
                        A(i, 6) = Universe.FLOOR1ST.POINT.POT1(i)
                        A(i, 7) = Universe.FLOOR1ST.POINT.n1(i)
                        A(i, 8) = Universe.FLOOR1ST.POINT.n2(i)
                        A(i, 9) = Universe.FLOOR1ST.POINT.n3(i)
                        A(i, 10) = Universe.FLOOR1ST.POINT.n4(i)
                        A(i, 11) = Universe.FLOOR1ST.POINT.n5(i)
                        A(i, 12) = Universe.FLOOR1ST.POINT.d1(i)
                        A(i, 13) = Universe.FLOOR1ST.POINT.d2(i)
                        A(i, 14) = Universe.FLOOR1ST.POINT.d3(i)
```

```
            A(i, 15) = Universe.FLOOR1ST.POINT.d4(i)
            A(i, 16) = Universe.FLOOR1ST.POINT.d5(i)
Next i


'●●ここからダイクストラ法のアルゴリズム●●

スタート地点のノードｓとする
'目的地ノードをｇとする
s = n
g = m

'●ここからSTEP1●

'dij(i,1)は計算用の値（ノードにおける仮の距離）を入れる
'dij(i,2)はフラグを入れる
'ノードとノードの距離を大数にする
'dij(i,2)=99は仮ラベル
'dij(i,2)=999は確定ラベル
'dij(i,3)は経過ノードを格納
For i = 0 To 62
            dij(i, 1) = 5000
            dij(i, 2) = 0
Next i

'スタートノードの設定
dij(s, 1) = 0                       '自分への距離を0とする
dij(s, 2) = 99                      '仮ラベル
dij(s, 3) = s                       '経過ノード

'●ここまでSTEP1●
'●ここからSTEP2●

labelcount = 0
'pラベルの数がノードの数になったら終了
Do Until (labelcount == 62)

            mindist = 10000
            '最も小さいものを格納
            For i = 0 To 62
            '仮ラベルが付いている●確定ラベルが付いていない
            If (dij(i, 2) == 99) Then
                                    '距離行列から最も小さい値を選択
                                    tmp = dij(i, 1)
                                    If (tmp <= mindist) Then
                                                mindist = tmp
                                                dij(i, 1) = tmp
                                                u = i
                                    End If
            End If
            Next i

'このときのｉ番目ノードに確定ラベルをつける
dij(u, 2) = 999

'カウントを増やす
'条件を満たせばDoを終了する
labelcount = labelcount + 1
If labelcount == 62 Then
break
End If

'●ここまでSTEP2●
'●ここからSTEP3●

'uと接続があるノードについて
For k = 7 To 11
If A(u, k) <> 99 Then
                        '確定ラベルが付いていないものの中で
                        If (dij(A(u, k), 2) <> 999) Then
                        '「枝の走査」
                                    If (dij(A(u, k), 1) > dij(u, 1) + A(u, k + 5)) Then
                                    dij(A(u, k), 1) = dij(u, 1) + A(u, k + 5)
                                    dij(A(u, k), 2) = 99
                                    dij(A(u, k), 3) = u     'どこから来たかを格納
                                    End If
                        End If
End If
Next k

'STEP2に戻る
Loop

'●ここまでSTEP3●


'●●ここから最短経路決定のアルゴリズム●●

'配列の初期化
For i = 1 To 30
            rev(i) = 99
```

```
                fwd(i) = 99
        Next i

        'ゴールからスタートへの経路を格納
        For i = 2 To 30
                rev(1) = g      'ノード番号を格納
                rev(i) = dij(rev(i - 1), 3)

                If rev(i - 1) == s Then
                fwdsize = i - 1
                Break
                End If
        Next i

        '順序を反転させて最短経路決定
        For i = 1 To fwdsize
                fwd(fwdsize + 1 - i) = rev(i)
        Next i

        'FWD1ST に格納
        For i = 1 to 30
                Universe.FWD1ST(i-1) = fwd(i)
        Next i

}

//2 階に関するダイクストラ法計算
Sub routedecide2nd (n as integer, m as integer)
{
        Dim i As Integer
        Dim j As Integer
        Dim u As Integer
        Dim k As Integer
        Dim s As Integer                                    'スタート地点のノード id
        Dim g As Integer                                    'ゴール地点のノード id
        Dim A(30, 16) As Double
        Dim dij(30, 3) As Double
        Dim rev(30) As Integer                              'ゴールからスタートへの順番を格納
        Dim fwd(30) As Integer                              'rev を反転させて最短経路を格納
        Dim fwdsize As integer                              'fwd の大きさとして使用
        Dim tmp As Double
        Dim mindist As Double
        Dim labelcount As Integer

        'A()にデータを格納
        For i = 0 To 29
                A(i, 1) = Universe.FLOOR2ND.POINT.ID(i)
                A(i, 2) = Universe.FLOOR2ND.POINT.X(i)
                A(i, 3) = Universe.FLOOR2ND.POINT.Y(i)
                A(i, 4) = Universe.FLOOR2ND.POINT.Layer(i)
                A(i, 5) = Universe.FLOOR2ND.POINT.Direction(i)
                A(i, 6) = Universe.FLOOR2ND.POINT.POT1(i)
                A(i, 7) = Universe.FLOOR2ND.POINT.n1(i)
                A(i, 8) = Universe.FLOOR2ND.POINT.n2(i)
                A(i, 9) = Universe.FLOOR2ND.POINT.n3(i)
                A(i, 10) = Universe.FLOOR2ND.POINT.n4(i)
                A(i, 11) = Universe.FLOOR2ND.POINT.n5(i)
                A(i, 12) = Universe.FLOOR2ND.POINT.d1(i)
                A(i, 13) = Universe.FLOOR2ND.POINT.d2(i)
                A(i, 14) = Universe.FLOOR2ND.POINT.d3(i)
                A(i, 15) = Universe.FLOOR2ND.POINT.d4(i)
                A(i, 16) = Universe.FLOOR2ND.POINT.d5(i)
        Next i


        '**ここからダイクストラ法のアルゴリズム**

        'スタート地点のノード s とする
        '目的地ノードを g とする
        s = n
        g = m

        '*ここから STEP1*

        'dij(i,1)は計算用の値（ノードにおける仮の距離）を入れる
        'dij(i,2)はフラグを入れる
        'ノードとノードの距離を大数にする
        'dij(i,2)=99 は仮ラベル
        'dij(i,2)=999 は確定ラベル
        'dij(i,3)は経過ノードを格納
        For i = 0 To 29
                dij(i, 1) = 5000
                dij(i, 2) = 0
        Next i

        'スタートノードの設定
                dij(s, 1) = 0           '自分への距離を 0 とする
                dij(s, 2) = 99          '仮ラベル
                dij(s, 3) = s           '経過ノード

        '*ここまで STEP1*
```

```
'●ここからSTEP2●

labelcount = 0
'Pラベルの数がノードの数になったら終了
Do Until (labelcount == 62)

            mindist = 10000
            '最も小さいものを格納
            For i = 0 To 29
                        '仮ラベルが付いている●確定ラベルが付いていない
                        If (dij(i, 2) == 99) Then
                                    '距離行列から最も小さい値を選択
                                    tmp = dij(i, 1)
                                    If (tmp <= mindist) Then
                                                mindist = tmp
                                                dij(i, 1) = tmp
                                                u = i
                                    End If
                        End If
            Next i

'このときのi番目ノードに確定ラベルをつける
dij(u, 2) = 999

'カウントを増やす
'条件を満たせばDoを終了する
labelcount = labelcount + 1
If labelcount == 29 Then
            break
End If

'●ここまでSTEP2●
'●ここからSTEP3●

'uと接続があるノードについて
For k = 7 To 11
            If A(u, k) <> 99 Then
            '確定ラベルが付いていないものの中で
                        If (dij(A(u, k), 2) <> 999) Then
                                    '「枝の走査」
                                    If (dij(A(u, k), 1) > dij(u, 1) + A(u, k + 5)) Then
                                                dij(A(u, k), 1) = dij(u, 1) + A(u, k + 5)
                                                dij(A(u, k), 2) = 99
                                                dij(A(u, k), 3) = u                    'どこから来たかを格納
                                    End If
                        End If
            End If
Next k

'STEP2に戻る
Loop

'●ここまでSTEP3●

'●●ここから最短経路決定のアルゴリズム●●

'配列の初期化
For i = 1 To 30
            rev(i) = 99
            fwd(i) = 99
Next i

'ゴールからスタートへの経路を格納
For i = 2 To 30
            rev(1) = g        'ノード番号を格納
            rev(i) = dij(rev(i - 1), 3)

            If rev(i - 1) == s Then
                        fwdsize = i - 1
                        Break
            End If
Next i

'順序を反転させて最短経路決定
For i = 1 To fwdsize
            fwd(fwdsize + 1 - i) = rev(i)
Next i

'FWD2NDに格納
For i = 1 to 30
            Universe.FWD2ND(i-1) = fwd(i)
Next i

}
```

1階職員エージェントコンポーネント

```
Agt_Init{
            //歩行速度の設定
            MY.V = 3.11 / Universe.GLID * Universe.DELTATIME
            MY.POT1 = 10000
            MY.S_POT1 = 100000
            MY.STATUS = 1
            MY.COLOR = Color_Black

            //2階から降りてきたエージェントの設定
            if (Universe.TIME <> 0 ) then
                        MY.COLOR = Color_Black
            end if

            //初期速度を格納
            MY.S_V = MY.V

}

Agt_Step{

            //最も近いエージェント計算用変数
            Dim dist as double                                          //計算用の仮距離
            Dim tmp as double
            Dim obj as Agt

            //それぞれエージェント集合を格納
            //aroundは(x,y)を中心とした集合を格納
            Dim human as AgtSet
            Dim human2nd as AgtSet
            Dim humanaround as AgtSet
            Dim staff as AgtSet
            Dim staff2nd as AgtSet
            Dim hito as AgtSet
            Dim point as AgtSet
            Dim aroundown as AgtSet
            Dim pointaround as AgtSet
            Dim nearesthumanpointaround as AgtSet

            //接触判定用エージェント集合
            Dim humantouch as AgtSet
            Dim pointtouch as AgtSet

            //最近接 human エージェントを格納
            Dim nearesthuman as Agt
            Dim nearestpoint as Agt
            Dim nearesthumanpoint as Agt

            //ダイクストラ法による経路走査に使用
            Dim fwd(30) as integer
            Dim i as integer
            Dim stairs1dist as double
            Dim stairs2dist as double
            Dim fwdnode as integer

            //自力移動レベル判別用
            Dim moveflaghuman(5) as Integer

            //初期化(右辺はwallだが問題なし)
            nearesthuman = Universe.FLOOR1ST.wall(0)
            nearestpoint = Universe.FLOOR1ST.wall(0)
            nearesthumanpoint = Universe.FLOOR1ST.wall(0)


            //stairs, humanを定義
            MakeAgtSet(human, Universe.FLOOR1ST.HUMAN)
            MakeAgtSet(staff, Universe.FLOOR1ST.STAFF)
            MakeAgtSet(point, Universe.FLOOR1ST.POINT)
            MakeAgtSet(human2nd, Universe.FLOOR2ND.HUMAN)
            MakeAgtSet(staff2nd, Universe.FLOOR2ND.STAFF)

            //humanaroundは自身周辺200のhumanを格納
            MakeAgtSetAroundOwnCell(humanaround, 300, human, false)
            MakeAgtSetAroundOwnCell(pointaround, 30, point, false)


            //STATUS1は目的地設定モード
            if (MY.STATUS ==1) then
                        //歩行速度を初期化
                        MY.V = MY.S_V

                        //避難誘導方針に従う
                        //同じ階にまだフリーな入居者がいるならば
                        if (Universe.FREEHUMAN1ST > 0 ) then
                                    MY.STATUS = 2
                        //もう同じ階にいなければ階段へ向かう
                        elseif (Universe.FREEHUMAN2ND > Universe.FREESTAFF2ND + Universe.TOSTAIRSSTAFF)  then
                                    MY.STATUS = 5
                        else
                                    MY.STATUS = 10
                        end if

            end if
```

```
//STATUS2 はダイクストラ法による経路決定モード
if ( MY.STATUS == 2 ) then

        //自力移動レベル毎のフリーの人数を算出
        for i = 0 to 4
                moveflaghuman(i) = 0
        next i
        for each obj in human
                for i = 0 to 3
                        if obj.MOVEFLAG == i AND obj.STATUS == 99 then
                                moveflaghuman(i) = moveflaghuman(i) + 1
                        elseif obj.MOVEFLAG == 8 then
                                moveflaghuman(4) = moveflaghuman(4) + 1
                        end if
                next i
        next obj

        //自力移動レベルが低い入所者からスキャン
        //自分に近い入所者 nearesthuman を設定する
        for i = 0 to 3
                //階段から降りてきたら
                if moveflaghuman(4) > 0 then
                        for each obj in humanaround
                                //まだ staff に認識されていなければ human の STATUS99
                                if (obj.STATUS == 99 AND obj.MOVEFLAG == 8 ) then
                                        nearesthuman = obj
                                        MY.DIRHUMANID = obj.ID
                                end if
                        next obj
                        break
                //それ以外
                elseif moveflaghuman(i) > 0 then
                        dist = 100000                   //大数に初期化
                        for each obj in humanaround
                                tmp = MeasureDistance(obj.X, obj.Y, MY.X, MY.Y, Universe.FLOOR1ST)
                                //human の STATUS を拾う
                                //まだ staff に認識されていなければ human の STATUS99
                                if (obj.STATUS == 99 AND obj.MOVEFLAG == i ) then
                                        if (tmp < dist) then
                                                dist = tmp
                                                nearesthuman = obj
                                        end if
                                end if
                        next obj
                        break                   //ループから出る
                end if
        next i

        //最近接の human に自身の ID を与える
        nearesthuman.STATUS = MY.ID

        //自分が最も近い POINT を nearestpoint として設定する
        dist = 100000                   //大数に初期化
        for each obj in pointaround
                tmp = MeasureDistance(obj.X, obj.Y, MY.X, MY.Y, Universe.FLOOR1ST)
                if (tmp < dist) then
                        dist = tmp
                        nearestpoint = obj
                end if
        next obj

        //nearesthuman に最も近い POINT を nearesthumanpoint として設定する
        dist = 100000                   //大数に初期化
        MakeAgtSetAroundPositionCell(nearesthumanpointaround, Universe.FLOOR1ST, nearesthuman.X, nearesthuman.Y,
0, 50, point)
        for each obj in nearesthumanpointaround
                tmp = MeasureDistance(obj.X, obj.Y, nearesthuman.X, nearesthuman.Y, Universe.FLOOR1ST)
                if (tmp < dist) then
                        dist = tmp
                        nearesthumanpoint = obj
                end if
        next obj

        //ダイクストラ法による経路設定
        '初期化
        for i = 0 to 29
                fwd(i) = 99
        next i

        //Univese の共通ルール呼び出し（出発ノード，到着ノード）
        @ routedecide1st(nearestpoint.ID, nearesthumanpoint.ID)

        //MINROUTE に格納
        For i = 0 to 29
                MY.MINROUTE(i) = Universe.FWD1ST(i)
        next i

        //ダイクストラによる経路走査終了
        MY.STATUS =3

        //初期値のままだったら全ての human は staff に認識されているため出口へ
```

```
                if ( nearesthuman == Universe.FLOOR1ST.wall(0) ) then
                        MY.STATUS = 1
                end if

        end if

        //STATUS5はダイクストラ法による階段への移動ルート決定モード
        if ( MY.STATUS == 5 ) then
                //階段への距離を算出
                stairs1dist = MeasureDistance(88, 52, MY.X, MY.Y, Universe.FLOOR1ST)
                stairs2dist = MeasureDistance(267, 82, MY.X, MY.Y, Universe.FLOOR1ST)

                //自分が最も近いPOINTをnearestpointとして設定する
                dist = 100000                   //大数に初期化
                for each obj in pointaround
                        tmp = MeasureDistance(obj.X, obj.Y, MY.X, MY.Y, Universe.FLOOR1ST)
                        if (tmp < dist) then
                                dist = tmp
                                nearestpoint = obj
                        end if
                next obj

                //ダイクストラ法による経路設定
                '初期化
                for i = 0 to 29
                        fwd(i) = 99
                next i

                //Univeseの共通ルール呼び出し（出発ノード，到着ノード）
                if (stairs1dist <= stairs2dist AND Universe.TIME < 240.0) then
                        @ routedecide1st(nearestpoint.ID, 21)
                else
                        @ routedecide1st(nearestpoint.ID, 60)
                end if

                //MINROUTEに格納
                For i = 0 to 29
                        MY.MINROUTE(i) = Universe.FWD1ST(i)
                next i

                //ダイクストラによる経路走査終了
                MY.STATUS =6

        end if

        //STATUS3はMINROUTEに沿ってへ入所者へ接近モード
        if (MY.STATUS == 3 ) then
                //歩行速度を初期化
                MY.V = MY.S_V

                //POINTを辿って移動
                for i = 0 to 29
                        MY.FWDNODE = MY.MINROUTE(i)
                        if MY.FWDNODE == 99 then
                                MY.STATUS = 4
                                break
                        elseif MY.FWDNODE <> 80 then
                                MY.DIRX = Universe.FLOOR1ST.POINT(MY.FWDNODE).X
                                MY.DIRY = Universe.FLOOR1ST.POINT(MY.FWDNODE).Y
                                MY.STATUS = 8
                                break
                        else
                                //IDが80ならば次へ
                        end if
                next i

                //同じノードまで近接していたら
                if (MY.FWDNODE == nearesthumanpoint.ID ) then
                        MY.STATUS = 4
                end if

        end if

        //同じノードが設定されていたら直線的に接近
        if (MY.STATUS == 4 ) then
                for each obj in humanaround
                        if (obj.STATUS == MY.ID) then       //humanのFLAGと自身のIDが一致していたらそれを目的地にする
                                MY.DIRX = obj.X
                                MY.DIRY = obj.Y
                        end if
                next obj
                //エラー回避
                //出口まで行ったら目的地設定
                //ENTERANCE1について
                if (1 < MY.X AND MY.X < 4) then
                        if ( 50 <= MY.Y AND MY.Y <= 61 ) then
                                MY.STATUS = 1
                        end if
                //ENTERANCE2について
                elseif (270 < MY.X AND MY.X < 273) then
                        if ( 74 <= MY.Y AND MY.Y <= 81 ) then
                                MY.STATUS = 1
```

```
                    end if
            //SAFETY1について
            elseif (113 < MY.Y AND MY.Y < 116) then
                    if ( 112 <= MY.X AND MY.X <= 256 ) then
                            MY.STATUS = 1
                    end if
            //SAFETY2について
            elseif (40 < MY.X AND MY.X < 43) then
                    if ( 135 <= MY.Y AND MY.Y <= 219 ) then
                            MY.STATUS = 1
                    end if
            end if
end if

//STATUS6は階段へ接近モード
if (MY.STATUS == 6) then
            //歩行速度を初期化
            MY.V = MY.S_V
            //POINTを辿って移動
            for i = 0 to 29
                    MY.FWDNODE = MY.MINROUTE(i)
                    if MY.FWDNODE == 99 then
                            break
                    elseif MY.FWDNODE <> 80 then
                            MY.DIRX = Universe.FLOOR1ST.POINT(MY.FWDNODE).X
                            MY.DIRY = Universe.FLOOR1ST.POINT(MY.FWDNODE).Y
                            MY.STATUS = 88
                            break
                    else
                            //IDが80ならば次へ
                    end if
            next i
            //階段までいったら上る
            if (86 < MY.X AND MY.X < 90) then
                    if ( 49 <= MY.Y AND MY.Y <= 51 ) then
                            stairs1()
                    end if
            elseif (265 < MY.X AND MY.X < 269) then
                    if ( 81 <= MY.Y AND MY.Y <= 83 ) then
                            stairs2()
                    end if
            end if
end if

//---ポテンシャルを用いて出口へ向かう---
if (MY.STATUS == 10) then
            //周辺のポテンシャルを抽出し，最も小さいポテンシャルを方向にする
            dist = 10000
            MakeAllAgtSetAroundOwn(aroundown, 2, true)              //自身周辺2グリッドのAgt全てを格納
            //ポテンシャルによる進行方向決定
            for each obj in aroundown
                    tmp = obj.POT1
                    if (tmp <= dist) then
                            dist = tmp
                            My.DIRX = obj.X
                            My.DIRY = obj.Y
                    end if
            next obj
            //出口まで行ったら消える
            //ENTERANCE1について
            if (1 < MY.X AND MY.X < 4) then
                    if ( 50 <= MY.Y AND MY.Y <= 61 ) then
                            goal()
                    end if
            //ENTERANCE2について
            elseif (270 < MY.X AND MY.X < 273) then
                    if ( 74 <= MY.Y AND MY.Y <= 81 ) then
                            goal()
                    end if
            //SAFETY1について
            elseif (113 < MY.Y AND MY.Y < 116) then
                    if ( 112 <= MY.X AND MY.X <= 256 ) then
                            goal()
                    end if
            //SAFETY2について
            elseif (40 < MY.X AND MY.X < 43) then
                    if ( 135 <= MY.Y AND MY.Y <= 219 ) then
                            goal()
                    end if
            end if
end if

//exitへの角度を計算する
My.Direction = GetDirection(My.X, MY.Y, My.DIRX, MY.DIRY, Universe.FLOOR1ST)

//前進
forward(MY.V)

/*POINTの更新*/
//人に向かっているとき
//POINTに接触したときに更新
if ( MY.STATUS == 8 ) then
```

```
                        //歩行速度を初期化
                        MY.V = MY.S_V

                        MakeAgtSetAroundOwnCell(pointtouch, 1, point, false)
                        //接触したPOINTのIDを獲得
                        dist = 100000                           //大数に初期化
                        for each obj in pointaround
                                tmp = MeasureDistance(obj.X, obj.Y, MY.X, MY.Y, Universe.FLOOR1ST)
                                if (tmp < dist) then
                                        dist = tmp
                                        nearestpoint = obj
                                end if
                        next obj
                        //接触していて，それが目的としているノードならば
                        if (CountAgtSet(pointtouch) > 0 AND MY.FWDNODE == nearestpoint.ID) then
                                for i = 0 to 29
                                        if MY.MINROUTE(i) == MY.FWDNODE then
                                                MY.MINROUTE(i) = 80
                                                MY.STATUS = 3
                                        end if
                                next i
                        end if
                end if

                //階段に向かっているとき
                //POINTに接触したときに更新
                if ( MY.STATUS == 88 ) then
                        //歩行速度を初期化
                        MY.V = MY.S_V

                        MakeAgtSetAroundOwnCell(pointtouch, 1, point, false)
                        //接触したPOINTのIDを獲得
                        dist = 100000                           //大数に初期化
                        for each obj in pointaround
                                tmp = MeasureDistance(obj.X, obj.Y, MY.X, MY.Y, Universe.FLOOR1ST)
                                if (tmp < dist) then
                                        dist = tmp
                                        nearestpoint = obj
                                end if
                        next obj
                        //接触していて，それが目的としているノードならば
                        if (CountAgtSet(pointtouch) > 0 AND MY.FWDNODE == nearestpoint.ID) then
                                for i = 0 to 29
                                        if MY.MINROUTE(i) == MY.FWDNODE then
                                                MY.MINROUTE(i) = 80
                                                MY.STATUS = 6
                                        end if
                                next i
                        end if
                end if

}

Sub stairs1()
{
                Dim new as Agt
                MY.V = 0
                //一定時間経過後を表現
                if (Universe.TIME >= Universe.STAIRS1UPSTAFF + 10.0 / 2 ) then                  //前の人が半分まで降りたら
                        //カウント開始
                        if (MY.STAIRSTIME == 0 ) then
                                MY.STAIRSTIME = Universe.TIME
                        end if
                        if (Universe.TIME >= MY.STAIRSTIME + 10.0 ) then                        //自分が全て降りたら生成
                                Universe.STAIRS1UPSTAFF = Universe.TIME
                                new = CreateAgt(Universe.FLOOR2ND.STAFF)
                                new.X = 104
                                new.Y = 61
                                KillAgt(my)
                        end if
                end if
}

Sub stairs2()
{
                Dim new as Agt
                MY.V = 0
                //一定時間経過後を表現
                if (Universe.TIME >= Universe.STAIRS2UPSTAFF + 10.0 / 2 ) then                  //前の人が半分まで降りたら
                        //カウント開始
                        if (MY.STAIRSTIME == 0 ) then
                                MY.STAIRSTIME = Universe.TIME
                        end if
                        if (Universe.TIME >= MY.STAIRSTIME + 10.0 ) then                        //自分が全て降りたら生成
                                Universe.STAIRS2UPSTAFF = Universe.TIME
                                new = CreateAgt(Universe.FLOOR2ND.STAFF)
                                new.X = 270
                                new.Y = 94
                                KillAgt(my)
                        end if
                end if
}
```

```
Sub goal()
{
        //消えて終了
        KillAgt(my)
}
```

2階職員エージェントコンポーネント

```
Agt_Init{
        //歩行速度の設定
        MY.V = 3.11 / Universe.GLID * Universe.DELTATIME
        MY.POT1 = 10000
        MY.S_POT1 = 100000
        MY.STATUS = 1
        MY.COLOR = Color_Black

        //初期速度を格納
        MY.S_V = MY.V
}

Agt_Step{
        //最も近いエージェント計算用変数
        Dim dist as double                                      //計算用の仮距離
        Dim tmp as double
        Dim obj as Agt

        //それぞれエージェント集合を格納
        //aroundは(x,y)を中心とした集合を格納
        Dim human as AgtSet
        Dim humanaround as AgtSet
        Dim staff as AgtSet
        Dim hito as AgtSet
        Dim point as AgtSet
        Dim aroundown as AgtSet
        Dim pointaround as AgtSet
        Dim staffaround as AgtSet
        Dim nearesthumanpointaround as AgtSet

        //接触判定用エージェント集合
        Dim humantouch as AgtSet
        Dim pointtouch as AgtSet

        //最近接humanエージェントを格納
        Dim nearesthuman as Agt
        Dim nearestpoint as Agt
        Dim nearesthumanpoint as Agt

        //自力移動レベル判別用
        Dim moveflaghuman(5) as Integer

        //ダイクストラ法による経路走査に使用
        Dim fwd(30) as integer
        Dim i as integer

        Dim fwdnode as integer

        //初期化(右辺はwallだが問題なし)
        nearesthuman = Universe.FLOOR2nd.wall(0)
        nearestpoint = Universe.FLOOR2nd.wall(0)
        nearesthumanpoint = Universe.FLOOR2ND.wall(0)

        //stairs, humanを定義
        MakeAgtSet(human, Universe.FLOOR2ND.HUMAN)
        MakeAgtSet(staff, Universe.FLOOR2ND.STAFF)
        MakeAgtSet(point, Universe.FLOOR2ND.POINT)

        //humanaroundは自身周辺200のhumanを格納
        MakeAgtSetAroundOwnCell(humanaround, 300, human, false)
        MakeAgtSetAroundOwnCell(pointaround, 30, point, false)


        //STATUS1は目的地設定モード
        if (MY.STATUS ==1) then
                //歩行速度を初期化
                MY.V = MY.S_V

                //避難誘導方針に従う
                //同じ階にまだ人がいるならば
                if (Universe.FREEHUMAN2ND > 0 ) then
                        MY.STATUS = 2
                //もう同じ階にいなければ
                else
                        MY.STATUS = 10
                end if

        end if

        //STATUS2はダイクストラ法による経路決定モード
        if ( MY.STATUS == 2 ) then
                //自力移動レベル毎のフリーの人数を算出
                for i = 0 to 4
                        moveflaghuman(i) = 0
                next i
                for each obj in human
                        for i = 0 to 3
                                if obj.MOVEFLAG == i AND obj.STATUS == 99 then
                                        moveflaghuman(i) = moveflaghuman(i) + 1
                                elseif obj.MOVEFLAG == 8 then
                                        moveflaghuman(4) = moveflaghuman(4) + 1
```

```
                                        end if
                                next i
                        next obj

                        //自力移動レベルが低い入所者からスキャン
                        //自分に近い入所者 nearesthuman を設定する
                        for i = 0 to 3
                                if moveflaghuman(i) > 0 then
                                        dist = 100000                   //大数に初期化
                                        for each obj in humanaround
                                                tmp = MeasureDistance(obj.X, obj.Y, MY.X, MY.Y, Universe.FLOOR2ND)
                                                //human の STATUS を拾う
                                                //まだ staff に認識されていなければ human の STATUS99
                                                if (obj.STATUS == 99 AND obj.MOVEFLAG == i ) then
                                                        if (tmp < dist) then
                                                                dist = tmp
                                                                nearesthuman = obj
                                                        end if
                                                end if
                                        next obj
                                        break       //ループから出る
                                end if
                        next i

                        //最近接の human に自身の ID を与える
                        nearesthuman.STATUS = MY.ID

                        //自分が最も近い POINT を nearestpoint として設定する
                        dist = 100000                   //大数に初期化
                        for each obj in pointaround
                                tmp = MeasureDistance(obj.X, obj.Y, MY.X, MY.Y, Universe.FLOOR2ND)
                                if (tmp < dist) then
                                        dist = tmp
                                        nearestpoint = obj
                                end if
                        next obj

                        //nearesthuman に最も近い POINT を nearesthumanpoint として設定する
                        dist = 100000                   //大数に初期化
                        MakeAgtSetAroundPositionCell(nearesthumanpointaround, Universe.FLOOR2ND, nearesthuman.X, nearesthuman.Y,
0, 50, point)
                        for each obj in nearesthumanpointaround
                                tmp = MeasureDistance(obj.X, obj.Y, nearesthuman.X, nearesthuman.Y, Universe.FLOOR2ND)
                                if (tmp < dist) then
                                        dist = tmp
                                        nearesthumanpoint = obj
                                end if
                        next obj

                        //ダイクストラ法による経路設定
                        '初期化
                        for i = 0 to 29
                                fwd(i) = 99
                        next i

                        //Univese の共通ルール呼び出し (出発ノード, 到着ノード)
                        @ routedecide2nd(nearestpoint.ID, nearesthumanpoint.ID)

                        //MINROUTE に格納
                        For i = 0 to 29
                                MY.MINROUTE(i) = Universe.FWD2ND(i)
                        next i

                        //ダイクストラによる経路走査終了
                        MY.STATUS =3

                        //初期値のままだったら全ての human は staff に認識されているため出口へ
                        if ( nearesthuman == Universe.FLOOR2ND.wall(0) ) then
                                MY.STATUS = 10
                        end if

        end if

        //STATUS3 は MINROUTE に沿ってへ入所者へ接近モード
        //STATUS6 は階段へ接近モード
        if (MY.STATUS == 3 OR MY.STATUS == 6) then
                //歩行速度を初期化
                MY.V = MY.S_V
                //**ここからが重要**
                //POINT を辿って移動
                for i = 0 to 29
                        MY.FWDNODE = MY.MINROUTE(i)
                        if MY.FWDNODE == 99 then
                                MY.STATUS = 4
                                break
                        elseif MY.FWDNODE <> 80 then
                                MY.DIRX = Universe.FLOOR2ND.POINT(MY.FWDNODE).X
                                MY.DIRY = Universe.FLOOR2ND.POINT(MY.FWDNODE).Y
                                MY.STATUS = 8
                                break
                        else
                                //ID が 80 ならば次へ
```

```
                        end if
                next i

                //同じノードまで近接していたら
                if (MY.FWDNODE == nearesthumanpoint.ID ) then
                        MY.STATUS = 4
                end if
        end if

        //同じノードが設定されていたら直線的に接近
        if (MY.STATUS == 4 ) then
                for each obj in humanaround
                        if (obj.STATUS == MY.ID) then       //human の FLAG と自身の ID が一致していたらそれを目的地にする
                                MY.DIRX = obj.X
                                MY.DIRY = obj.Y
                        end if
                next obj
        end if

        //---ポテンシャルを用いて出口へ向かう---
        if (MY.STATUS == 10) then
                //歩行速度を初期化
                MY.V = MY.S_V
                //周辺のポテンシャルを抽出し，最も小さいポテンシャルを方向にする
                dist = 10000
                MakeAllAgtSetAroundOwn(aroundown, 2, true)      //自身周辺 2 グリッドの Agt 全てを格納
                //ポテンシャルによる進行方向決定
                for each obj in aroundown
                        tmp = obj.POT1
                        if (tmp <= dist) then
                                dist = tmp
                                My.DIRX = obj.X
                                My.DIRY = obj.Y
                        end if
                next obj
        end if

        //階段まで行ったら消える
        //human についているときは human に消される処理をされる
        if (MY.STATUS == 10) then
                //STAIRS1 について
                if (100 < MY.X AND MY.X < 106) then
                        if ( 60 <= MY.Y AND MY.Y <= 63 ) then
                                stairs1()
                        end if
                //STAIRS2 について
                elseif (267 < MY.X AND MY.X < 273) then
                        if ( 92 <= MY.Y AND MY.Y <= 95 ) then
                                stairs2()
                        end if
                end if
        else
                MY.V =MY.S_V
        end if

        //exit への角度を計算する
        My.Direction = GetDirection(My.X, MY.Y, My.DIRX, MY.DIRY, Universe.FLOOR2ND)

        //前進
        forward(MY.V)

        //POINT に接触したときに更新
        if ( MY.STATUS == 8 ) then
                MakeAgtSetAroundOwnCell(pointtouch, 1, point, false)

                //接触した POINT の ID を獲得
                dist = 100000                       //大数に初期化
                for each obj in pointaround
                        tmp = MeasureDistance(obj.X, obj.Y, MY.X, MY.Y, Universe.FLOOR2ND)
                        if (tmp < dist) then
                                dist = tmp
                                nearestpoint = obj
                        end if
                next obj

                //接触していて，それが目的としているノードならば
                if (CountAgtSet(pointtouch) > 0 AND MY.FWDNODE == nearestpoint.ID) then
                        for i = 0 to 29
                                if MY.MINROUTE(i) == MY.FWDNODE then
                                        MY.MINROUTE(i) = 80
                                        MY.STATUS = 3
                                end if
                        next i
                end if
        end if


}

Sub stairs1()
{
        Dim new as Agt
```

```
		MY.V = 0
		//notリレー方式
		//一定時間経過後を表現
		//背負いで降ろすものとする
		if (Universe.TIME >= Universe.STAIRS1TIMEHUMAN + 10.0 / 2 ) then		//前の人が半分まで降りたら
			//カウント開始
			if (MY.STAIRSTIME == 0 ) then
				MY.STAIRSTIME = Universe.TIME
			end if
			if (Universe.TIME >= MY.STAIRSTIME + 10.0 ) then		//自分が全て降りたら生成
				//new = CreateAgt(Universe.FLOOR1ST.STAFF)
				//new.X = 90
				//new.Y = 52
				KillAgt(my)
			end if
		end if

}

Sub stairs2()
{
		Dim new as Agt
		MY.V = 0
		//notリレー方式
		//一定時間経過後を表現
		if (Universe.TIME >= Universe.STAIRS2TIMEHUMAN + 10.0 / 2 ) then		//前の人が半分まで降りたら
			//カウント開始
			if (MY.STAIRSTIME == 0 ) then
				MY.STAIRSTIME = Universe.TIME
			end if
			if (Universe.TIME >= MY.STAIRSTIME + 10.0 ) then		//自分が全て降りたら生成
				//new = CreateAgt(Universe.FLOOR1ST.STAFF)
				//new.X = 269
				//new.Y = 80
				KillAgt(my)
			end if
		end if

}
```

```
Agt_Init{
            //STATUSを設定
            MY.STATUS = 99

            //2階から降りてきたエージェントの設定
            if (Universe.TIME <> 0 ) then
                        MY.POT1 = 10000
                        MY.S_POT1 = 10000
                        MY.MOVEFLAG = Universe.MOVEFLAG
                        if (MY.MOVEFLAG <> 8 ) then
                                    MY.MOVEFLAG = 8
                        end if
            end if

            //歩行速度を設定
            if MY.MOVEFLAG == 0  then
                        MY.V = 0.0 / Universe.GLID * Universe.DELTATIME
                        MY.COLOR = Color_Red
            elseif MY.MOVEFLAG == 1  then
                        MY.V = 0.0 / Universe.GLID * Universe.DELTATIME
                        MY.COLOR = Color_Yellow
            elseif MY.MOVEFLAG == 2  then
                        MY.V = 0.0 / Universe.GLID * Universe.DELTATIME
                        MY.COLOR = Color_Green
            elseif MY.MOVEFLAG == 3  then
                        MY.V = 0.63 / Universe.GLID * Universe.DELTATIME
                        MY.COLOR = Color_Blue
            elseif MY.MOVEFLAG == 8  then
                        MY.V = 0.0 / Universe.GLID * Universe.DELTATIME
                        MY.COLOR = Color_Mazenta
            end if

            //初期速度を格納
            MY.S_V = MY.V

}

Agt_Step{

            //最近接エージェント計算用変数
            Dim dist as double                                          //計算用の仮距離
            Dim tmp as double
            Dim obj as Agt
            Dim tmp2 as double

            //スタッフに関するエージェント集合を定義
            Dim staff as AgtSet
            Dim staffaround as AgtSet
            Dim stafftouch as AgtSet

            //入所者に関するエージェント集合を定義
            Dim human as AgtSet

            //人エージェントを定義
            Dim hito as AgtSet

            //最近接エージェントを格納
            Dim nearestaff as Agt

            //自身の周辺エージェントを格納（人に関するポテンシャル計算用）
            Dim aroundown as AgtSet
            Dim ownpot as AgtSet
            Dim humanpot2 as AgtSet
            Dim humanpot4 as AgtSet
            Dim humanpot6 as AgtSet
            Dim humanpot as AgtSet

            //初期速度を格納
            MY.V = MY.S_V

/*---ポテンシャルを用いて出口へ向かう---*/
            //周辺のポテンシャルを抽出し，最も小さいポテンシャルを方向にする
            dist = 10000
            MakeAllAgtSetAroundOwn(aroundown, 2, true)                  //自身周辺2グリッドのAgt全てを格納
            MY.POT1 = 0

            //ポテンシャルによる進行方向決定
            //目的地のポテンシャルを自身のポテンシャルとする（高速化のため）
            for each obj in aroundown
                        tmp = obj.POT1
                        if (tmp <= dist) then
                                    dist = tmp
                                    My.DIRX = obj.X
                                    My.DIRY = obj.Y
                                    MY.POT1 = obj.POT1 + 10
                        end if
            next obj

            //重ならない処理
            //全てのhumanとstaffをスキャンする
            MakeAgtSet(human, Universe.FLOOR1ST.HUMAN)
            MakeAgtSet(staff, Universe.FLOOR1ST.STAFF)
```

```
        JoinAgtSet(hito, human)
        JoinAgtSet(hito, staff)
        for each obj in hito
                tmp2 = MeasureDistance(obj.X, obj.Y, MY.X, MY.Y, Universe.FLOOR1ST)
                if (tmp2 <= 1.0 / Universe.GLID) then                    //体の中心点と中心点が•m近づいたらの判定
                        if (MY.POT1 > obj.POT1) then                     //ポテンシャルが高いほうが止まる
                                MY.V = 0
                        elseif (MY.POT1 < obj.POT1) then
                                MY.V = MY.V
                        elseif (MY.POT1 == obj.POT1) then                //同じポテンシャルにいるときは移動速度で判断
                                if (MY.V >= obj.V) then
                                        MY.V = MY.V
                                elseif ( MY.V < obj.V) then
                                        MY.V = 0
                                end if
                        end if
                else
                        MY.V = MY.S_V
                end if
        next obj


/*---staffとの対応---*/
        //初期化(右辺はwallだが問題なし)
        nearestStaff = Universe.FLOOR1ST.WALL(0)

        //staffを定義
        MakeAgtSet(staff, Universe.FLOOR1ST.STAFF)
        MakeAgtSetAroundOwnCell(staffaround, 50, staff, false)

        //nearestStaffに最近接のstaffを格納する
        dist = 10000
        for each obj in staffaround
                tmp = MeasureDistance(obj.X, obj.Y, MY.X, MY.Y, Universe.FLOOR1ST)
                if (tmp < dist) then
                        if (MY.STATUS == obj.ID) then
                                dist = tmp
                                nearestStaff = obj
                        end if
                end if
        next obj

        //staff接近時の作業ルーチン
        //stafftouchはstaffと接触の判定用
        MakeAgtSetAroundOwnCell(stafftouch, 2, staff, false)
        if (CountAgtSet(stafftouch) > 0 AND MY.STATUS == nearestStaff.ID) then
                //到着時間を格納する
                if (MY.TIME == 0 ) then
                        MY.TIME = Universe.TIME
                end if

                //staffの作業のために歩行を停止させる
                //nearestStaff.V = 0
                MY.V = 0

                //避難準備作業
                if ( MY.MOVEFLAG == 0) then
                        //一定時間経過後(この場合ベッドのロック解除し移動)
                        if ( Universe.TIME >= MY.TIME + 5.7 ) then
                                MY.V = 1.29 / Universe.GLID * Universe.DELTATIME
                                nearestStaff.V = My.V
                        end if
                elseif ( MY.MOVEFLAG == 1) then
                        //一定時間経過後(この場合ベッドから車椅子へ乗せ換え)
                        if ( Universe.TIME >= MY.TIME + 16.6) then
                                MY.V = 1.72 / Universe.GLID * Universe.DELTATIME
                                nearestStaff.V = My.V
                        end if
                elseif ( MY.MOVEFLAG == 2) then
                        //一定時間経過後(この場合車椅子介助付きへ変更)
                        if ( Universe.TIME >= MY.TIME + 0.0 ) then
                                MY.V = 1.72 / Universe.GLID * Universe.DELTATIME
                                nearestStaff.V = My.V
                        end if
                elseif ( MY.MOVEFLAG == 3) then
                        //一定時間経過後(この場合歩行から背負いへ変更)
                        if ( Universe.TIME >= MY.TIME + 16.1) then
                                MY.V = 1.50 / Universe.GLID * Universe.DELTATIME
                                nearestStaff.V = My.V
                        end if
                elseif ( MY.MOVEFLAG == 8) then
                        //背負いで階段から降りてきて，そのまま移動
                        if ( Universe.TIME >= MY.TIME + 0.0) then
                                MY.V = 1.50 / Universe.GLID * Universe.DELTATIME
                                nearestStaff.V = My.V
                        end if
                end if
        end if

        //exitへの角度を計算する
        My.Direction = GetDirection(My.X, MY.Y, My.DIRX, MY.DIRY, Universe.FLOOR1ST)
```

```
        //前進
        forward(MY.V)

        //出口まで行ったら消える
        //ENTERANCE1 について
        if (1 < MY.X AND MY.X < 4) then
                if ( 50 <= MY.Y AND MY.Y <= 61 ) then
                        goal()
                end if
        //ENTERANCE2 について
        elseif (270 < MY.X AND MY.X < 273) then
                if ( 74 <= MY.Y AND MY.Y <= 81 ) then
                        goal()
                end if
        //SAFETY1 について
        elseif ( 112 <= MY.X AND MY.X <= 256 ) then
                if (113 < MY.Y AND MY.Y < 116) then
                        goal()
                end if
        //SAFETY2 について
        elseif (40 < MY.X AND MY.X < 43) then
                if ( 135 <= MY.Y AND MY.Y <= 219 ) then
                        goal()
                end if
        end if

}


Sub goal()
{
        Dim staff as AgtSet
        Dim tmp as double
        Dim obj as Agt

        MakeAgtSet(staff, Universe.FLOOR1ST.STAFF)

        //初期化(右辺は wall だが問題なし)
        nearestaff = Universe.FLOOR1ST.WALL(0)

        //誘導した STAFF を戻す
        for each obj in staff
                if (MY.STATUS == obj.ID) then
                        obj.STATUS = 1
                end if
        next obj
        KillAgt(my)

}
```

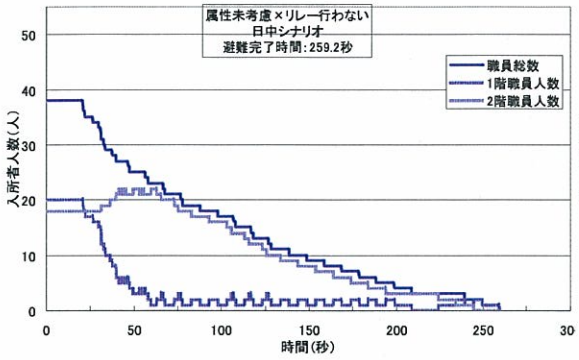
```
Agt_Init{
            //STATUS を設定
            MY.STATUS = 99
            MY.DIJFLAG = 0

            //歩行速度を設定
            if MY.MOVEFLAG == 0  then
                        MY.V = 0.0 / Universe.GLID * Universe.DELTATIME
                        MY.COLOR = Color_Red
            elseif MY.MOVEFLAG == 1  then
                        MY.V = 0.0 / Universe.GLID * Universe.DELTATIME
                        MY.COLOR = Color_Yellow
            elseif MY.MOVEFLAG == 2  then
                        MY.V = 0.0 / Universe.GLID * Universe.DELTATIME
                        MY.COLOR = Color_Green
            elseif MY.MOVEFLAG == 3  then
                        MY.V = 0.63 / Universe.GLID * Universe.DELTATIME
                        MY.COLOR = Color_Blue
            end if

            //初期速度を格納
            MY.S_V = MY.V

}

Agt_Step{

            //最近接エージェント計算用変数
            Dim dist as double                                          //計算用の仮距離
            Dim tmp as double
            Dim obj as Agt
            Dim tmp2 as double

            //スタッフに関するエージェント集合を定義
            Dim staff as AgtSet
            Dim staffaround as AgtSet
            Dim stafftouch as AgtSet

            //入所者に関するエージェント集合を定義
            Dim human as AgtSet

            //最近接エージェントを格納
            Dim nearestaff as Agt

            //自身の周辺エージェントを格納（人に関するポテンシャル計算用）
            Dim aroundown as AgtSet
            Dim ownpot as AgtSet
            Dim humanpot2 as AgtSet
            Dim humanpot4 as AgtSet
            Dim humanpot6 as AgtSet
            Dim humanpot as AgtSet

            //それぞれエージェント集合を格納
            //around は(x,y)を中心とした集合を格納
            Dim humanaround as AgtSet
            Dim hito as AgtSet
            Dim point as AgtSet
            Dim pointaround as AgtSet
            Dim pointtouch as AgtSet
            Dim nearesthumanpointaround as AgtSet

            //最近接 human エージェントを格納
            Dim nearesthuman as Agt
            Dim nearestpoint as Agt
            Dim nearesthumanpoint as Agt

            //ダイクストラ法による経路走査に使用
            Dim fwd(30) as integer
            Dim i as integer
            Dim fwdnode as integer

            MakeAgtSet(point, Universe.FLOOR2ND.POINT)
            MakeAgtSetAroundOwnCell(pointaround, 30, point, false)

            //初期化(右辺は wall だが問題なし)
            nearesthuman = Universe.FLOOR2nd.wall(0)
            nearestpoint = Universe.FLOOR2nd.wall(0)
            nearesthumanpoint = Universe.FLOOR2ND.wall(0)

            /*---ポテンシャルを用いて出口へ向かう---*/
            //周辺のポテンシャルを抽出し，最も小さいポテンシャルを方向にする
            dist = 10000
            MakeAllAgtSetAroundOwn(aroundown, 2, true)                  //自身周辺 2 グリッドの Agt 全てを格納
            MY.POT1 = 0

            //ポテンシャルによる進行方向決定
            //目的地のポテンシャルを自身のポテンシャルとする（高速化のため）
            for each obj in aroundown
                        tmp = obj.POT1
                        if (tmp <= dist) then
                                    dist = tmp
                                    My.DIRX = obj.X
```

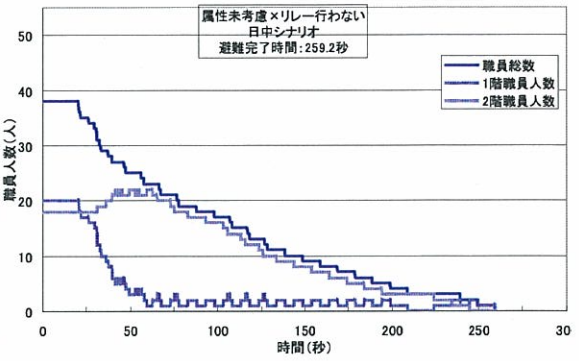
```
                        My.DIRY = obj.Y
                        MY.POT1 = obj.POT1 + 10
            end if
next obj

//4 分経過後はネットワークに沿って stairs2 に向かう
if Universe.TIME > 240 AND MY.DIJFLAG == 0 then
            //自分が最も近い POINT を nearestpoint として設定する
            dist = 100000                       //大数に初期化
            for each obj in pointaround
                        tmp = MeasureDistance(obj.X, obj.Y, MY.X, MY.Y, Universe.FLOOR2ND)
                        if (tmp < dist) then
                                    dist = tmp
                                    nearestpoint = obj
                        end if
            next obj

            //ダイクストラ法による経路設定
            //初期化
            for i = 0 to 29
                        fwd(i) = 99
            next i
            //Univese の共通ルール呼び出し (出発ノード, 到着ノード)
            @ routedecide2nd(nearestpoint.ID, 28)
            //MINROUTE に格納
            For i = 0 to 29
                        MY.MINROUTE(i) = Universe.FWD2ND(i)
            next i

            //4 分経過フラグ
            MY.DIJFLAG = 1
end if

if MY.DIJFLAG == 1 then
            for i = 0 to 29
                        MY.FWDNODE = MY.MINROUTE(i)
                        if MY.FWDNODE == 99 then
                                    break
                        elseif MY.FWDNODE <> 80 then
                                    MY.DIRX = Universe.FLOOR2ND.POINT(MY.FWDNODE).X
                                    MY.DIRY = Universe.FLOOR2ND.POINT(MY.FWDNODE).Y
                                    break
                        end if
            next i
end if

/*---staff との対応---*/
//初期化(右辺は wall だが出題なし)
neareststaff = Universe.FLOOR2ND.WALL(0)

//staff を定義
MakeAgtSet(staff, Universe.FLOOR2ND.STAFF)
MakeAgtSetAroundOwnCell(staffaround, 50, staff, false)

//neareststaff に最近接の staff を格納する
for each obj in staffaround
            tmp = MeasureDistance(obj.X, obj.Y, MY.X, MY.Y, Universe.FLOOR2ND)
            if (tmp < dist) then
                        if (MY.STATUS == obj.ID) then
                                    dist = tmp
                                    neareststaff = obj
                        end if
            end if
next obj

//staff 接近時の作業ルーチン
//stafftouch は staff と接触の判定用
MakeAgtSetAroundOwnCell(stafftouch, 3, staff, false)
if (CountAgtSet(stafftouch) > 0 AND MY.STATUS == neareststaff.ID) then
            //到着時間を格納する
            if (MY.TIME == 0 ) then
                        MY.TIME = Universe.TIME
            end if

            //staff の作業のために歩行を停止させる
            //neareststaff.V = 0
            MY.V = 0

            //一定時間経過後に背負いへ変更
            if ( Universe.TIME >= MY.TIME + 16.1 ) then
                        MY.V = 1.50 / Universe.GLID * Universe.DELTATIME
                        neareststaff.V = My.V
            end if

end if

//職員が付いていて
//階段まで行ったら消える
//STAIRS1 について
if (MY.STATUS <> 99) then
            //STAIRS1 について
            if (100 < MY.X AND MY.X < 106) then
```

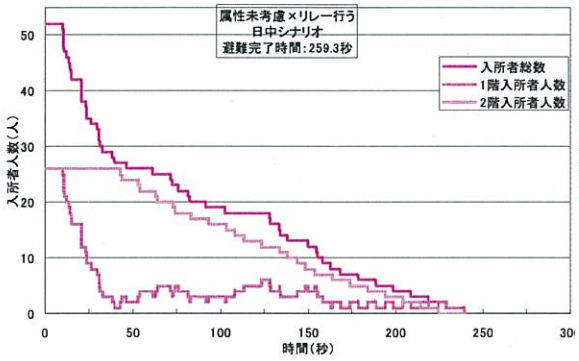
```
                        if ( 60 <= MY.Y AND MY.Y <= 62 ) then
                                stairs1()
                        end if
                //STAIRS2について
                elseif (267 < MY.X AND MY.X < 273) then
                        if ( 93 <= MY.Y AND MY.Y <= 95 ) then
                                stairs2()
                        end if
                end if
        end if

        //角度を計算する
        My.Direction = GetDirection(My.X, MY.Y, My.DIRX, MY.DIRY, Universe.FLOOR2ND)

        //前進
        forward(MY.V)

        //ポイントの更新
        if Universe.TIME > 240 AND MY.DIJFLAG == 1 then
                MakeAgtSetAroundOwnCell(pointtouch, 1, point, false)
                //接触したPOINTのIDを獲得
                dist = 100000                   //大数に初期化
                for each obj in pointaround
                        tmp = MeasureDistance(obj.X, obj.Y, MY.X, MY.Y, Universe.FLOOR2ND)
                        if (tmp < dist) then
                                dist = tmp
                                nearestpoint = obj
                        end if
                next obj

                //接触していて，それが目的としているノードならば
                if (CountAgtSet(pointtouch) > 0 AND MY.FWDNODE == nearestpoint.ID) then
                        for i = 0 to 29
                                if MY.MINROUTE(i) == MY.FWDNODE then
                                        MY.MINROUTE(i) = 80
                                end if
                        next i
                end if
        end if
}

Sub stairs1()
{
        Dim i as Integer
        Dim newhuman as Agt
        Dim newstaff as Agt

        Dim obj as Agt
        Dim staff as AgtSet
        MakeAgtSet(staff, Universe.FLOOR2ND.STAFF)

        //階段前待機時間
        Universe.STAIRSWAITINGTIME = Universe.STAIRSWAITINGTIME + Universe.DELTATIME

        MY.V = 0
        //一定時間経過後を表現
        if (Universe.TIME >= Universe.STAIRS1TIMEHUMAN + 20.0 / 2 ) then                //前の人が半分まで降りたら
                if (MY.STATUS <> 99) then                                               //職員がついていたら
                        //カウント開始
                        if (MY.STAIRSTIME == 0 ) then
                                MY.STAIRSTIME = Universe.TIME
                        end if
                        if (Universe.TIME >= MY.STAIRSTIME + 20.0 ) then        //自分が全て降りたら生成
                                Universe.STAIRS1TIMEHUMAN = Universe.TIME
                                Universe.MOVEFLAG = MY.MOVEFLAG
                                //入所者を生成
                                newhuman = CreateAgt(Universe.FLOOR1ST.HUMAN)
                                newhuman.X = 86
                                newhuman.Y = 52
                                //自身を消す
                                KillAgt(my)
                                //職員について
                                for each obj in staff
                                        if MY.STATUS == 99 then
                                                break
                                        elseif MY.STATUS == obj.ID then
                                                //職員を生成
                                                newstaff = CreateAgt(Universe.FLOOR1ST.STAFF)
                                                newstaff.X = 90
                                                newstaff.Y = 52
                                                //職員を消す
                                                KillAgt(obj)
                                        end if
                                next obj
                        end if
                end if
        end if
}

Sub stairs2()
{
        Dim i as Integer
```

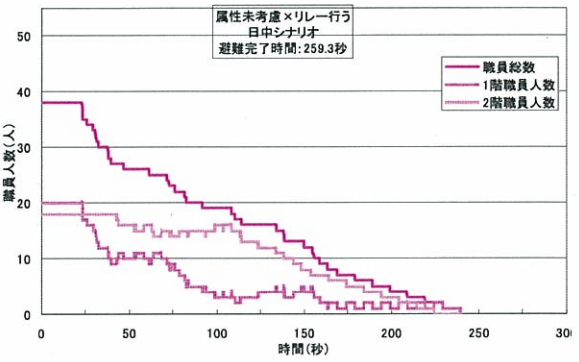
```
        Dim newhuman as Agt
        Dim newstaff as Agt

        Dim obj as Agt
        Dim staff as AgtSet
        MakeAgtSet(staff, Universe.FLOOR2ND.STAFF)

        //階段前待機時間
        Universe.STAIRSWAITINGTIME = Universe.STAIRSWAITINGTIME + Universe.DELTATIME

        MY.V = 0
        //一定時間経過後を表現
        if (Universe.TIME >= Universe.STAIRS2TIMEHUMAN + 20.0 / 2 ) then                    //前の人が半分まで降りたら
                if ( My.STATUS <> 99 ) then                                                 //職員が付いていたら
                //カウント開始
                        if (MY.STAIRSTIME == 0 ) then
                                MY.STAIRSTIME = Universe.TIME
                        end if
                        if (Universe.TIME >= MY.STAIRSTIME + 20.0 ) then                    //自分が全て降りたら生成
                                Universe.STAIRS2TIMEHUMAN = Universe.TIME
                                Universe.MOVEFLAG = MY.MOVEFLAG
                                //入所者を生成
                                newhuman = CreateAgt(Universe.FLOOR1ST.HUMAN)
                                newhuman.X = 265
                                newhuman.Y = 80
                                //自身を消す
                                KillAgt(my)
                                //職員について
                                for each obj in staff
                                        if MY.STATUS == 99 then
                                                break
                                        elseif MY.STATUS == obj.ID then
                                                //職員を生成
                                                newstaff = CreateAgt(Universe.FLOOR1ST.STAFF)
                                                newstaff.X = 269
                                                newstaff.Y = 80
                                                //職員を消す
                                                KillAgt(obj)
                                        end if
                                next obj
                        end if
                end if
        end if

}
```

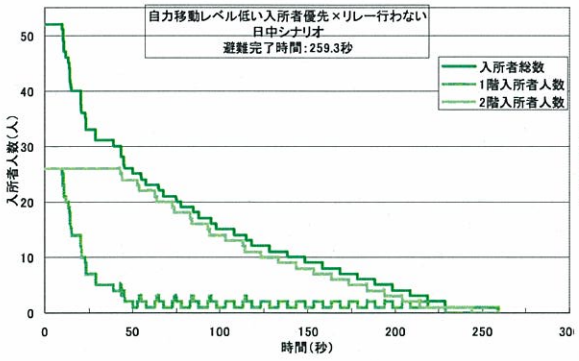
D1_1～D1_4　出力結果（左：入所者人数，右：職員人数）

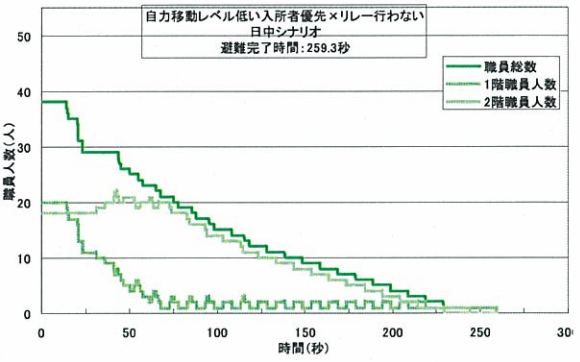


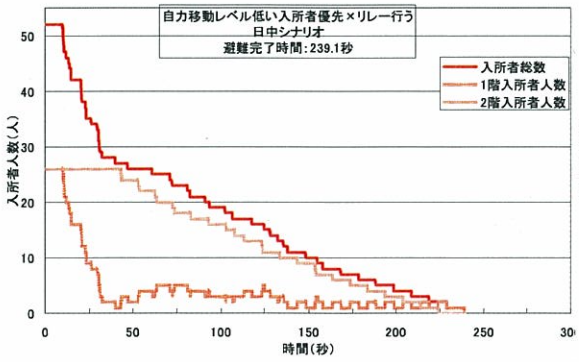


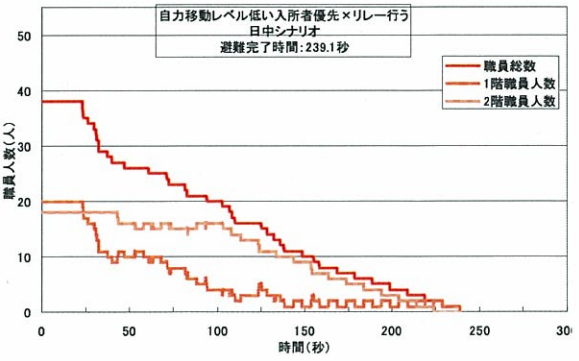




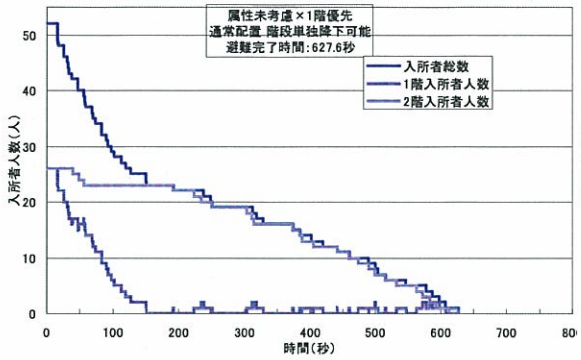


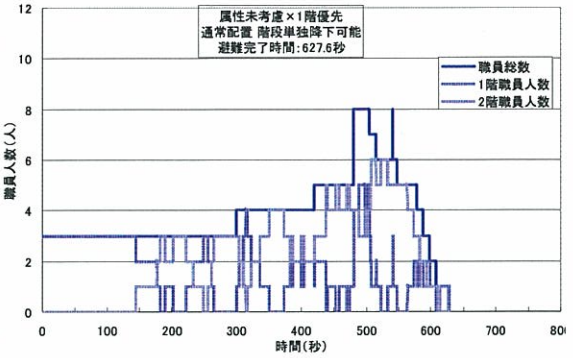


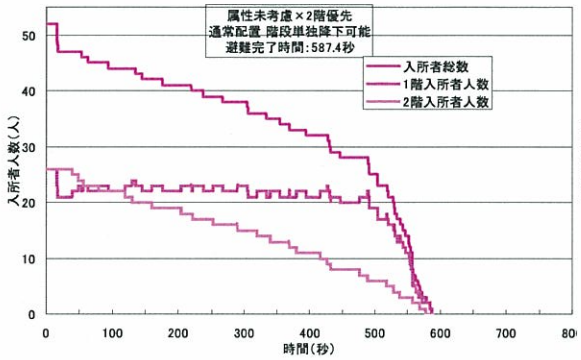


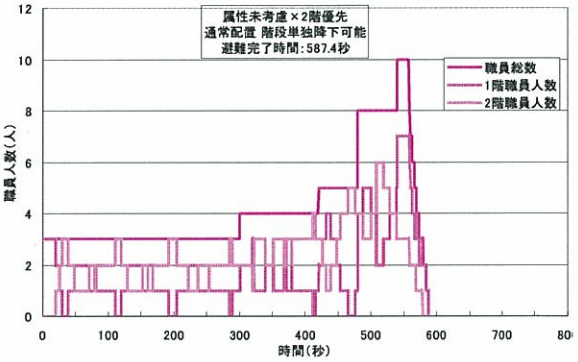


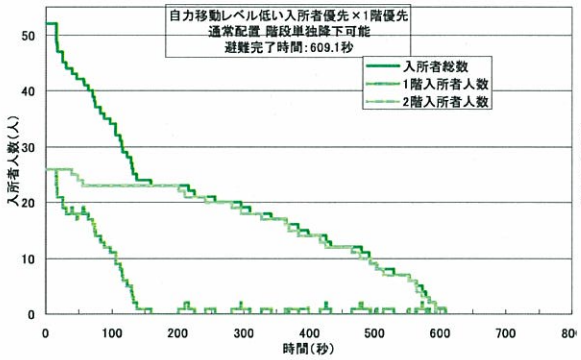
N1_1～N1_4　出力結果（左：入所者人数，右：職員人数）

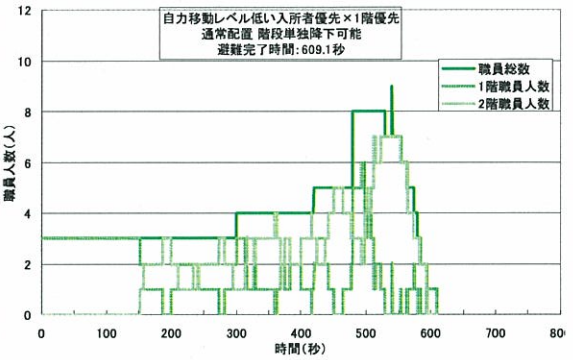


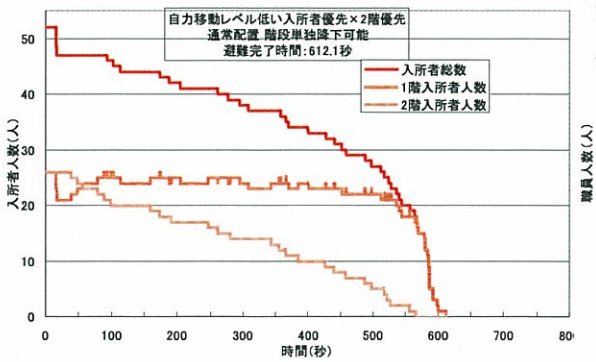


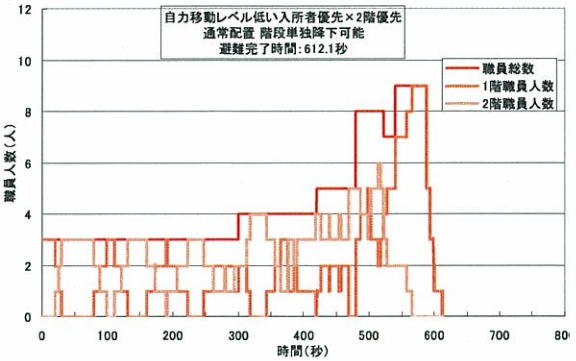




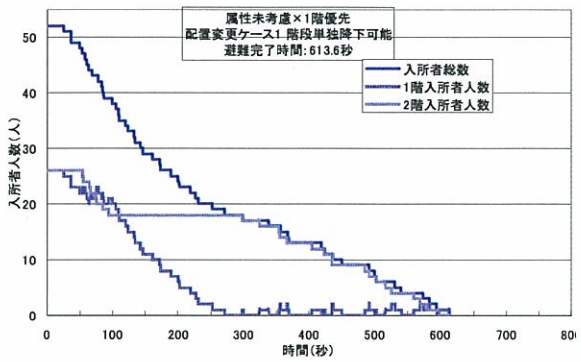


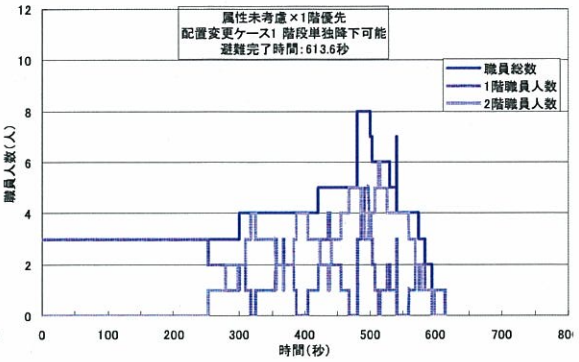


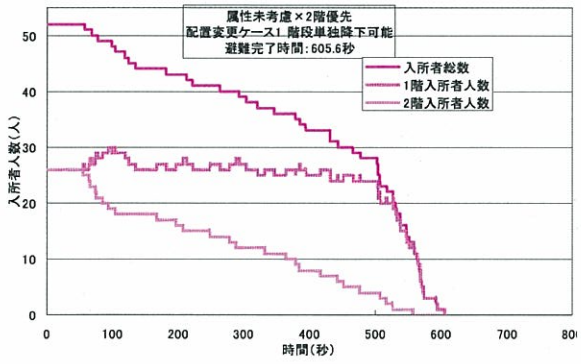


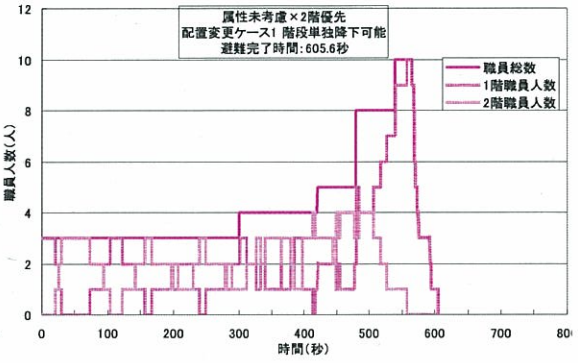


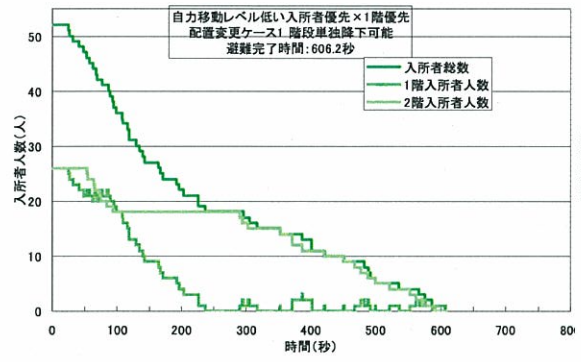
N2_1～N2_4　出力結果（左：入所者人数，右：職員人数）

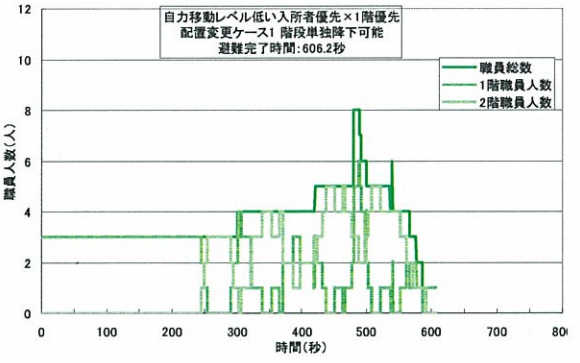


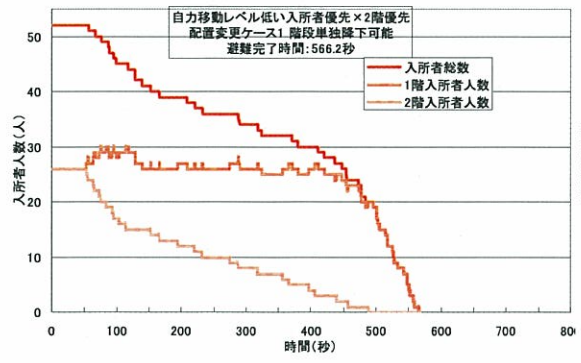


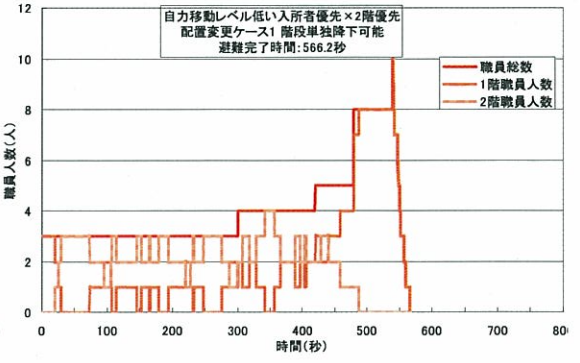












N3_1～N3_4　出力結果（左：入所者人数，右：職員人数）