令和元年度

修士論文

ＢＩＭを用いた
空間シミュレーションの活用に関する研究

指導教員　加藤彰一　教授

三重大学大学院工学研究科建築学専攻
山口恭平

目次

第1章

序論

1－1 研究の背景

1－1－1 BIM の普及

我が国における BIM の普及は、一部の大手建築設計事務所や総合建設会社(以下、ゼネコン)が研究目的・実証実験として扱っていたものが 2009 年の「日本の BIM 元年」を機に中小規模の設計事務所や工務店での利用が急速に増えたことに始まる[「BIM その進化と活用」委員会, 2016]。これにより BIM は、単に 3DCAD の延長ではなく固有のシステムとして位置づけられ、今日では BIM を使って「何をするか」という明確な目的をもって取り組まれている[日本建築学会, 2019]。

近年 ICT はより進化しており、インターネット利用の増大と IoT の普及で様々な人・モノ・組織がネットワークにつながるっている。それに伴い、大量のデジタルデータの生成、収集、蓄積が進みつつある。デジタル技術のもたらす社会像の一つとして Society 5.0 が内閣府により提唱され、より一層社会での情報共有が求められている[総務省, 2018]。特に建設産業では、多様な知識と経験が必要であることから、建築設計・建築施工・構造設計・機械設備設計・電気設備設計などが分業制にて行われており、関係者内のみであっても情報伝達や情報共通を行うことが難しく、齟齬が生まれるといった問題が発生する可能性が高い。ゆえに情報共有の高度化は非常に重要である。

現在、建設業のデジタル技術による変革は進んでいないと考えられているが、BIM や 3D プリンタ、ドローン、ロボット、VR、AR、MR など様々な技術やそれを利用したサービスが生まれつつある。建築におけるデジタル情報は様々であるが、その根幹をなすのが BIM によるものである。BIM はコンピュータ上に作成した 3 次元の建物形状情報に、コストや仕上げ、プロジェクト管理情報などの属性データを追加した建物の統合したデータベースであり、企画・計画・設計・施工・運用・維持管理の一連の建築生産プロセスにおける業務を合理化、効率化する支援ツールとされている[「BIM その進化と活用」委員会, 2016]。設計から施工へと建築が作られていく過程で様々なデジタル情報が作成され、竣工後の FM 段階でも有効に利用できる情報となる。前述したように多くの職能が関わる建築産業では、つくり手だけでなく、つかい手にデジタル情報を共有し一貫して活用できる。このことから建築のデジタル情報化の必要性が高まっているといえる。国土交通省は 2018 年に「BIM/CIM 推進委員会」を設立し、Society 5.0 を見据えた BIM/CIM を総合的に活用する建築・建設生産・管理システムについて議論しており、今後一層建築のデジタル情報化が重要となるだろう。

1－1－2 BIM とプログラミング

建築設計の手法の一つにアルゴリズムを利用しパラメータ化、デジタル化さ

れたデータを活用するアルゴリズミックデザインと呼ばれる設計手法がある。アルゴリズミックデザインでは設計条件が変わってもパラメータの値を変更することで、形を生成することができるため執行錯誤を行いながらの検討を容易に繰り返し行うことができる。2000 年代後半からこの設計手法を用いて作業効率や生産性を向上させる動きが見られるようになった[日本建築学会, 2009]。その中で、プログラミング言語を扱う教育を受けていない設計者が自身でコーディングを行うことなく利用できる VPL が普及しつつある。VPL は従来のテキストプログラミングを記述するのではなく、視覚的なオブジェクトを配置し繋ぎ合わせることでコーディングを行うことのできるプログラミング言語である。

建築設計において利用される VPL は主に Grasshopper と Dynamo の二種類が挙げられる。3D モデリングソフトである Rhinoceros と連動する Grasshopper が 2007 年に、BIM ソフトである Revit と連動する Dynamo が 2014 年に登場した。VPL は理解や習得が容易であることから、テキストプログラミングを扱うことのできない設計者でも比較的容易にコーディングを行うことができる。そのため、大手建築設計事務所やゼネコンを中心に普及が進んでいる。また Grasshopper は従来のテキストプログラミングによる各種プラグインの開発をユーザーが行うことを許容している。その開発環境のオープン性から、開発から 10 年以上が過ぎた現在、ユーザーが開発した様々な拡張機能がオープンソースとして配布され、共有されている[ノイズ・アーキテクツ, 2014]。そのほとんどの開発方法がブラックボックス化されているものの、一般ユーザーは問題なく多くの拡張機能を利用できるほか、開発可能なユーザーが情報交換を行い開発までつなげるサイクルとコミュニティが生まれている。一方、Dynamo は開発されて間もないため、Grasshopper と比較すると拡張機能は少ないがユーザーによる開発のための手引きが公式で配布されているなど開発が推奨されている。また、ユーザー同士が情報交換を行えるコミュニティも公式ページにて用意されており、開発者のための環境は整っている。そのため、発展途上ではあるものの今後急速に普及が進むと考えられる。Dynamo は BIM での利用を前提としているため、BIM のもつ情報をそのまま使用した解析を容易に行うことができる。BIM の持つ多くの情報も相まってシミュレーションから VR、デジタルファブリケーションなどのデジタル機器への活用まで応用性の幅を見せている。Grasshopper も 2016 年に Rhinoceros を経由することで BIM ソフトである ArchiCAD との連携を可能とした。しかし、Dynamo は BIM との情報の相互処理を行うことができるのに対し、BIM 上の建築情報を全て利用することは難しく建物の形状のパラメトリック化などを行うに留まっている。このようにデジタル情報を使って「何をするか」という意識が、利用者や開発者を中心に広まりつつある。

1－1－3　BIM を用いたシミュレーション

1-1-1・1-1-2 では BIM による建築のデジタル情報化やその応用性について述べた。BIM は壁・床などの躯体情報から設備、点景に至るまで「BIM パーツ」と呼ばれるに 3D モデルで構成される。それぞれの BIM パーツに多くのデジタル情報を持たせることで管理されている。図 1-1 に BIM パーツの持つデジタル情報を使用している簡単な例を示す。車体の持つ、幅・長さ・最小回転半径の情報を基に回転の軌跡を描画するプログラムである。車体の寸法情報を変更することで軌跡もそれに従い変更される。BIM を用いたシミュレーションではこのように BIM パーツの持つあらゆるデジタル情報を活用することができる。

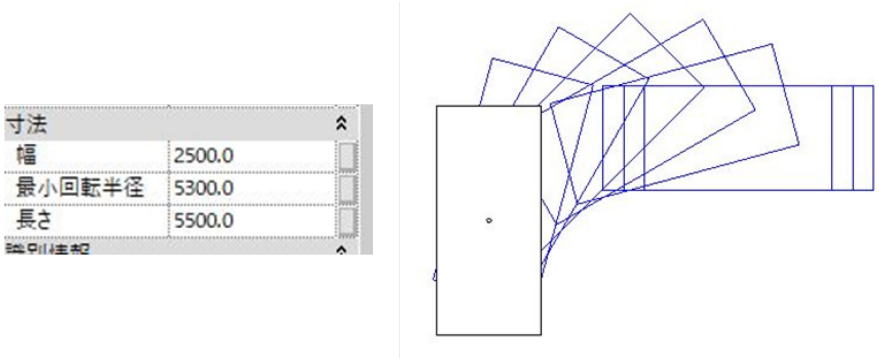


図 1-1　BIM パーツの持つデジタル情報

以下で BIM を用いたシミュレーションの事例について、いくつか取り上げる。

・流体解析[Autodesk, 日付不明]

水族館の水槽の設計に数値流体力学のシミュレーションが用いられた。日本海の海底地形を縮小した複雑な形の水槽に対し、水槽の中を水がよどみなく循環し、かつ魚を来場者に見やすいように分布させるために水流を解析している。海水や砂利などの材料物性、流入出口の流量や季節・時間毎の蒸発潜熱などを

図 1-2　流体解析の出力結果

境界条件としてデジタル情報を活用している。図 1-2 は流体解析後に表示された流跡線と、それを基に 3D プリンタを用いて作成された水流を可視化した模型である。通常は、建屋の設計が固まった上で水槽の設計を行い、最後に解析を行って性能を確認する。しかし、設計前に既に検討は済み、水流という属性情報をフロントローディングで得ることができる。また、同様のモデルを用いて、水温の上昇による温度分布などのシミュレーションも行っている。同様のモデルで様々なシミュレーションを行えることも強みであろう。

・照度解析[前田建設 建築設計統括部門, 日付不明]

オフィスビルの設計において執務空間の照度解析が用いられた。季節や時間、天気といった情報の他、室内空間の仕上げ、什器などの反射率などの情報、室内照明器具のワット数などの情報から照度や輝度をヒートマップや数値データとして得ている。外部自然光と照明器具を合わせたシミュレーションを行うことで、実際の環境を忠実に再現でき、照明器具の効率的な配置や場所ごとに適した種類を選ぶ際のデータとして活用できる。図 1-3 は執務空間のモデルで行われた照度解析である。各時間の室内の外光を含む明るさ感の確認を行っている。

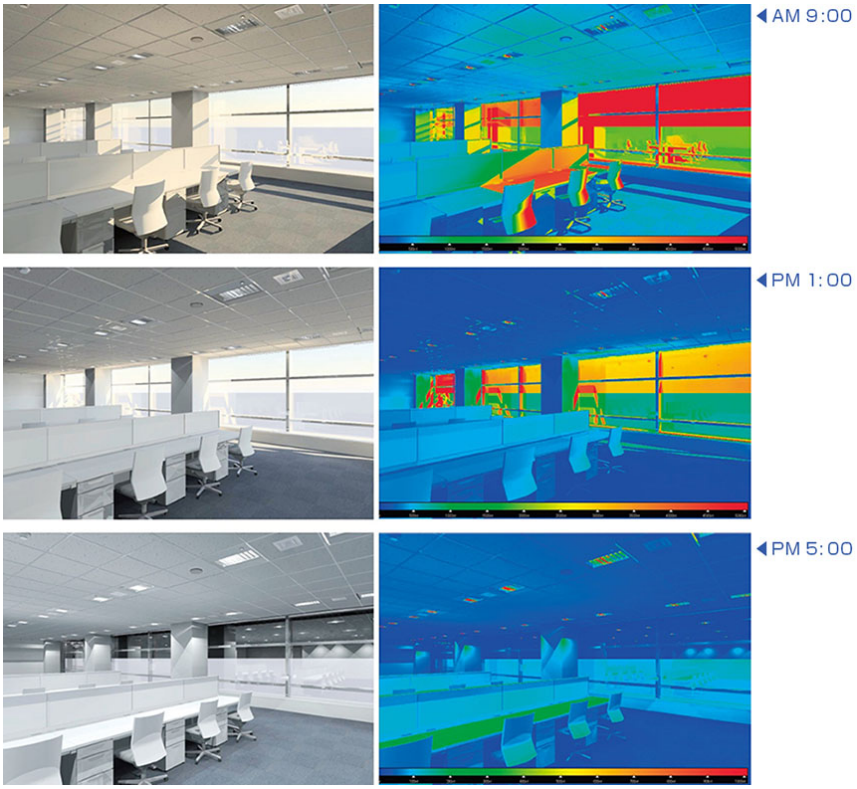


図 1-3　執務空間のモデルで行われた照度解析

・設備設計の自動化[Autodesk, 日付不明]

VPL により、BIM 上の室諸元などの情報を基にした Excel 等の負荷計算連携による空調能力の自動選定プログラムを作成している。従来の設計方法では負荷条件を図面から拾い、負荷計算を行い、空調機器を選定するといった時間と労力を費やしていた作業を自動化することにより短時間での設計変更への対応を可能としている。必要な設備の BIM モデルが作成されていない場合を考慮し、仕様をパラメータとして埋め込んだ「機器記号」に置き換えて定義されている。必要な情報のみを出力することは、シミュレーションの結果の反映において重要な手段である。図 1-4 は BIM と VPL の連携を示したものである。

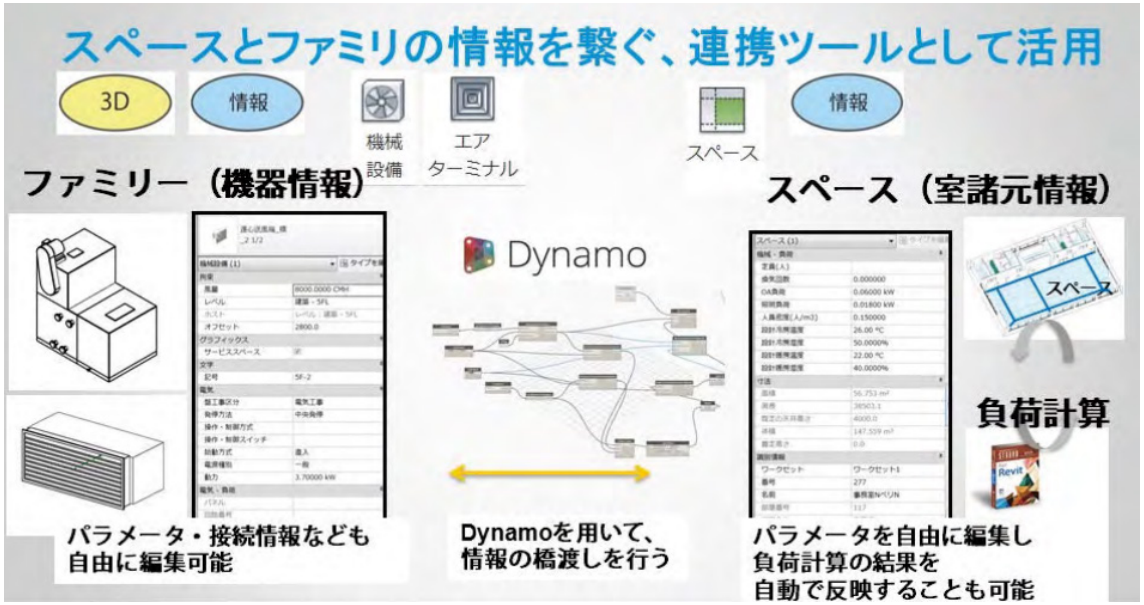


図 1-4　BIM と VPL の連携図

このように、様々な分野でシミュレーションが行われている。

近年、BIM を用いたシミュレーションについての研究も、いくつか報告されている。藤澤らは BIM と GIS を連携することによって、双方の強みを結合し、日照シミュレーションの手法とパラメトリックな都市景観デザインの新たな手法を提案している[藤澤他, 2015]。大西らは応急仮設団地におけるビジュアルシミュレーションに用いる BIM と VR のモデル構築手法の提案や仮設住宅の配置計画案の自動作成手法の提案など一連の研究を行っている[竹澤他, 2017][正宗他, 2019][原他, 2019]。

1－2 研究の目的

BIM の普及により、建築のデジタル情報を活用することで、設計プロセスや運営方法などに大きく変革がもたらされている。本研究では BIM を用いたシミュレーションの開発及び活用方法の提案を行うことで、建築計画分野において BIM を有効に活用するための知見を得ることを目的とし、今後の BIM 活用に関する可能性を検討する。

1－3 論文の構成

本論文は図 1-5 に示す構成とする。

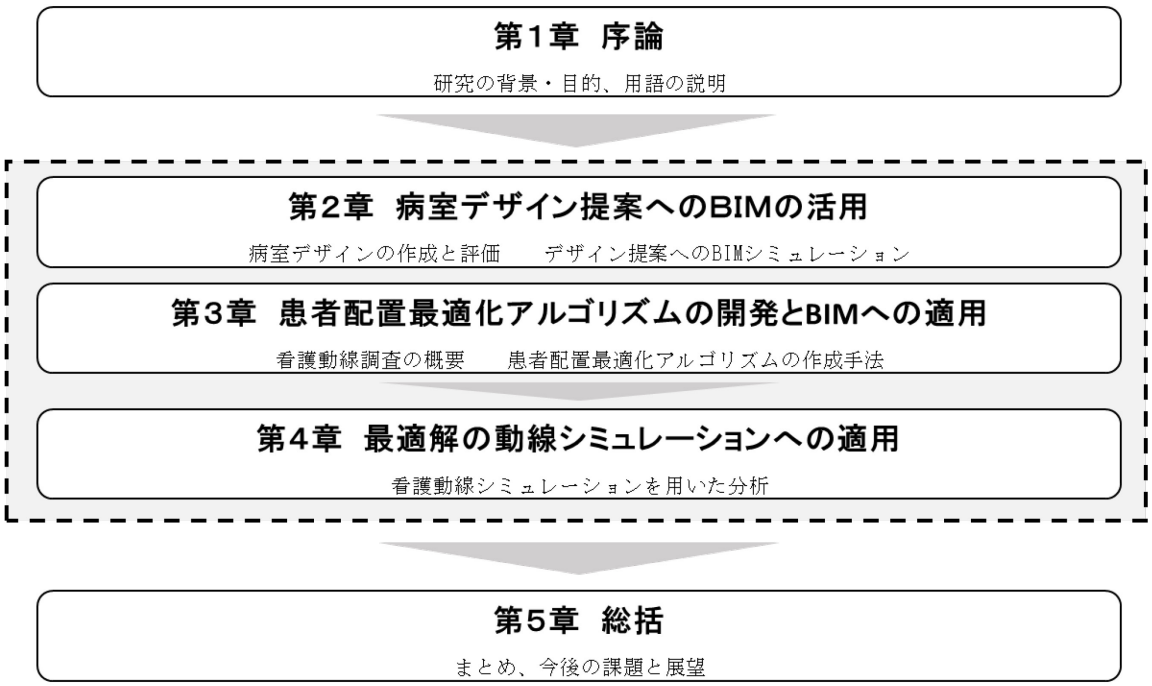


図 1-5　論文構成

第 2 章では病室空間における色彩や設えなどの計画や病室デザインと設計段階の BIM 利用について論じる。第 3 章では病棟内で看護動線量の観点から最適な患者配置を得ることのできるアルゴリズムの作成について論じる。第 4 章では、前章で得た患者配置について BIM を用いたシミュレーションにより看護動線量及び看護効率の有無について論じる。

1－4 研究の方法

以下に、各章で行った研究の方法について概要を記述する。また、詳細な調査方法は各章で記載することとする。

・第 2 章　病室デザイン提案への BIM の活用

病室空間における色彩計画や設えについて、アンケート調査を行い、居住者の心理的評価を把握する。また、簡易的な病室デザインを作成できるアルゴリズムの作成を通し、設計段階の BIM 利用について考察する。

・第 3 章　患者配置最適化アルゴリズムの開発と BIM への適用

病棟内において看護動線量の観点から見て必ずしも適切であるとは限らない患者配置について看護動線量の観点から最適化するシステムを開発し、いくつかの条件における最適な患者配置を把握する。

・第 4 章　最適解の動線シミュレーションへの適用

既に 開発されている看護動線量を把握するシミュレーションを用い実際の看護動線量と最適化した患者配置の看護動線量との比較を行う。

1－5 用語の説明

本研究で取り扱う用語、特にソフトウェアなどを以下で説明する。

・**BIM (Building Information Modeling)**

コンピュータ上に作成した 3 次元の建物形状情報に、コストや仕上げ、プロジェクト管理情報などの属性データを追加した建物の統合したデータベース[「BIM その進化と活用」委員会, 2016]。

・**VPL (Visual Programming Language)**

従来のテキストプログラミングを記述するのではなく、視覚的なオブジェクトを配置し繋ぎ合わせることでコーディングを行うことのできるプログラミング言語。

・**ArchiCAD**

GRAPHISOFT 社が開発した BIM ソフトウェア。

・**Revit**

Autodesk 社が開発した BIM ソフトウェア。

・**Dynamo**

Revit と連動する VPL。

・**Rhinoceros**

Robert McNeel & Associates 社が開発した 3D モデリングに特化した 3 次元 CAD ソフトウェア。

・**Grasshopper**

Rhinoceros と連動する VPL。

・**Python**

汎用のプログラミング言語の一つで、AI や機械学習など広い分野で使用されている。建築分野でも Dynamo、Grasshopper などの VPL に対応している。

・**FM (Facility Management)**

企業・団体などが組織活動のために、施設とその環境を総合的に企画、管理、活用する経営活動のこと[FM 推進連絡協議会, 2018]。

・**PNS (Partnership Nursing System:パートナーシップ・ナーシング・システム)**

2 人の看護師が安全で質の高い看護を共に提供することを目的に、パートナーとして活動し、相互に補完し協力し合って責任を共通する看護方式[福井大学医学部附属病院 看護部, 日付不明]。

・**チームナーシング**

1 病棟で 2 つ以上のチームをつくり、そのチームで一定の患者を受け持つ看護方式。

・**シミュレーション**

コンピュータを用いて模擬的に行う実験。

・**ファミリ**

Revit 内で使用される BIM パーツ。

・**マテリアル**

BIM パーツの外観や材質、断熱などを決定する要素。

第2章

病室デザイン提案へのＢＩＭの活用

2ー1 はじめに

色彩による心理的効果は、建築分野では景観、建物の外装、インテリアなど、その他分野でも自動車、食品のパッケージに至るまで様々な分野で重視されている。入院患者にとって生活空間は極めて重要な要因であり、たとえ入院期間であっても快適に生活できる必要がある。Sara O. Marberry らや Hassam Ghamari ら、Jain Malkin は病院空間において、色彩は患者温健康の回復やメンタルケアに有効な手段としており、米国では病室や待合空間をはじめとする患者が過ごす多くの場所で色彩に富んだ計画が行われている[Marberry et al., 1995][Marberry, 1997][Ghamari et al., 2013][Malkin, 2002]。特に病室は、長期入院患者にとって主要な生活空間となるが、我が国では機能性や清潔感が求められ、白い壁や天井といった色彩や自然環境に乏しい閉鎖的で人工的な空間となる傾向にあった。しかし近年、我が国でも新しい病院の小児病棟や小児専門病院のエントランス、手術室などで色彩計画を含めた子供が居心地よく過ごすことができ、ストレスを軽減するような工夫が行われている。一方で、最も患者の滞在時間が多くなる病室へのアプローチは多くはない。

また、病棟における看護拠点の分散化や分散看護拠点への必要物品の適切配置は、看護動線量の短縮に有効な手段であるとされている。近年、看護師の負担軽減による直接看護の充実を図るために看護物品をさらに分散化し、配置箇所をスタッフステーションの外まで広げている。病棟廊下の壁に設置した収納棚にリネン類や手袋などの使用頻度の高い処置グッズを収納するなどの工夫も行われている。しかし、実際に患者のケアを行う病室内まで入り込み物品配置を行う事例は少ない。

2ー2 研究の方法と目的

まず、米国における病院建築の作品集からいくつかの病室を取り上げ、色彩計画や設えについて分析を行い、その事例を参考に病室パースを作成する。それぞれのサンプルについて SD 法による印象評価を行うことで、我が国の病室空間において従来あまり重要視されてこなかった色彩及び設えなどの計画について居住者の心理的評価を客観的に把握する。これにより病室空間の在り方について計画的知見を得る。その後、病室デザインと設計段階の BIM 利用及びシミュレーションの利用の可能性を探ることを本章の目的とする。

2－3 米国における病室デザインの分析

本章では色の表記はマンセル表色系を用い、素材に木が使用されている場合は色相に(W)を加え表記する。また、床・壁・ベッドなど各要素において割合の多い色を「基調色」、アクセントとなる色を「アクセント色」として表記する。

本節で分析対象とする病室は米国における病院建築の作品集 『Hospital Interior Architecture』『Health Care Architecture』から取り上げたものである[Malkin, 2002][Nesmith, 1995]。

図 2-1 に取り上げた病室を示す。

それぞれ①天井に廻縁を設ける、②壁面収納を設ける、③多彩でモダンな病室１、④多彩でモダンな病室２、⑤壁面収納を設ける(子ども病院)、⑥多彩な子ども病室、として分析を行う。

①天井に廻縁を設ける

二段の木の廻縁を設け、幅広の中間部を高明度の RP 系アクセント色を持つ柄入りクロスとしている。この廻縁にはカーテンケースや間接照明のカバーの役割を持たせている。全体では家具やチェアレールを中明度の YR(W)系とし、壁・床・天井を高明度の YR 系の穏やかなものとして空間全体をまとめ上げている。また、ソファやカーテンを廻縁のアクセント色と同様の色として全体的に温かみのある雰囲気をつくっている。

②壁面収納を設ける

壁・天井及び窓のドレープに高明度の YR 系を、壁面収納仕上げやベッドのヘッド及びフットボードなどにその色に近い、高明度の YR(W)系を使用している。それに対して、壁収納扉のタイル調仕上げや窓辺のソファ、床カーペットなどに低明度の GY 系を対比させることでメリハリのある空間となっている。病床頭部の間接照明やランプシェードが穏やかな空間を演出している。また、病床頭部側に設置された壁面収納には看護用の物品が収められており、飾られた絵画は医療ガスのカバー扉となっている。

③多彩でモダンな病室 1

壁面収納やベッド頭頂部、窓際の壁の一部が高明度・中彩度の BG 系、ベッド背景は中明度・低彩度の RP 系を使用しており、反対色を使いうまく空間をまとめている。ほかの壁や扉、ランプシェードは Y 系、ランプスタンドや机の脚は高彩度の R 系、椅子低明度・低彩度の G 系、カーテンは収納に近い色、多くの小物を置くなど無数に多彩な色が用いられており、楽し気で奥深い環境がつく

られている。

④多彩でモダンな病室２

壁や家具は白色を基調とし椅子や天板、天井面は白木を基調としている。ベッドと対面し患者の目が向けられる壁面は低明度の G 系に Y 系の絵を飾っている。また、ベッドカバーに YR 系、壁面飾りや植木鉢の R 系など多彩な色彩が用いられている。床カーペットは低明度の YR 系を用いるなど全体的にアースカラーでまとめられている。

⑤壁面収納を設ける(子ども病院)

壁や天井は高明度の Y 系を基調色として使用し、床は Y 系、G 系とし類似色でまとめている。ベッドカバーや棚には対称的な B 系を使用し、内装と什器で対比させメリハリのある空間としている。また白の壁面収納の中に時計やテレビ、ビデオデッキなどを組み込み、家型からも子ども病院らしさが現れている。飾り棚には絵本を置いており、引き出しの取手部分などにも多彩な色彩を加えている。

⑥多彩な子ども病室

清潔感のある低明度の B 系をベッド背後の収納扉やカーテン、窓辺の長椅子などに基調色として用い、全体を同系色とし統一感を示しながら、多彩な色彩を細かく使用しているカーテンケースや窓辺の植栽などがアクセント色となり楽し気な色彩を加えている。

⑦その他デザインに関わる要素

取り上げた病室には、花や観葉植物、絵画などを飾ることで空間に多彩な色を加える傾向がある。また、飾り棚やデスクに本を置くことで色を加えている。さらに、ベッドサイドのライトに電球色を使用し Y 系の色を加え、空間に温かみを加えるなどの工夫が随所になされている。

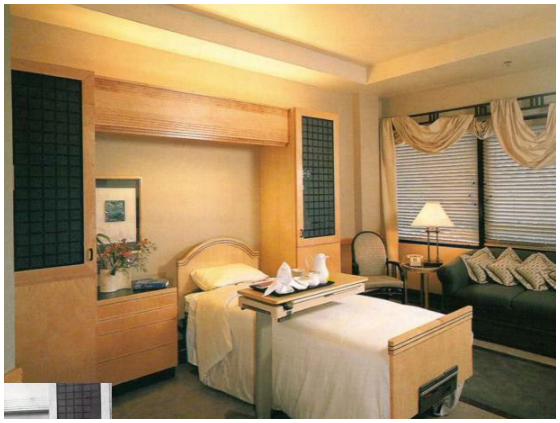
①天井に廻縁を設ける	②壁面収納を設ける
	
③多彩でモダンな病室１	④多彩でモダンな病室２
	
⑤壁面収納を設ける(子ども病院)	⑥多彩な子ども病室
	

図 2-1　対象とした病室写真

2－4 BIM を使用したデザインサンプルの作成

アンケート調査を行うあいち小児保健医療総合センター(以下、「あいち小児センター」)の BIM データを用いて実際の 4 床室の病室デザインを変更することで病室パースを作成する。本調査での被験者への理解を少しでも良くするため、である。

作成したサンプルを図 2-2 に示す。

それぞれ①和室風 1、②和室風 2、③小児病室 1、④デザイナー感覚でモダン、⑤小児病室 2、⑥多彩でモダンとし、日本の伝統的な空間 2 種、小児病室的な空間 2 種、色彩が豊かな空間 2 種をサンプルとする。本章ではそれぞれサンプルを病室①～⑥として表記する。

以下に病室①～⑥の特徴を記す。

・病室①

日本的な空間である数寄屋などの和室を参照し、全体的に明度の高い木を中心に使用している。病床頭部の廻縁は間接照明のカバーとして用いている。

・病室②

日本的な民芸調の和室を参照し、全体的に明度の低い木を中心に使用している。看護側の収納として壁の中央に収納を設けている。

・病室③

米国の小児病院の病室を参照し、白色を基調色とし、明度・彩度ともに高い多数の色彩をランダムに家具や壁に散りばめている。

・病室④

絵画的な色彩を用いたシュレーダー邸を参照し、水平垂直線を意識し、無彩色を基調色としながら、一部の面や家具に原色を使用している。

・病室⑤

米国の小児病院の病室を参照し、高明度の Y 系と G 系を基調色として、アクセント色として B 系を用い、類似の色相を用い構成している。

・病室⑥

米国の病室を参照し、彩度・明度を抑えたアースカラーを基調色として構成している。飾り棚を設け、花を飾る、本を置くなどの色彩を加える工夫にも対応できる。

病室①	病室②
	
病室③	病室④
	
病室⑤	病室⑥
	

図 2-2　作成したサンプル

2－5 病室サンプルの印象評価の把握

2－5－1 調査の概要

本節では、作成したそれぞれのサンプルについて SD 法による印象評価を行う。

2019 年 10 月 13 日にあいち小児センターで行われたイベントにてアンケート調査を行うブースを設置し、イベント来場者にアンケート用紙とサンプルが印刷された用紙を配布し、それぞれの病室の印象について回答を得た。回答を得ることのできた人数は 41 名であり、性別の内訳は男性が 22 名、女性は 19 名であった。また、年齢層の内訳は 21 歳以下が 3 名、20 歳から 40 歳が 20 名、41 歳から 60 歳が 15 名、61 歳以上が 3 名であった（表 2-1)。SD 法に用いる形容詞は図 3 に示す 10 対であり、5 段階評価とした（図 2-3)。

表 2-1 アンケート調査概要

調査日		2019 年 10 月 13 日
場所		あいち小児保健医療総合センター
アンケート対象人数		41 名
性別	男	22 名
	女	19 名
年齢	~20 歳	3 名
	21~40 歳	20 名
	41~60 歳	15 名
	61~歳	3 名

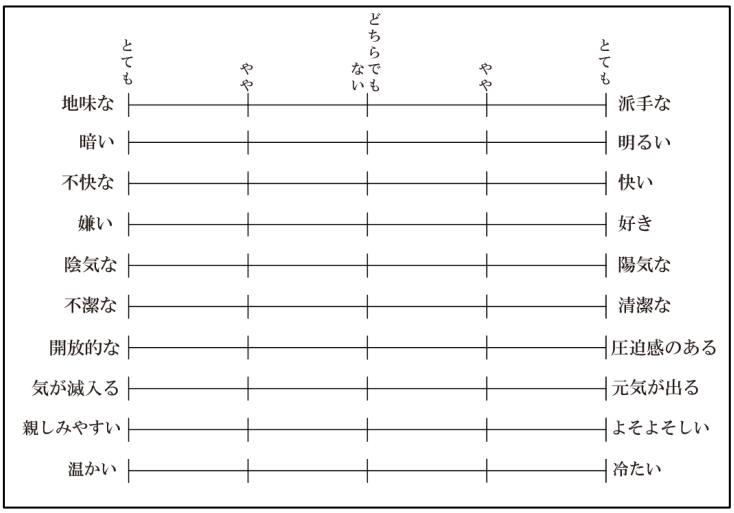


図 2-3 ＳＤ法調査表(一部抜粋)

2ー5ー2 病室プロファイルによる分析

5 段階の尺度の平均値を線で結んだ 6 病室の SD 法プロファイルを表 1 に示す。形容詞対によってはどちらが肯定的でどちらが否定的であるか一概に比較し難いが、本研究では便宜上「地味な―派手な」のような尺度の左の印象を否定的、右の印象を肯定的と表記し、尺度名を表記する場合は肯定的な印象のみを表記することとする。図 2-4 に SD 法のプロファイルを示す。

・病室①

「好き」「親しみやすい」「温かい」などの 6 つの尺度で最高値を取った。これらの尺度は親近的な感覚を得るものが多く選択されていると考えられる。こうした評価傾向には木を基調とした素材感のある日本的空間であることが関係していると考えられる。「派手な」の尺度ではやや否定的な印象に寄っているが全体的に肯定的な印象が多くみられた。

・病室②

全体的に否定的な印象が多くみられた。特に「派手な」「明るい」「陽気な」など 6 つの尺度で最低値を取った。視覚などの五感を刺激するような尺度が選択されていると考えられる。一方で「快い」「好き」など病室①で最高値となった尺度でやや肯定的な印象を与えている。彩度の低い色を基調色としていることで視覚などに対し否定的な印象を与えているが、日本的な空間による肯定的な印象もやや与えていると考えられる。

・病室③

「派手な」「明るい」「陽気な」などの 4 つの尺度で最高値を取った。一方で「快い」「好き」では最低値を取り、「温かい」などもやや否定的な印象を与えている。五感を刺激する尺度は肯定的な印象を与え、親近感を得るものは否定的な印象を与えている。白色を基調色に明度・彩度ともに高い多くの色を用いていることで視覚に対する影響を与えており、それにより冷たい印象を与えていると考えられる。

・病室④

病室③と同じく「派手な」「明るい」などの五感を刺激する尺度は肯定的な印象を与え、「快い」「好き」などの親近感を得るものは否定的な印象を与えており、病室③にかなり近いグラフ型となった。特に、「親しみやすい」の尺度では最低値を取った。黒色が取り入れられ、彩度のそれほど高くない色彩を限定して使用しているため病室③と比較して多くの尺度でやや平均値に寄る結果となったと考えられる。

・病室⑤

病室③・④と同じく「派手な」「明るい」の 2 つの尺度で肯定的な印象を与え

ている。グラフ型も病室③・④に近いが、上記 2 つ以外の尺度はかなり平均値に近い値を取っており、「快い」「好き」などの尺度もやや肯定的な印象を与えている。ここでも多くの色彩を使用しているが類似色相で構成されているため病室③・④と比較して否定的な印象を与えていないと考えられる。

・病室⑥

すべての尺度でやや肯定的な印象を与えており、「好き」「快い」「暖かい」など親近感を与える尺度で全体的にみて高い値を取っている。これはアースカラーを明度・彩度をともに低く抑え、全体的に用いていることから自然的な素材感のある空間であることが関係していると考えられる。

このように、各病室に対する印象は視覚などの五感を刺激しているものと親近的な感覚を得るものに影響を受けていると考えられる。

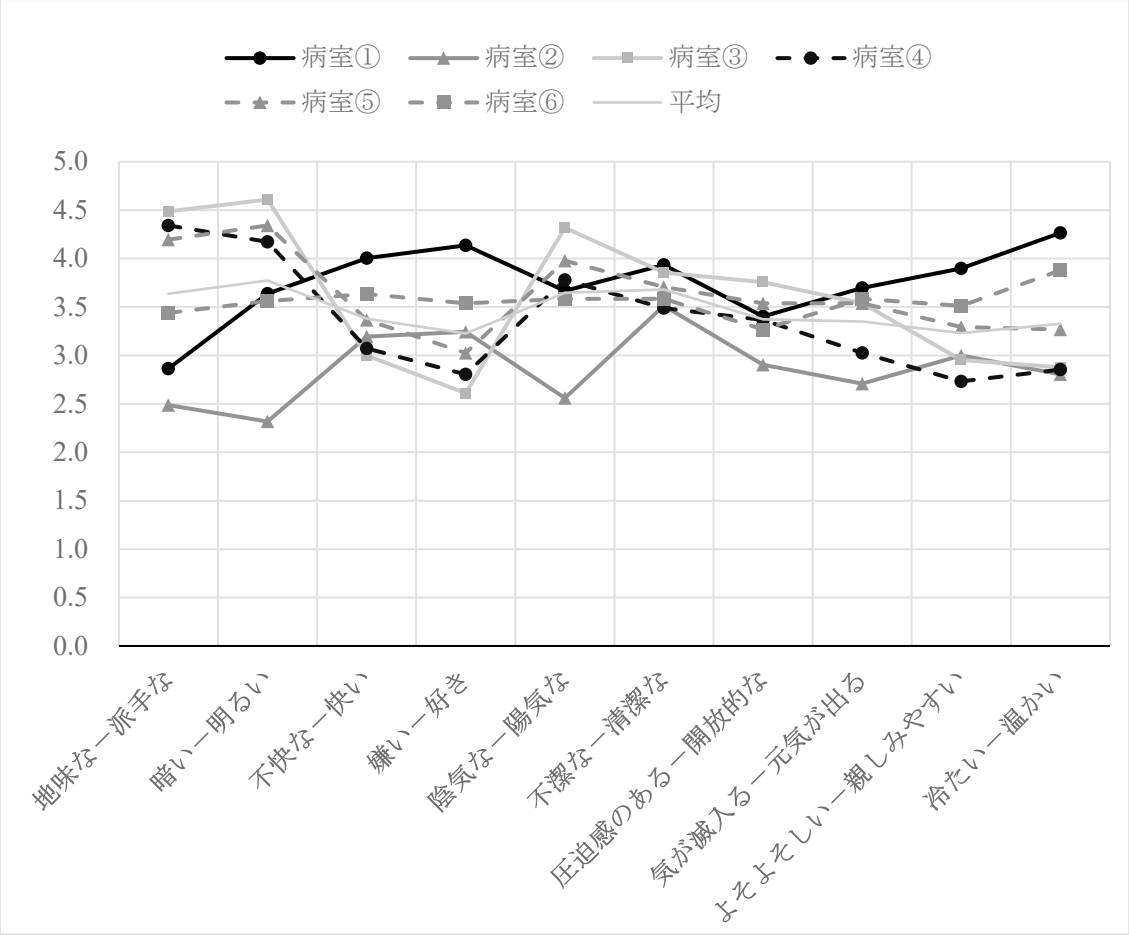


図 2-4　SD 法プロファイル

2－5－3 因子分析による分析

SD 法の結果を用いて因子分析を行った。因子の推定方法は主因子法、回転にはバリマックスを用い固有値 1 以上としたところ 2 つの主要因子が抽出された。絶対値が 0.4 以上の因子負荷量に着目し解釈を行うこととした。因子負荷量を表 2-2 に、各病室における 2 つの因子得点の散布図を図 2-4 に示す。

第 1 因子として抽出された因子は「好き」「快い」「親しみやすい」といった尺度で表される自己の病室空間に対しての親近性と、「暖かい」「元気が出る」「清潔な」といった尺度で表される病室空間の居心地と結びついた尺度から構成されていると考えられる。これは個人の感性により感じる親近感や居心地に関係していると考えられる。また第 2 因子として抽出された因子は「明るい」「派手な」「陽気な」「開放的な」といった尺度で表される視覚など五感で感じる感覚と結びついた尺度から構成されていると考えられる。

各病室の第 1 因子と第 2 因子の因子得点の関係図を見ると、第 1 因子は病室①、病室⑥は因子得点が高く、特に病室①はかなり高い得点となった。この 2 つの病室は日本的な木を基調色とした空間、アースカラーを基調色としアクセント色として木を用いるといった素材感のある空間としている点で共通しており、白色が占める面積も他の病室と比べると少ない。また病室③、病室④は因子得点が低く、病室②、病室⑤もやや低い得点となった。病室③、病室④は基調色の白色を占める面積が多く、さらにアクセント色も明度・彩度の高い多様な色彩を使用していることが共通点として挙げられる。このことから、第 1 因子は素材感や色彩の多様さ・明度・彩度などから影響を受けていると考えられる。

第 2 因子は病室③で因子得点がかなり高くなっており、病室④、病室⑤でやや高い得点となった。この 3 病室は明度･彩度ともに高い多様な色彩を使用し、白色を基調色としている。しかし、同じく多様な色彩を使用している病室⑥は正の因子得点を得ており上記の病室と比較すると多彩ではあるが彩度が低い。また、病室②で因子得点がかなり低くなっており、病室①と病室⑥でやや低い得点となった。病室②は基調色として黒色が多くの面積を占めていること、無彩色以外の色相を使用せず病室全体の彩度が低いことから因子得点が－1.0 以下の非常に低い得点となっていると考えられる。このことから、第 2 因子は使用されている色彩の多様さ・彩度、無彩色が占める面積などから影響を受けていると考えられる。

第 1 因子、第 2 因子両方で正の因子得点を得た病室はなく、病室②が両方で不の因子得点を得た。

表 2-2　因子負荷量

形容詞対	因子１	因子２
嫌い　―　好き	0.863	-0.064
不快な　―　快い	0.844	0.056
冷たい　―　暖かい	0.717	0.080
よそよそしい　―　親しみやすい	0.649	0.060
気が滅入る　―　元気が出る	0.600	0.352
不潔な　―　清潔な	0.511	0.343
暗い　―　明るい	0.061	0.857
地味な　―　派手な	0.600	0.833
陰気な　―　陽気な	0.267	0.797
圧迫感のある　―　開放的な	0.216	0.402

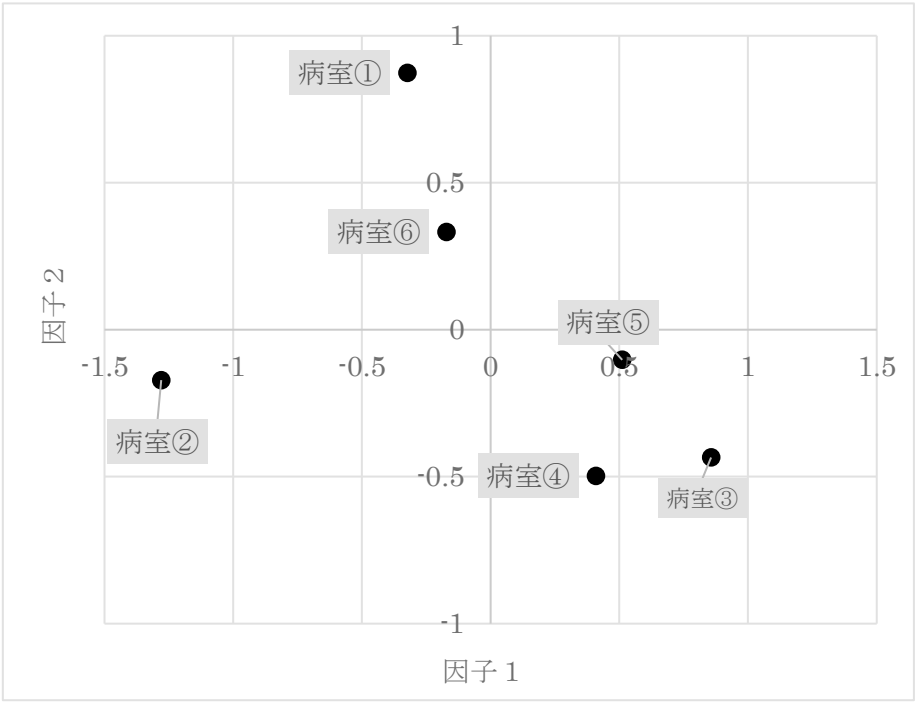


図 2-4　因子得点

2－6 デザイン提案への BIM シミュレーション

2－6－1 シミュレーション開発環境

シミュレーションの作成には、BIM ソフトウェアに Autodesk 社の Revit を使用し、Revit のプラグインソフトである Dynamo をプログラミングソフトとして使用する。Dynamo は BIM における VPL のことである(図 2-5)。また、Dynamo 上にテキストプログラミング言語を入力する際に使用する主要な言語 IronPython2.7 及び Design Script を使用する。また、リアルタイムレンダリングソフトとして LUMION を使用する(表 2-3)。

表 2-3 アルゴリズム作成環境

用途	ソフトウエア名	補足
BIMソフト	Autodesk Revit	
プログラミングソフト	Autodesk Dynamo	Revit上で動作するビジュアルプログラミングソフト
プログラミング言語	IronPython2.7	Dynamoに入力する主要なテキストプログラミング言語
	Design Script	Revitのために開発された言語
	Python3.7	Dynamo使用外で使用するプログラム言語
レンダリングソフト	LUMION	パースを出力するリアルタイムレンダリングソフト

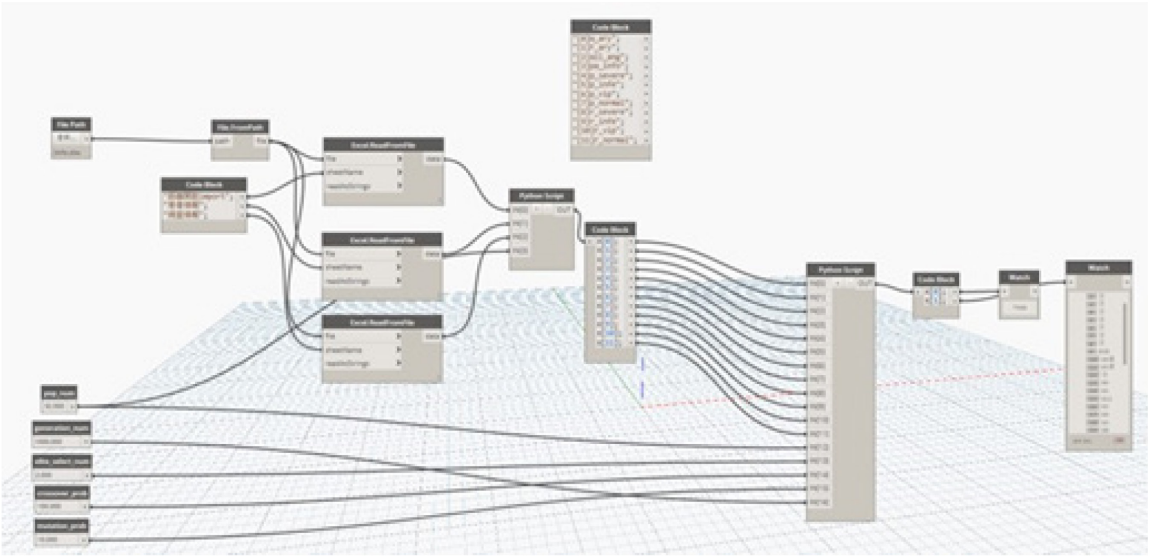


図 2-5 Dynamo 使用画面

2－6－2　作成したデザインシミュレーション

本研究では病室の構成要素である家具や仕上げなどを選択することで簡易的な病室のデザインを作成することのできるアルゴリズムを作成する。図 2-6 に Dynamo で作成したシミュレーションのプログラムを示す。

選択可能である構成要素は、①壁面仕上げ、②床面仕上げ、③幅木、④ベッド間に設ける間仕切り家具や収納棚など及び、その仕上げ、⑤ベッド頭側に設けるパネルなど及び、その仕上げ、⑥窓及び窓枠とその仕上げ、⑦病室入り口のドア及び、その仕上げとしている。

本シミュレーションにより、病室デザインについて容易に把握できるだけでなく、照度や空調など各種シミュレーションを行うモデルを作成する段階を簡略化することができる。これにより、設計者はシームレスに各種シミュレーションへとつなげることができる。複数の提案を容易に行うことができる補助的役割になると考えられる。施主としても提案を直感的に確認できるようになるだけでなく、自ら直感的に病室のデザインを行うことができるため、設計者への提案など意思疎通を図りやすくなると考えられる。

さらに、今後の発展として前述した病室デザインの評価の結果などを通して機械学習をさせていくことが可能であると考えられる。これにより形容詞などを通し自動的に病室のデザインを行い、更なる発展的な使用・提案が可能になるだろう。

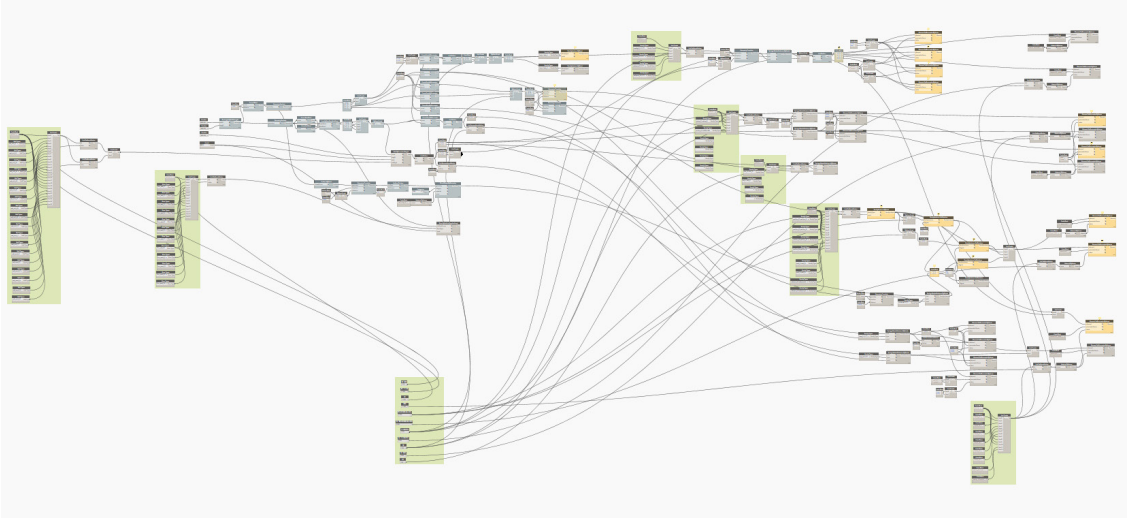


図 2-6　Dynamo で作成したプログラム

図 2-7 に、病室デザインを行った際の Revit への出力例と LUMION への出力例を示す。

Revit に出力の後、LUMION へと出力することで、よりリアルなパースを作成できていることが分かる。また、LUMION を使用することで VR にも対応し、より直感的な提案や意思疎通を行うことができると考えられる。

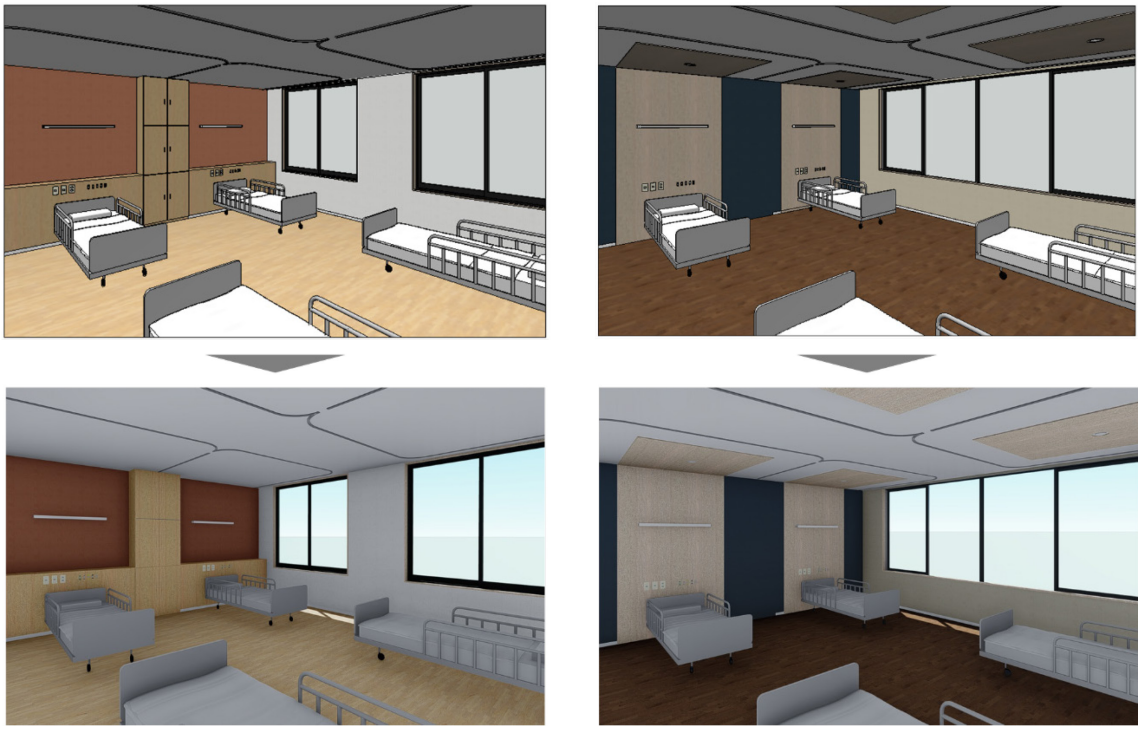


図 2-7　出力例 (上：Revit への出力　下：LUMION への出力)

2-6-2.1　システムフローについて

図 2-8 はデザインシミュレーションの自動化に関するシステムフローを示したものである。システムフローとしては、まず Revit にてシミュレーションへ登録する什器や窓などのファミリ及びマテリアルの作成を行い、Dynamo に登録しプログラムにかける。その後、作成されたモデルを Revit 及び LUMION に出力する。Revit にてファミリの作成を行い Dynamo に登録する点以外は自動化されている。

図 2-8 で示したシステムフローの詳細な内容について記述する。

・R-1　<Revit>ファミリ及びマテリアルの作成、もしくはインポート

まず、シミュレーションに使用する什器などのファミリの作成とマテリアルの作成を行う必要がある。使用する Revit ファイルに手動で作成する。既に作成されているファミリをインポートすることも可能である。マテリアルについても既に作成されているものをインポートできる。

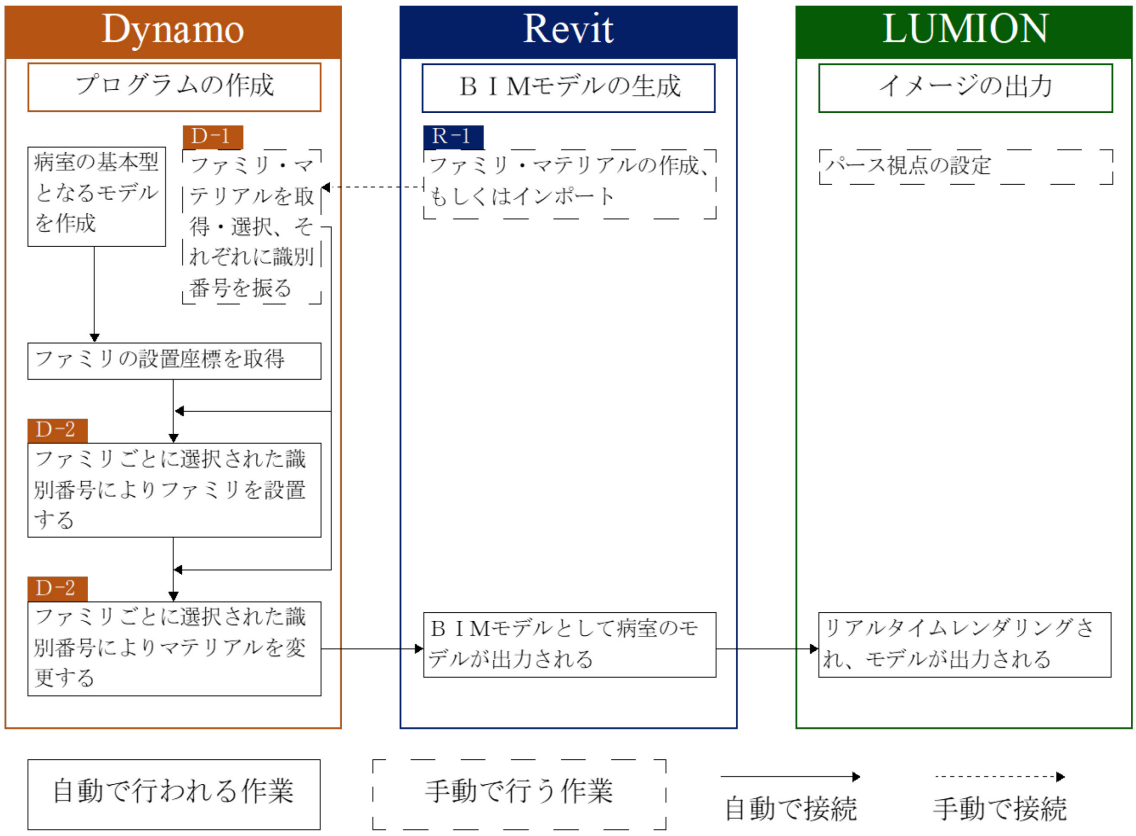


図 2-8　システムフロー

・D-1　<Dynamo>ファミリ・マテリアルを取得・選択、それぞれに識別番号を振る

Revit で作成したファミリ及びマテリアルを Dynamo 上で取得し、選択する。さらに要素ごとに識別番号を振る。この識別番号を選択することで設置する要素を決めて行くことになる。本研究では識別番号を管理するために Excel に表を作成し管理している。この管理表で識別番号を確認し、Dynamo 上で識別番号を選択する。図 2-9 にファミリ及びマテリアルの管理表の例を示す。

番号で管理することで随時ファミリ・マテリアルを作成し、Dynamo 上に登録していくことでシミュレーションに使用できる素材を更新し増やしていくことができる。

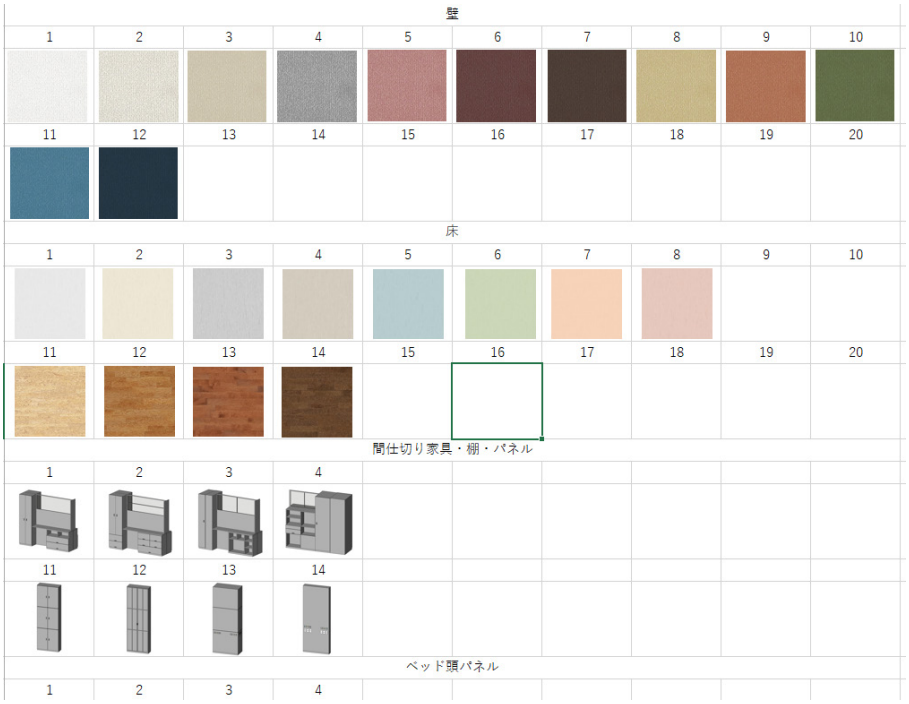


図 2-9　ファミリ及びマテリアルの管理表の例

D-2　<Dynamo>ファミリごとに選択された識別番号によりファミリを設置
ファミリごとに選択された識別番号によりマテリアルを変更

識別番号を選択し、Dynamo 上の入力フォームに入力する。図 2-10 に Dynamo 上の入力フォームを示す。番号を入力するとリアルタイムで選択した素材に変更することができる。

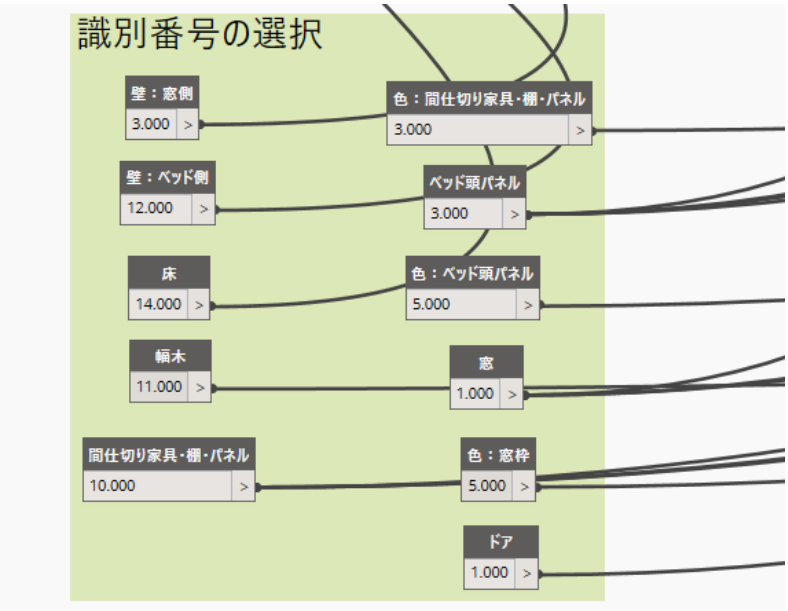


図 2-10　識別番号の入力フォーム

2－7 まとめ

現在、我が国における病室の色彩に富んだ事例や壁面収納などのデザインをもつ事例は少ないため、米国の事例などを分析・参照し、病室パースを作成 6 つの病室サンプルについて SD 法による印象評価を行った。得られた結果の因子分析により①病室空間に対する親近性や居心地からなると解釈できる因子、②五感で感じる感覚と結びついていると解釈できる因子の 2 つの主要因子が抽出された。また各病室の因子得点から第 1 因子は木やアースカラーから得られる素材感や使用する色彩の多様さ・明度・彩度などから影響を受け、第 2 因子は色彩の多様さ・彩度、無彩色が占める面積などから影響を受けていると推測された。

米国では 1970 年ごろから個室化が一般的になり、2006 年には AIA が監修する病院建築設備ガイドライン（Guideline for Design and Construction of Hospital and Health Care Facilities）において、病床は基本的に全室個室化すると定めており、今回分析・参照した事例もほとんどが個室の計画であった。そのため、ソファや棚、デスクなど病室に多彩なデザインが可能であり、絵画を飾る、飾り棚に本を置くなど細かい工夫も可能であると考えられる。我が国においても、患者のプライバシーを重視した個室化の動きが広まっており、今後は個室中心の計画が行われることが予想される。同時に病室の色彩計画や直接看護を増やすための壁面収納するデザイン、間接照明をうまく利用することに関しての様々な検討と提案を行っていくとともに患者・看護師両者からの多様な要求を満たす環境を整える価値はあると考えられる。

第3章

患者配置最適化アルゴリズムの開発と BIM への適用

3－1 はじめに

建築計画の分野において、患者の病床配置や看護師の動線に関する調査が多くなされ、様々な分析やシミュレーションが行われてきた。渡辺らはスタッフステーションから病床までの距離と病室タイプが看護必要度とどのように関連しているか考察を行った[渡辺他, 2009]。谷口らは看護師の動線調査を行ったうえで、患者配置を変更したときの看護動線量の予測や病棟平面形式ごとの看護動線量の予測を行った[谷口他, 1984]。しかし、この患者配置の変更案の作成やシミュレーションは手動で行われている。

このような背景から竹内は BIM 上にて利用可能な看護動線量を把握するシミュレーションを含めた BIM で利用可能なシステムの開発や、病棟の運用実態の評価・分析を行うことで開発したシミュレーションの有用性を明らかにするなどの研究を行った[竹内他, 2020]。しかし、ここでも患者配置の変更案の作成は手動で行われている。

3－2 研究の方法と目的

本研究では病棟内において看護動線量の観点から見て必ずしも適切であるとは限らない患者配置について看護動線量の観点から最適化するシステムを開発し、いくつかの条件における最適な患者配置を把握する。さらに、既に開発されている看護動線量を把握するシミュレーションを用い実際の看護動線量と最適化した患者配置の看護動線量との比較を行うとともにシステムの導入についても次章で考察する。

また、アルゴリズムの作成には前章で用いた開発環境を用いる。

3－3 看護動線調査の概要

調査対象とする病院は、愛知県碧南市の碧南市民病院と岐阜県下呂市の下呂温泉病院の 2 病院である。以下でそれぞれの調査対象病院及び調査対象病棟についての基本情報及び調査日の病棟情報を述べる。

3－3－1 碧南市民病院の概要

碧南市民病院は、愛知県碧南市に位置し三河地方の中核病院として第 2 次救急医療を担っている。表 3-1 は碧南市民病院の基本情報である。1988 年に竣工した後、増改築が行われている。

表 3-1　碧南市民病院基本情報

竣工年	1988 年
敷地面積	50,800 ㎡
延床面積	12,243 ㎡
病床数	320 床(うち地域包括ケア 40 床)
1 床当延床面積	85.3 ㎡
階数	地上 5 階
構造	RC 造
事業者	碧南市
設計者	久米設計 名古屋大学工学部柳沢研究室
施工者	鹿島・白竹・親和建設共同企業体

3-3-1.1　調査対象病棟の概要

調査病棟は、4 階東病棟(以下、東病棟)及び 4 階西病棟(以下、西病棟)である。対象病棟については、以前は病棟平面全体を 1 つの看護単位とみなし、3 つの分散看護拠点を用いた看護を行っていた。しかし、看護体制の変遷などにより現在は東病棟・西病棟の 2 病棟で運営している。それにより東病棟の稼働病床は 36 床、西病棟は 50 床としている。東病棟は東ナースコーナー(以下、NC)を主な看護拠点として、西病棟では中央のスタッフステーション(以下、SS)を主な看護拠点としている。そのため、西 NC は主な看護拠点としては利用されていない。看護方式は東病棟が PNS+チームナーシング、西病棟がチームナーシングとなっており、東病棟では午前に PNS を行い、午後にチームナーシングを行っている。図 3-1 に調査病棟の平面構成を示す。

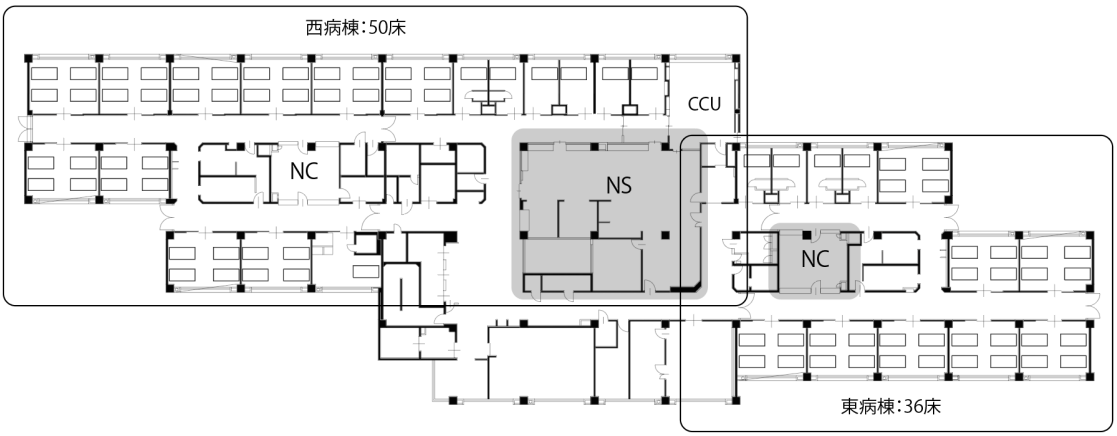


図 3-1　碧南市民病院調査病棟の平面構成

3－3－2　下呂温泉病院の概要

下呂温泉病院は、岐阜県下呂市に位置し、南飛騨地方の中核病院として救急医療を担っている。表 3-2 は下呂温泉病院の基本情報である。2014 年に新築され、全室個室の病棟運営となっている。

表 3-2　下呂温泉病院基本情報

竣工年	2014 年
敷地面積	25,506 ㎡
延床面積	19,605 ㎡
病床数	206 床
1 床当延床面積	95.2 ㎡
階数	地上 5 階
構造	RC 造
事業者	岐阜県立下呂温泉病院
設計者	安井・熊谷設計共同体
施工者	岐建・市川・中島・飛騨特定建設共同企業体

3-3-2.1　調査対象病棟の概要

調査病棟は、5 階東病棟(以下、東病棟)及び 5 階西病棟(以下、西病棟)である。本病院はユニットホール空間を中心に多床室のように個室を並べるユニット型個室と呼ばれる病室を採用し、中央の SS を中心に 8 つのユニットホールを設けている。また東病棟、西病棟ともに 1 看護単位 38 床で、うち重症室が東病棟に 3 床、西病棟に 2 床、各病棟に感染病室 1 床、特別個室 2 床で運営している。ま

た、見守りルームと呼ばれる重症患者や重度の認知症患者を見守るための 4 床室が設けられているが、許可病床ではない。看護方式は 2015 年より東病棟で新看護提供方式 PNS を導入し、2018 年から全病棟で運営されている。図 3-2 に調査病棟の平面構成を示す。

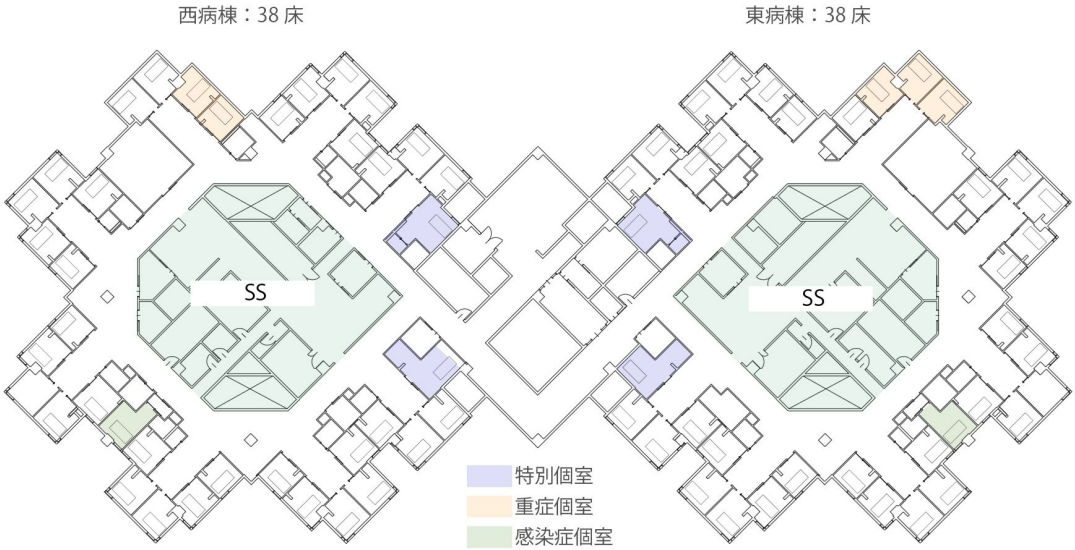


図 3-2　下呂温泉病院調査病棟の平面構成

3－3－3　調査の概要

本研究では両対象病院とも同様の看護動線調査を行う。看護師 1 人に対し調査員 1 人が追跡するタイムスタディ法にて調査を行った。調査員は SS 内、病棟廊下、病室の扉前までの範囲内の動きを追跡し、業務内容・時刻・地点・携帯物等を記録する。記録は看護師の行動を視認し、調査票へ調査員が記入する。看護師看護師が対象病棟外に移動した場合は看護師が戻り次第追跡を再開する。

以下で各病院における調査日病棟の基本情報について述べる。

3-3-3.1　碧南市民病院

調査員は日勤開始(8 時 30 分)から夜勤終了(翌日 9:30)まで追跡調査を行い、日勤看護師(8 時 30 分から 17 時 15 分)2 名、日中看護師(8 時 30 分から 21 時 30 分)2 名、遅出看護師(12 時 45 分から 21 時 30 分)1 名、夜勤看護師(20 時 30 分から 9 時 30 分)2 名を対象とした。また調査当日、西病棟は残業が発生していた。そのため調査時間は、日勤看護師は 8 時 30 分から 18 時 30 分(1 時間 15 分残業)、日勤看護師は 8 時 30 分から 22 時 30 分(1 時間残業)、遅出看護師は 12 時 45 分から 22 時 30 分(1 時間残業)である。表 3-3 に調査日の基本情報を示す。また、以降調査対象看護師を表 3-3 中チーム分け内のシフト+アルファベットで示した看護師名を用い表示する。

表 3-3　碧南市民病院調査日基本情報

<table>
<tr><td>調査対象病院</td><td colspan="4">碧南市民病院</td></tr>
<tr><td rowspan="2">調査対象病棟</td><td colspan="2">4 階東病棟</td><td colspan="2">4 階西病棟</td></tr>
<tr><td colspan="2">血液内科･内分泌系内科･
神経内科</td><td colspan="2">循環器内科･呼吸器内科</td></tr>
<tr><td>稼働病床数</td><td colspan="2">36 床</td><td colspan="2">50 床</td></tr>
<tr><td>調査日</td><td colspan="2">2018.12.05~06</td><td colspan="2">2018.12.12~13</td></tr>
<tr><td>調査日患者数</td><td colspan="2">23/36</td><td colspan="2">43/50</td></tr>
<tr><td>看護方式</td><td colspan="2">PNS+チームナーシング</td><td colspan="2">チームナーシング</td></tr>
<tr><td>看護師勤務シフト</td><td colspan="4">早出　6：30　～　15：15
日勤　8：30　～　17：15
日中　8：30　～　21：30
夜勤　20：30　～　09：30
遅出　12：45　～　21：30</td></tr>
<tr><td>対象看護師</td><td colspan="4">日勤　2 人
日中　2 人
遅出　1 人
夜勤　2 人</td></tr>
<tr><td rowspan="4">調査当日の
チーム分け</td><td rowspan="2">A チーム</td><td>日中F</td><td rowspan="2">A チーム</td><td>日中 F</td></tr>
<tr><td>日勤 E</td><td>日勤 E</td></tr>
<tr><td rowspan="2">B チーム</td><td>日中 D</td><td rowspan="2">B チーム</td><td>日中 D</td></tr>
<tr><td>日勤 C</td><td>日勤 C</td></tr>
</table>

3-3-3.2　下呂温泉病院

調査員は日勤開始（8 時 30 分）から長日勤終了（21 時 30 分）まで追跡調査を行い、日勤帯は日勤看護師（8 時 30 分から 17 時 30 分）2 名、長日勤看護師（8 時 30 分から 21 時 30 分）2 名を対象とした。調査日当日は東病棟の見守りルームに患者はおらず、午後からワックスがけが外注業者により行われていた。一方、西病棟の見守りルームには 2 名の入院患者がいた。見守りルームは重症患者や重度の認知症患者を見守るための 4 床室であり、許可病床ではない。表 3-4 に調査日の基本情報を示す。また、以降調査対象看護師を表 3-4 中チーム分け内のシフト+アルファベットで示した看護師名を用い表示する。

表 3-4　下呂温泉病院調査日基本情報

<table>
<tr><td>調査対象病院</td><td colspan="4">下呂温泉病院</td></tr>
<tr><td rowspan="2">調査対象病棟</td><td colspan="2">5 階西病棟</td><td colspan="2">5 階東病棟</td></tr>
<tr><td colspan="2">脳外科･整形外科･歯科</td><td colspan="2">内科系</td></tr>
<tr><td>稼働病床数</td><td colspan="4">38</td></tr>
<tr><td>調査日</td><td colspan="2">2019.07.26</td><td colspan="2">2019.07.29</td></tr>
<tr><td>調査日患者数</td><td colspan="2">30/38</td><td colspan="2">26/38</td></tr>
<tr><td>看護方式</td><td colspan="4">PNS</td></tr>
<tr><td>看護師勤務シフト</td><td colspan="4">日勤　8：30 ～ 17：30
長日勤　8：30 ～ 21：30
夜勤　20：30 ～ 9：30
遅出　12：30 ～ 20：45</td></tr>
<tr><td>対象看護師</td><td colspan="4">日勤　2 人
長日勤　2 人</td></tr>
<tr><td rowspan="4">調査当日の PNS のチーム分け</td><td rowspan="2">A チーム</td><td>長日勤 B</td><td rowspan="2">B チーム</td><td>長日勤 B</td></tr>
<tr><td>日勤 C</td><td>日勤 D</td></tr>
<tr><td rowspan="2">B チーム</td><td>長日勤 A</td><td rowspan="2">C チーム</td><td>長日勤 A</td></tr>
<tr><td>日勤 D</td><td>日勤 C</td></tr>
</table>

3－4 最適化アルゴリズムの作成手法

本節では調査対象病院対象病棟における患者配置の最適化アルゴリズムについて、必要な入力データ、データ処理のプロセスについて実践例を交えて提案する。また、BIM モデルへの適用における活用方法を考察する。

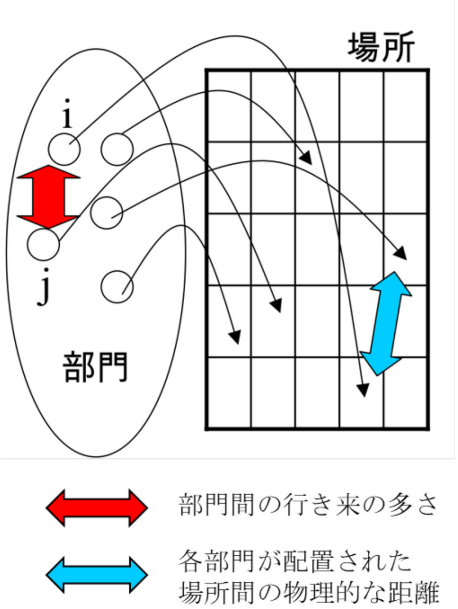
3－4－1 アルゴリズムの概要

本研究ではアルゴリズムを、組み合わせ最適化問題の一つである二次割り当て問題と最適解の解探索手法である遺伝的アルゴリズムを参考に作成し最適化を行う。以下でそれぞれについて概要を述べる。

3-4-1.1 二次割り当て問題

二次割り当て問題は人材や物資、資金などをどのように配置すると最適な配置となるかを見出すことを目的とした組み合わせ最適化の問題である。例として生産工場の新設､改善時に様々な作業部門を配置するという問題がある。n 個の配置箇所と n 個の部門があり、各部門が配置された場所間の物理的な距離を対象行列 W（距離行列）、各部門間のつながりの強さ(行き来の多さなど)の相互関係を対象行列 F（相互関係行列）で与える。各部門を各配置箇所に割り当てたとき各行列の要素を掛け合わせ総和した $C(P)$が最小になるような n 個の順列 P を最適解とする。図 3-3 に二次割り当て問題の概念図を示す。

本研究では部門を入院患者、配置箇所をベッドとみなすことで二次割り当て問題をアルゴリズムに適用する。



部門・場所数：n

距離関数　$\mathrm{W} = [w_{ij}](1 \le i,j \le n)$

相互関係行列　$\mathrm{F} = [f_{ij}](1 \le i,j \le n)$

$$\mathrm{C(P)} = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} f_{\pi(i)\pi(j)}$$

$$\mathrm{P} = (\pi(1), \pi(2), \ldots, \pi(n))$$

C(P)が最小になるような配置を求める

図 3-3　二次割り当て問題概念図[橋本，日付不明]

3-4-1.2　遺伝的アルゴリズム

二次割り当て問題は総当たりで行うと組み合わせ数が膨大になるため、遺伝的アルゴリズム（Genetic Algorithm：以下、GA）を用いる。

GA は生物の進化過程をモデル化した進化的計算法であり、最適化問題に広く応用されている。生物界では生殖過程で染色体の遺伝子を交換し、優良な子孫を生成することにより進化する。一方、 GA では個体（染色体）を文字列（遺伝子）などによりコード化して解を表現する。その個体集団に対して、GA の基本となる選択(selection)、 交叉(crossover)、突然変異(mutation)などの操作を施し、世代交代を繰り返すことにより解の最適化を行う[立命館大学 光情報通信(左貝)研究室, 日付不明]。

建築計画分野においては、青木が総交通コストに基づき室配置を最適化する論文にて、建築プランに GA を適用するにあたり遺伝子列である染色体の定義を提案した[青木, 1996]。その後、村岡らは簡単な形式のプランについて部屋の配置と形状を GA で最適化する手法を提案[村岡他, 1997]、岩田らは病院手術部の廊下パターンと室配置を移動コストから候補解の獲得を試みた[岩田, 1999]。このように GA を平面計画へ応用しようという試みが行われてきている。

以下は GA の処理アルゴリズムにおける用語の説明である。

・選択

GA では単純に最優秀個体のみを保存してしまうと、初期収束に陥る可能性があるため、最優秀解を保存する。その際、解のバリエーションが失われることの無いような選択方法の工夫を行う必要がある。

・交叉

交叉(組み換え)は生物が交配によって子孫を残すことをモデル化したもので、個体の遺伝子の一部を入れ替える操作である。

・突然変異

突然変異は生物に見られる遺伝子の突然変異をモデル化したもので、個体の遺伝子の一部を変化させる操作である。

以下に GA の処理アルゴリズムを示す[数値解析Ⅱ 北海学園大学工学部電子情報工学科，日付不明]。図 3-4 には GA の概念図を示す。

1. 現世代に N 個の個体をランダムに生成する。
2. 評価関数により、現世代の各個体の適応度をそれぞれ計算する。
3. 適応度の低い個体を廃棄し、優秀な個体に次の処理を行う。
4. ある確率で次の 3 つの動作のいずれかを行い、その結果を次世代に保存する。

   現世代の優秀個体も保存しておく。

   i. 個体を二つ選択して交叉を行う。

   ii. 個体を一つ選択して突然変異を行う。

   iii. 個体を一つ選択してそのままコピーする。

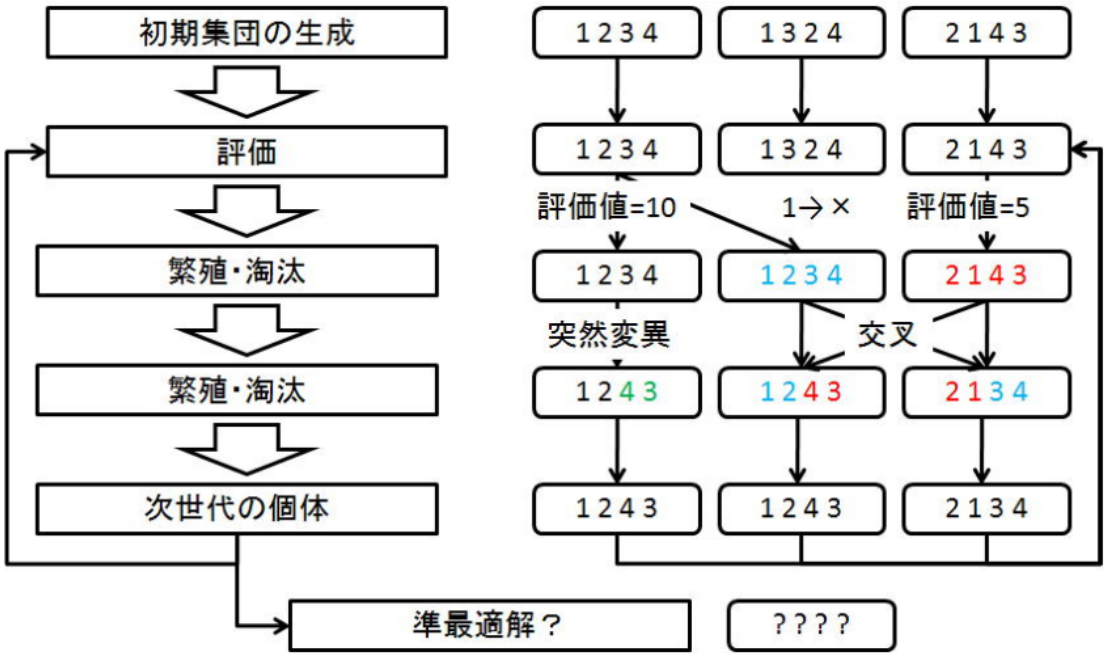


図 3-4　遺伝的アルゴリズム概念図

[数値解析Ⅱ 北海学園大学工学部電子情報工学科，日付不明]

5. 次世代の個体数が N 個になるまで上記の動作を繰り返す。
6. 次世代の個体数が N 個になったら次世代の内容を全て現世代に移す。
7. 3. 以降の動作を最大世代数 G 回まで繰り返し、最終的に「現世代」の中で最も適応度の高い個体を「解」として出力する。

3－4－2 アルゴリズムへの適用手法

次に、前述した二次割り当て問題と遺伝的アルゴリズムについて、患者配置の最適化アルゴリズムへの適用手法について述べる。開発するシステムは n 人の患者(配置されたベッドが空ベッドになる仮患者を含む)を n 個のベッドに配置する二次割り当て問題を遺伝的アルゴリズムで最適化するものとする。また、病棟において患者配置を行う上では患者の看護必要度や病室間の距離だけでなく、病室の SS からの距離も考慮する必要がある。そのため、距離行列、相互関係行列に加え患者の看護必要度、病室の SS からの距離も評価値に加えることとする。

適用する病棟は調査対象病棟である碧南市民病院の東病棟及び西病棟、下呂温泉病院の東病棟及び西病棟である。二次割り当て問題における距離行列は病室（ベッド）間の距離、相互関係行列は看護師チームの受け持ち患者から移動頻度を仮定し、移動頻度が高いほど数値が大きくなるように表記する。碧南市民病院は主に 4 床室で構成しているのに対し、下呂温泉病院は全室個室の病棟となっている。全室個室の場合は同室となる患者の条件を考える必要はないが、4 床室などで最適化を行う場合は男女が同じ室に入室しないようにするなど、必要な評価軸が増加する。本研究では両病院ともにアルゴリズムを作成するが、両病院以外の病棟でも適用を容易にするため、システムの主要な軸は変更することなく、全室個室の病棟以外で必要な評価軸を追加することで適用することとする。

図 3-5 にアルゴリズムのシステムフローを示す。

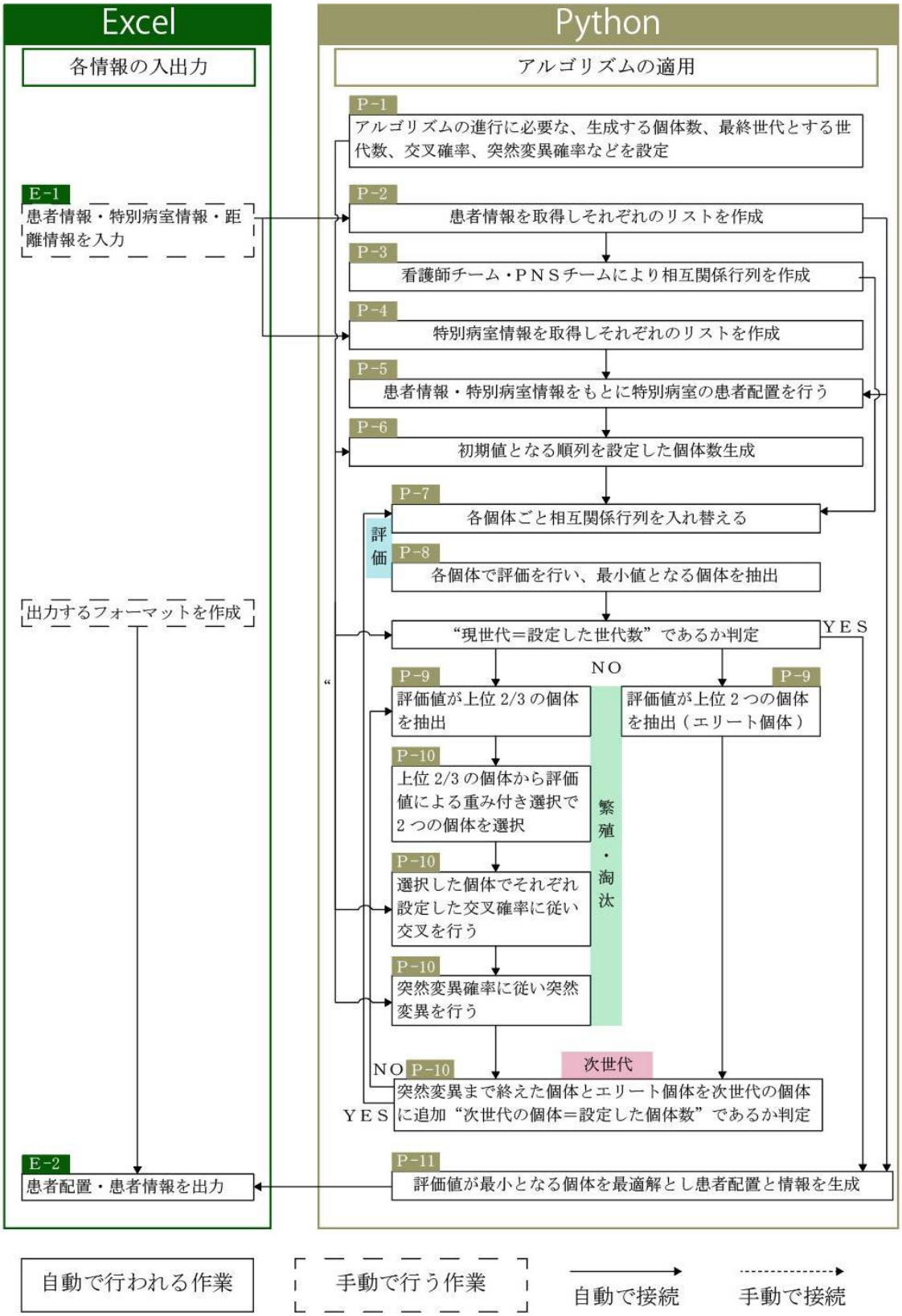

Excel
Python
各情報の入出力
アルゴリズムの適用
P-1
アルゴリズムの進行に必要な、生成する個体数、最終世代とする世代数、交叉確率、突然変異確率などを設定
E-1
患者情報・特別病室情報・距離情報を入力
P-2
患者情報を取得しそれぞれのリストを作成
P-3
看護師チーム・ＰＮＳチームにより相互関係行列を作成
P-4
特別病室情報を取得しそれぞれのリストを作成
P-5
患者情報・特別病室情報をもとに特別病室の患者配置を行う
P-6
初期値となる順列を設定した個体数生成
P-7
各個体ごと相互関係行列を入れ替える
評価
P-8
各個体で評価を行い、最小値となる個体を抽出
出力するフォーマットを作成
“現世代＝設定した世代数”であるか判定
YES
NO
P-9
評価値が上位 2/3 の個体を抽出
P-9
評価値が上位２つの個体を抽出（エリート個体）
P-10
上位 2/3 の個体から評価値による重み付き選択で 2 つの個体を選択
繁殖・淘汰
P-10
選択した個体でそれぞれ設定した交叉確率に従い交叉を行う
P-10
突然変異確率に従い突然変異を行う
次世代
NO
P-10
突然変異まで終えた個体とエリート個体を次世代の個体に追加“次世代の個体＝設定した個体数”であるか判定
YES
E-2
患者配置・患者情報を出力
P-11
評価値が最小となる個体を最適解とし患者配置と情報を生成
自動で行われる作業
手動で行う作業
自動で接続
手動で接続

図 3-5　システムフロー

ここから図 3-5 に示したシステムフローの内容について記述する。ここでは碧南市民病院の東病棟(以下、碧南東病棟)での実際のアルゴリズムの実行を例として挙げ、記述していくこととする。

E-1　<Excel>患者情報・特別病室情報・距離情報を入力

最適化を行う上で必要となる情報を Excel に入力する。入力するのは患者情報・重症室、感染症室、特別個室などの特別室情報・各病室の SS からの距離情報・各病室間の距離情報である。以下でそれぞれについて記述する。

・患者情報

患者配置の最適化を行う対象病棟に入院する患者の情報を入力する。入力する情報は患者を識別するための患者名、性別、特別室に入室するか否か、看護必要度、受け持ちチーム、PNS が行われている場合は受け持ちペアの 6 つである。表 3-5 に患者情報の入力例を示す。

・特別室情報

病棟には一般患者が入室する病室だけでなく、重症者が入室する個室や一般の病室よりも広く設けられた特別個室などがある。そのような特別室がそれぞれどの病室にあたるのかを入力する。碧南東病棟では特別病室は普通個室のみとなっている。表 3-6 に特別室情報の入力例を示す。

・各病室の SS からの距離情報

病棟における看護師の移動には SS から病室の距離が重要になる。そのため評価値として利用する各病室の SS からの距離をメートル単位で入力する。SS からの距離の行列は *1×ベッド数 n* となる。表 3-7 に SS からの距離情報の入力例を示す。

表 3-5　患者情報入力例→巻末資料

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	2
患者名	A	B	C	D	E	F	G	H	I	J	K	L	N	M	O	P	Q	R	S	T	U	V	W			
性別	男	女	男	女	女	女	男	男	男	男	男	男	男	女	女	女	女	女	女	女	女	女	女			
特別室	個	個	個																							
看護必要度B	0	5	8	9	5	3	5	4	0	0	7	10	10	12	8	10	0	11	2	0	0	4	0			
受け持ちチーム	1	2	2	1	1	1	2	2	1	2	2	2	2	2	2	2	2	2	2	1	2	1	1			
ペア	A	C	B	A	A	A	C	C	A	C	B	B	B	B	C	C	B	B	C	A	C	A	A			

表 3-6→巻末資料

特別室情報入力例

個	0	1	2	3

表 3-7　SS からの距離情報入力例→巻末資料

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
SS	10	6	5	10	13	13	13	13	20	20	20	20	25	25	25	25	25	25

・各病室間の距離情報

SS からの距離と同じく病室間の距離も重要となる。評価値として利用する各病室間の距離をメートル単位で入力する。*ベッド数 n*×*ベッド数 n* の行列となり、二次割り当て問題における距離行列 *W* にあたる行列となる。

P-1　<Python>・アルゴリズムの進行に必要な、生成する個体数、最終世代とする世代数、交叉確率、突然変異確率などを設定

遺伝的アルゴリズムの進行には進化させていく個体の数、アルゴリズムを終了させる代数、交叉を行う確率、突然変異を行う確率を設定する必要がある。個体数と世代数についてはアルゴリズムの処理速度に関わってくる要素である。本研究ではすべての病棟で一律して個体数=32、交叉確率、突然変異確率は一般的に用いられる交叉確率=90%、突然変異確率=1%を用いる。世代数についてはそれぞれの病棟で評価値の収束値が変わってくるため変更する必要がある。用いる世代数については実際のアルゴリズムの実行により収束値の検証を行い決定することとする。世代数の決定については後述する。

P-2　<Python>患者情報を取得しそれぞれのリストを作成

患者情報をアルゴリズム内で取得し、それぞれ *1*×*ベッド n* のリストを作成する。

P-3　<Python>看護師チーム・PNS チームにより相互関係行列を作成

二次割り当て問題における相互関係行列 *F* にあたる行列を作成する。本アルゴリズムでは、相互関係行列は看護師チームの受け持ち患者から移動頻度を仮定し、移動頻度が高いほど数値が大きくなるように表記する。*n* 個の病室があるとき患者数も *n* 人必要となる。対象病棟では 2 つの看護師チームがそれぞれ一定の患者を受け持っており、A チームの受け持ち人数を *i* 人、B チームの受

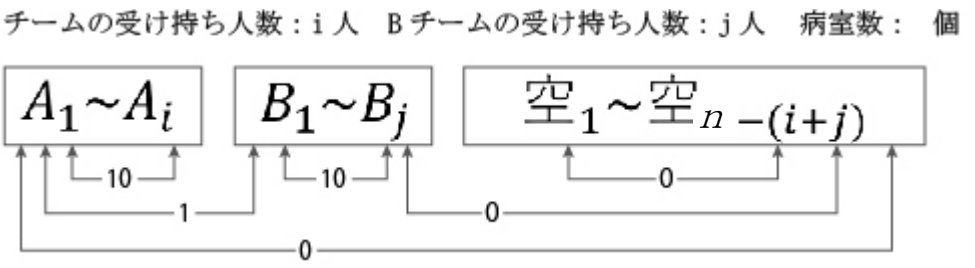


図 3-6　相互関係行列作成概念図

け持ち人数を *j* 人としたときに *j+i<n* となる場合 *n-(i+j)*人の割り当てられたベッドが空ベッドになるダミーの患者 (以下、空患者)が必要となる。本アルゴリズムではチーム内の患者同士の移動頻度を最大値として 10、別チームの患者同士の移動はほとんどないものとし 1、空ベッドには移動することはないものとし最小値の 0 を与える。図 3-6 に相互関係行列作成における概念図を示す。対象病棟が PNS を採用している場合は同チームで、別 PNS ペアである場合は同チーム同ペアと別チームの間の値である 5 を与えるなど移動頻度は自由に設定することが可能である。

P-4　<Python>特別病室情報を取得しそれぞれのリストを作成

特別病室情報をアルゴリズム内で取得し、それぞれ *1×特別病室数*のリストを作成する。

P-5　<Python>患者情報・特別病室情報をもとに特別病室の患者配置を行う

特別病室情報の特別病室名が一致する特別病室に入室する患者をそれぞれのベッドにランダムに配置された順列を P-1 で設定した個体数作成する。入室する患者がベッド数に満たない場合は空患者を配置する。図 3-7 に特別病室の患者配置の例を示す。各ベッドの識別番号として 0 からベッド数*(n-1)*までの番号が振られている(以下、ベッド No.)。碧南東病棟ではベッド No.0~3 が個室となっており患者 A・B・C が個室に入室する患者の場合、患者 A・B・C は No.0~3 のいずれかにランダムに配置され、残り 1 つは空ベッドになる。作成された個体は図 4-7 の個体の場合、*個体 1={空, C, A, B}*、*個体 X={C, A, B, 空 }*のように順列で表現される。これが GA にて進化させていく遺伝子表現となる。

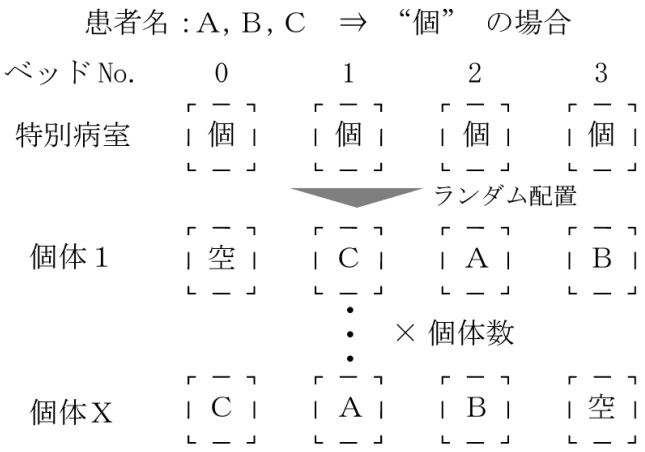


図 3-7　特別病室の患者配置の例

P-6　<Python>初期値となる順列を設定した個体数生成

ここでは一般病室のベッドに入室する患者の配置を行う。特別病室で配置した特別病室に入室する患者及び空患者を除いた残りの患者をランダムに配置する。また、碧南市民病院のように 4 床室で構成されている病院では、男女が同じ病室に配置されないように留意する必要がある。

図 3-8 に初期個体の作成例を示す。男性と女性からランダムに 1 人を 4 床室の 1 ベッドに配置し、その性別に合わせてその室の残りのベッドに同じ性別の患者もしくは空患者をランダムに配置することで、男性室と女性室を作成する。その後 P-5 で作成した順列を加え、初期遺伝子とする。これを設定された個体数作成する。ここでも遺伝子は順列で表現され、個体 1={*空, C, A, B, 空, N, K, 空, 空, 空, 空, 空, 空, T, Q, U, 空, 空, G, H, R, E, 空, S, M, 空, F, O, I, J, L, 空, D, V, W, P* }のような表現になる。

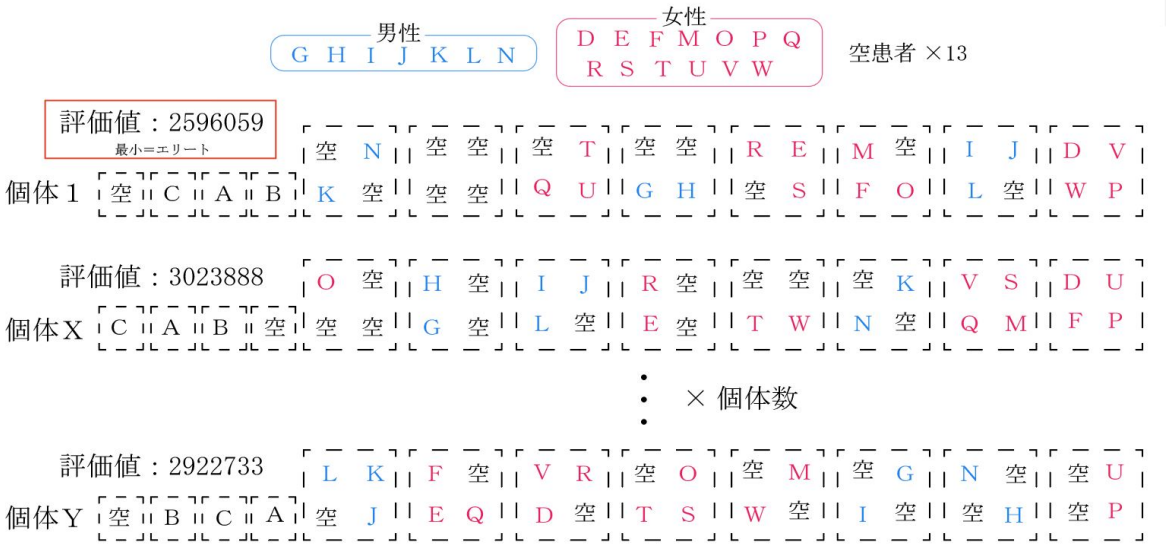


図 3-8　特別病室の患者配置の例

P-7　<Python>各個体で相互関係行列を入れ替える

初期の相互関係行列は Excel に入力した順の患者の配置で作成されている。評価値を計算するにあたり、P-6 で生成した遺伝子の順列に対応するように行及び列を入れ替える必要がある。図 3-9 に相互関係行列の入れ替えの例を示す。図 3-8 に示した個体 1 の特別病室に入室する患者のみを例として抜き出している。実際はこれに加え一般病室に入室する患者も入れ替える必要がある。これを生成されたすべての個体分行う。また同様に、評価に使用する看護必要度についても生成した遺伝子の順列に対応するように列を入れ替える。

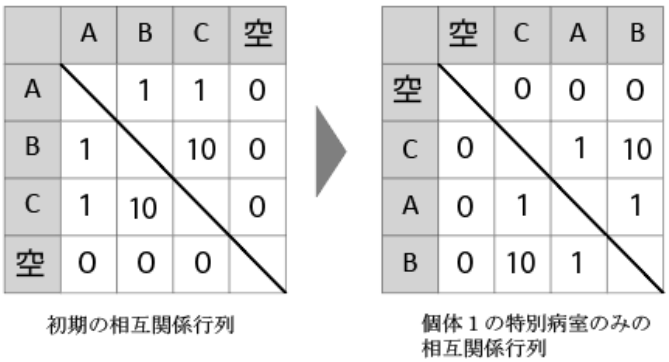


図 3-9　相互関係行列の入れ替えの例

P-8　<Python>各個体で評価を行い、最小値となる個体を抽出

次に、各個体で評価を行う。距離行列と相互関係行列、病室の SS からの距離、看護必要度から評価を行う。図 3-3 で示した二次割り当て問題で用いられる式に、さらに病室の SS からの距離と看護必要度を乗じたものを評価値とする。

病室数を n、距離行列を $W=[wij](1 \leqq i, j \leqq n)$、相互関係行列を $F=[fij](1 \leqq i, j \leqq n)$、病室から SS までの距離を $D=[Dj]\ (j \leqq n)$、看護必要度を $N=[Nj]\ (j \leqq n)$とすると評価値 $C(P)$は以下の式で求められる。

$$\mathrm{C(P)} = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} f_{\pi(i)\pi(j)} D_j N_{\pi(j)}$$

図 3-8 に求められた評価値を表記している。ここでは個体 1 が最小値の個体で初期世代の最適解、エリート個体となる。

P-9　<Python>評価値が上位 2/3 の個体を抽出

評価値が上位 2 つの個体を抽出(エリート個体)

ここでは次世代に進化させていく個体を抽出する。本アルゴリズムでは上位 2/3 の個体のみを抽出し、下位 1/3 の個体は淘汰する。上位 1/3 や 1/2 のみを抽出すると、ある個体の遺伝子が集中的に増加する初期収束が起こる確率が急増したため本アルゴリズムでは上位 2/3 としている。

また上位 2 つの個体はエリート個体として遺伝子の配列を変更することなく、次世代に引き継ぐ。

P-10　<Python>・上位 2/3 の個体から評価値による重み付き選択で 2 つの個体を選択

・選択した個体でそれぞれ設定した交叉確率に従い交叉を行う

・突然変異確率に従い突然変異を行う

・突然変異まで終えた個体とエリート個体を次世代の個体に追加“次世代の個体＝設定した個体数”であるか判定

遺伝子の交叉を行う 2 つの親個体を P-9 で抽出した上位 2/3 の個体から選択する。選択方法はルーレット選択と呼ばれる選択方法を用いる。ルーレット選択は個体の評価値に比例して選択される個体が決まる。低確率ではあるが評価値の低い個体を選択する可能性を残しているのは、その個体のある部分が変化することで突如優秀な個体に進化する可能性を残すためである。

選択した二つの個体で交叉を行う。交叉は設定した交叉確率に従い行われる。図 3-10 に交叉実行の例を示す。本アルゴリズムで用いる交叉方法は順序交叉と呼ばれるものを、男女による病室の配置の制限を設けるために、一部変更したものを用いる。図 3-10 では図 3-8 に示した個体 X と個体 Y が上記のルーレット選択にて選択されたものとして交叉を行っている。特別病室に配置された患者は一般病室には配置されないので、ここでは交叉を行う必要はない。ゆえに一般病室の患者のみ交叉を行う。順序交叉は片方の親からは一部の配列をそのまま受け継ぎ、残りの部分については相対的な順番は保存しつつもう一方の親から受け継ぐ方法である。これを両親で行うので 2 つの子個体が作成される。順列内で一つの要素が重複してはならない問題などに使用される方法である。

交叉の流れとしては、まずこの交叉で用いる切断箇所を 2 箇所ランダムで選択する。ここでは 0 から 8 までの数字の中から 5 と 7 が選択されている。まず親 A の一部の配列を残し、親 B から残りの部分を受け継いだ個体を作成する。親 A では選択された切断箇所内を残し、それ以外の要素は一度リセットする。親 B では親 A で残した要素は重複するため消去する。空患者であってもそれぞ

れに識別番号が振られているため空患者が含まれている場合も同じ番号が一致するものを消去する。次に、親 A に配置し直すために親 B の要素を男性室と女性室別に抜き出す。親 A に配置し直すには 4 床室に配置する 4 の倍数で男女別

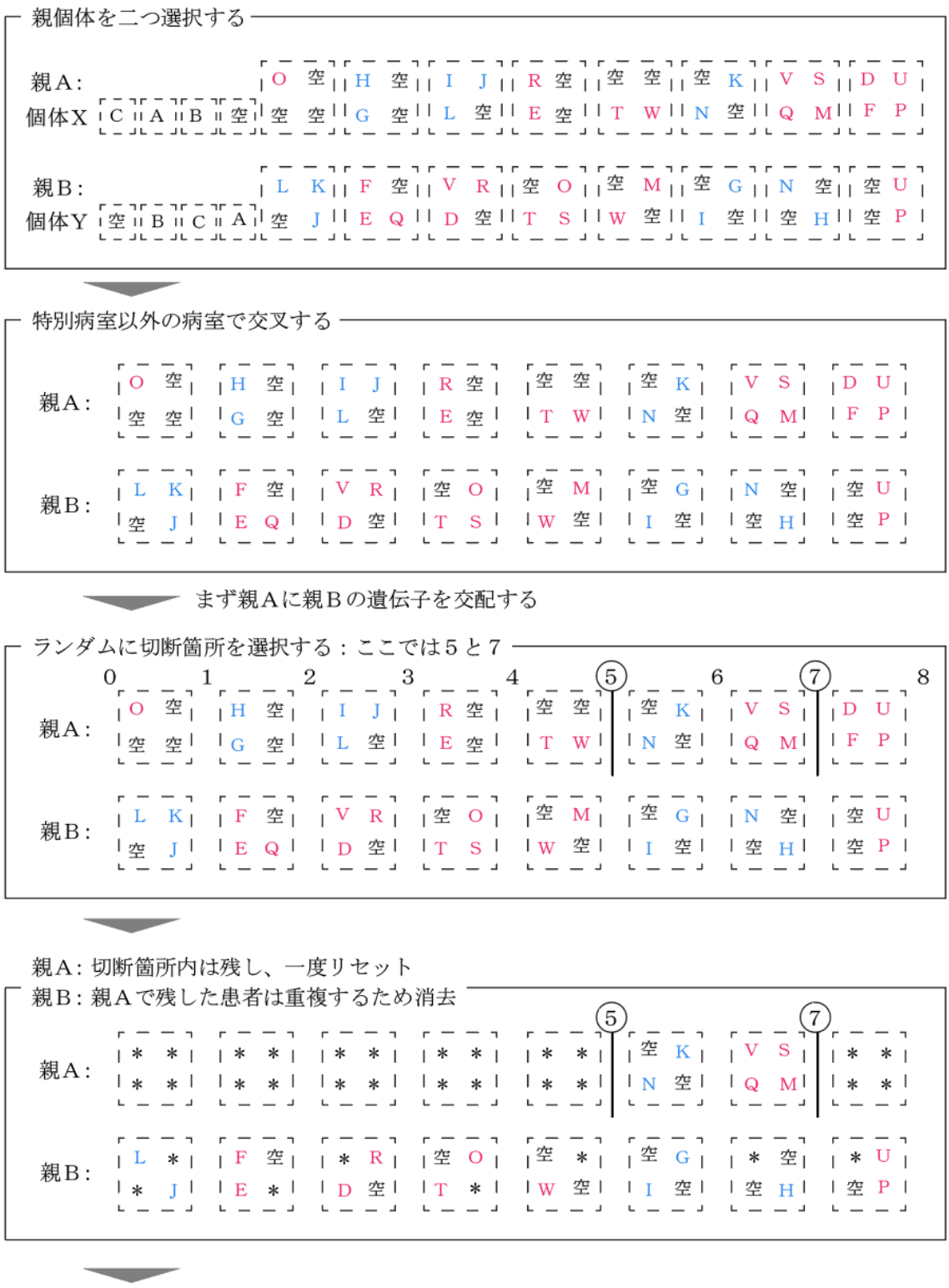


図 3-10　交叉実行の例

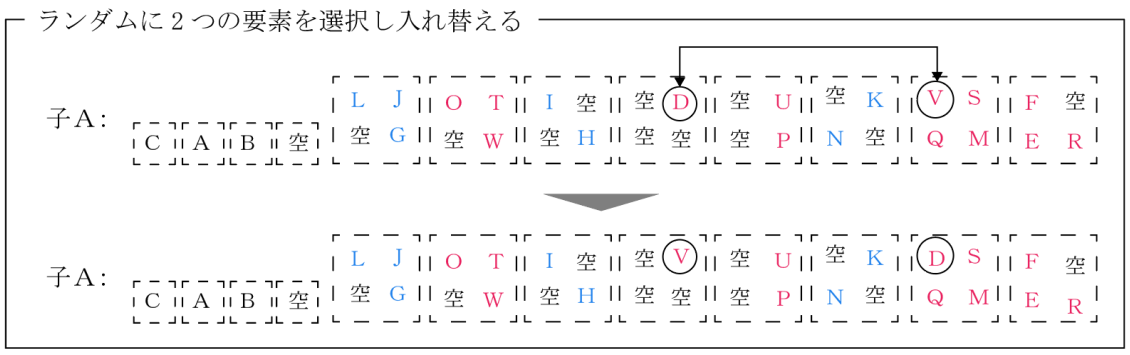


図 3-11　突然変異実行の例

になっている必要があるが、このままだと 4 の倍数でない可能性がある。男女患者はそれぞれ別性別の室に移動はできないので、空患者をランダムに選択・移動を行い 4 の倍数になるまで続ける。その後、親 A のリセットした場所にランダムに配置し直す。そこに親 A の元の特別病室を戻し、子 A とする。同様の作業を親 A と親 B を逆にして行い、親 B の一部の配列を残し、親 A から残りの部分を受け継いだ子 B を作成する。

作成した子個体を対象に突然変異を行う。突然変異は設定した突然変異確率に従い行われる。図 3-11 に突然変異実行の例を示す。突然変異は個体の要素を二つランダムに選択し、それを入れ替えることで実行される。

突然変異まで終えた子個体とエリート個体を次世代の個体とし、次世代の個体が設定した個体数になるまで交叉、突然変異を行う。設定した個体数に達し

図 3-12　世代ごとの移行

た後、P-8 で示したその世代の評価を行う。これを設定した世代数に達するまで行う。

図 3-12 に遺伝的アルゴリズムの世代ごとの評価値と患者配置の移行を示す。図 3-11 までで示し作成した 2 代目では 1 代目のエリート個体の評価値を子 Z が更新している。そのため次のエリート個体は子 Z となる。その後も 15 代目、100 代目を示したが、世代ごとに評価値が徐々に減少しており、最適化されていることが分かる。

P-11　<Python>評価値が最小となる個体を最適解とし患者配置と情報を生成

最終世代まで評価・交叉・突然変異が終了した後、最終世代で最小となる個体を最適解として抽出する。仮に図 4-12 に示した 100 代目を最終世代とすればこの例の最適解は*{空, C, A, B, S, E, F, O, H, 空, G, I, 空, 空, J, 空, 空, 空, 空, 空, T, Q, 空, U, K, 空, N, L, M, V, W, P, 空, R, D, 空}*となる。この順列を基に患者名以外の患者情報の順序を入れ替える。

E-2　<Excel>患者配置・患者情報を出力

P-11 で最適解を基に入れ替えた患者名、性別、看護必要度、担当看護師チーム、担当 PNS チームを Excel に出力する。これにより患者配置をビジュアル的に確認できるようにしている。図 3-13 は碧南東病棟の Excel への出力の例である。

421	422	423	425	426	
空	C	A	B	S	E
0 0	男 8	男 0	女 5	女 2	女 5
0 0	2 B	1 C	2 C	2 C	1 A
				F	O
				女 女	女 8
				1 A	2 C

NS	427		428	
	H	空	0	空
	男 4	0 0	0 0	0 0
	2 C	0 0	0 0	0 0
	G	I	J	空
	男 5	男 0	男 0	0 0
	2 C	1 A	2 C	0 0

429		430		431		432		433	
空	R	M	V	U	空	T	Q	空	空
0 0	女 11	女 12	女 4	女 0	0 0	女 0	女 0	0 0	0 0
0 0	2 B	2 B	1 A	2 C	0 0	1 A	2 B	0 0	0 0
D	空	W	P	N	L	空	K	空	空
女 9	0 0	女 0	女 10	男 10	男 10	0 0	男 7	0 0	0 0
1 A	0 0	1 A	2 C	2 B	2 B	0 0	2 B	0 0	0 0

図 3-13　Excel への出力の例

P-1　<Python>・アルゴリズムの進行に必要な、生成する個体数、最終世代とする世代数、交叉確率、突然変異確率などを設定

設定する世代数については実際のアルゴリズムの実行により収束値の検証を行い決定する。図 3-13～3-16 は下呂温泉病院の西病棟と東病棟及び碧南市民病院の西病棟と東病棟それぞれの最終世代数を 10,000 としたときの世代別の最適解の推移を示したものである。それぞれ 10 回アルゴリズムを実行し、10,000 世代目の最適解の評価値に収束した世代と 1,000 世代以上評価値の変化しなかった世代を示している。

本アルゴリズムでは収束した世代の平均からそれぞれ最終世代を 2,500、1,500、8,000、5,000 と設定した。各病棟での最終的な最適解が増加するにつれて収束した世代も増加することが分かる。今後はこの値を参考にして最終世代の決定を行う。

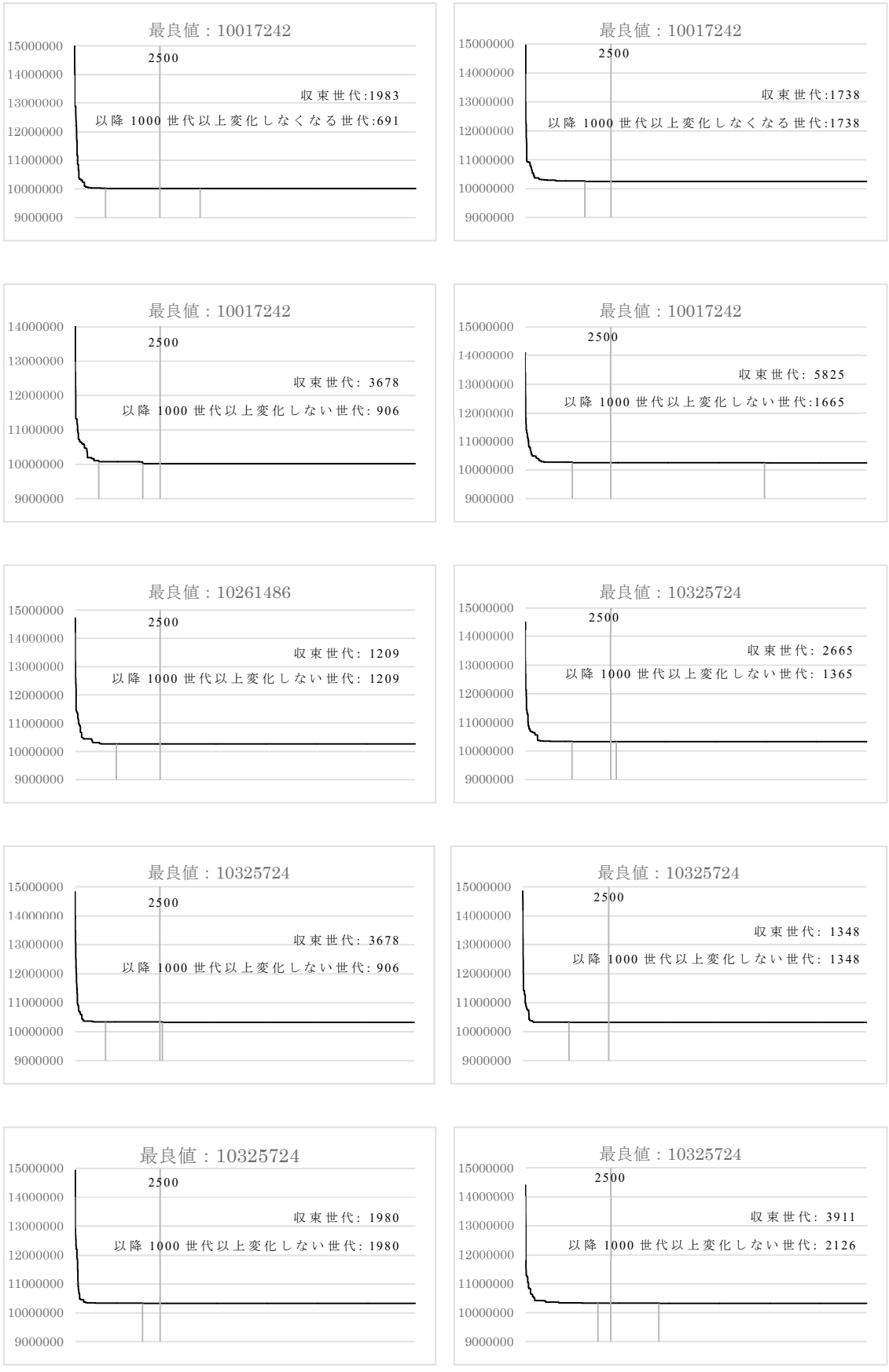
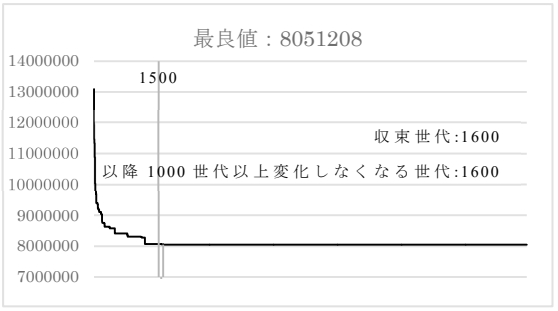
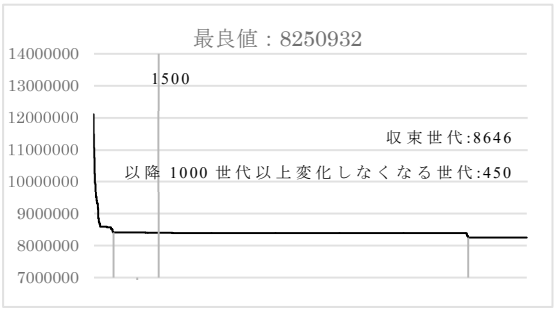
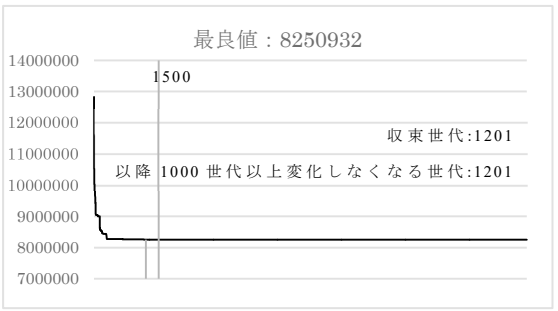


図 3-13　下呂温泉病院西病棟の最適解の推移







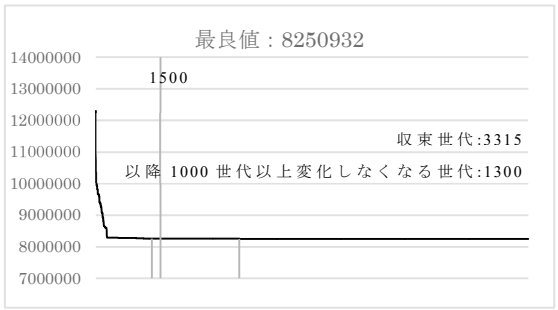


図 3-14　下呂温泉病院東病棟の最適解の推移

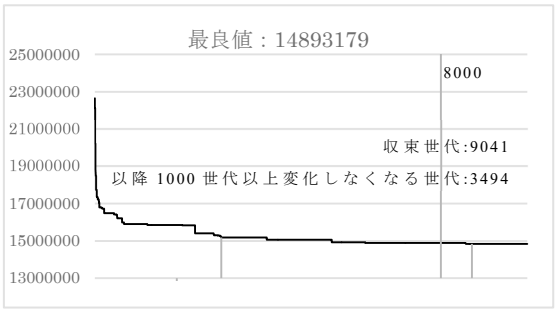
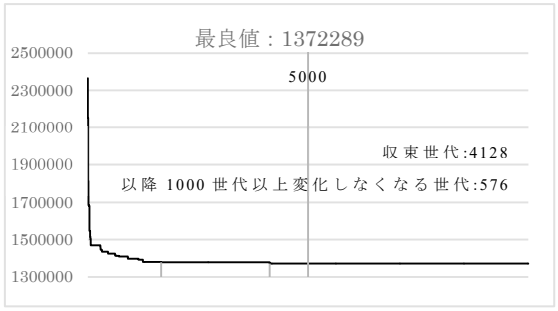
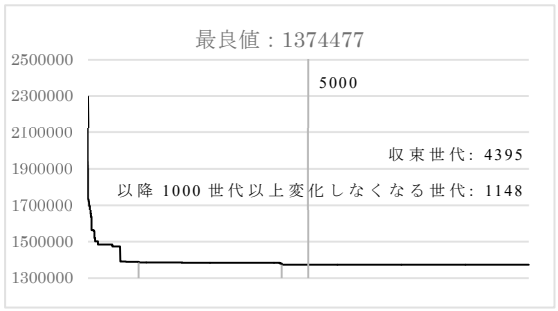


図 3-15　碧南市民病院西病棟の最適解の推移





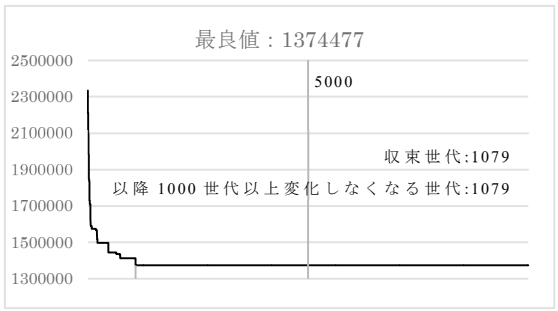


図 3-16　碧南市民病院東病棟の最適解の推移

3－4－3　BIM への適用

ここまでは、看護師の一般業務での利用も容易にするため、Excel に情報を入力することで最適解を得られるアルゴリズムとしていた。設計段階での利用には次章で使用する竹内が用いた BIM を用いたシミュレーションとの連携が想定される。シミュレーションはそれぞれの部屋などに着点を設定し、着点同士の距離を測定、動線量を算出するものである。

Dynamo を用いて BIM への適用を行う。図 3-17 に Dynamo で作成したプログラムを示す。本アルゴリズムの BIM への適用方法は以下のとおりである。

① 病室内の着点を Dynamo 内に取り込む
② Excel から患者情報・特別病室情報・距離情報を取り込む
③ Python を用いて遺伝的アルゴリズムにて患者配置の最適解を得る
④ 最適解に従い、病室内の着点を変更する

このように、動線量の算出に用いる各病室の着点を自動で変更することで、その後の動線量シミュレーションにシームレスにつなげることができる。

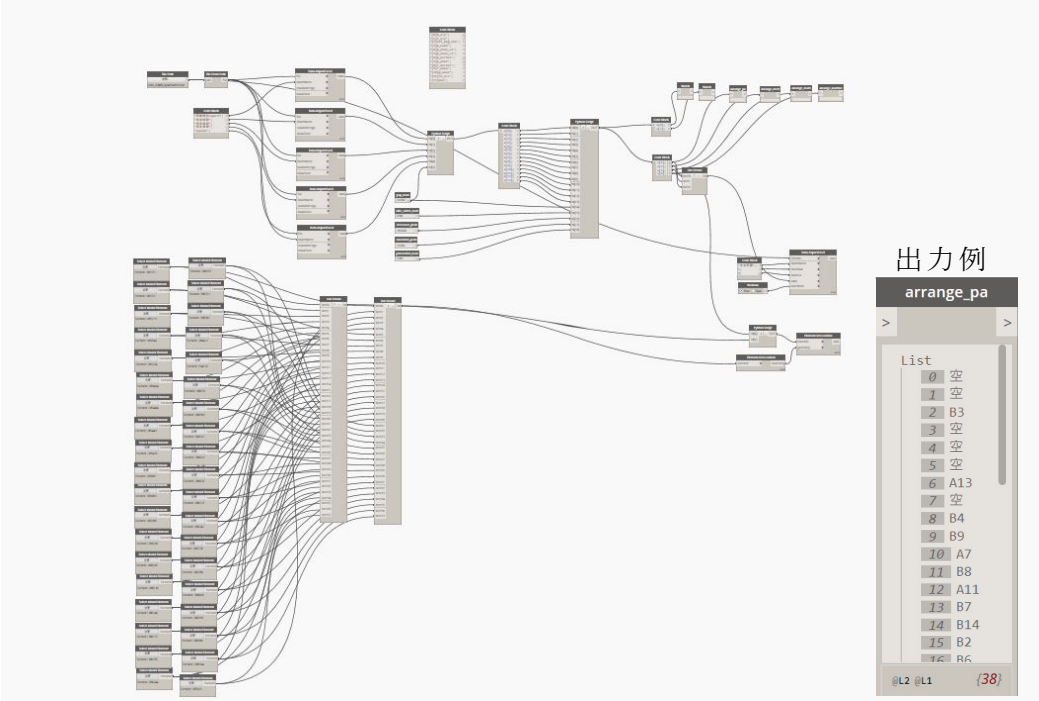


図 3-17　Dynamo で作成したプログラム

3－5 まとめ

本章では看護師の動線量の観点から見て最適な患者配置を得るために、二次割り当て問題及び遺伝的アルゴリズムを用いたアルゴリズムを作成した。二次割り当て問題には、距離行列と相互関係行列が必要であり、本アルゴリズムでは、それをそれぞれ病室間の距離と看護師チームの受け持ち患者から仮定した移動頻度により適用可能であることを確認した。また、患者を配置する際、評価値として SS から病室の距離、患者の看護必要度の取り扱い及び 4 床室への男女別の患者配置についてもアルゴリズムに取入れ、最適化の実践を通して示した。また、BIM を用いたシミュレーションとの連携のため、BIM モデルへの適用も可能であることを示した。しかし、本アルゴリズムでは空病室の取り扱いについては考慮していない。そのため、最適化を行った際、空きベッドのみの病室や空病室のみのユニットが生まれることがある。実際の看護業務の中で、空きベッドをどのように配置しているのか、ルールを確認することでアルゴリズムに組み込むことで、実務での活用の幅は広がるだろう。

第4章

最適解の動線シミュレーションへの適用

4－1 BIM 上の看護動線シミュレーションを用いた分析

本章ではまず、BIM を用いた看護動線量シミュレーションにより調査日における現状の看護動線量を把握する。その後前章のアルゴリズムから得た、各病棟で最適化した患者配置に変更し、看護動線量シミュレーションを行う。さらに看護動線量及び看護効率の上昇の有無を分析する。

4－1－1 シミュレーションの方法及びデータの処理

本研究では竹内が用いた BIM を用いたシミュレーションにて解析を行う。分析方法は、以下のとおりである[竹内他, 2020]。

① 看護追跡調査の結果から、病棟内の各地点間の移動回数のみ抽出する。（スタッフステーションや倉庫内の詳細な移動については、本シミュレーションでは省略している。）
② 通過頻度の高い経路を明らかにするため、あらかじめ病棟図面（BIM モデル）上に単純化した動線を設定する。
③ ①のデータから最適な看護動線を探索し、②で想定した経路上に動線回数を表示させる。
④ シミュレーションにより動線量を算出しさらに、動線量から一般的な人の歩く速さ（秒速 1.25m と仮定）で除した移動時間を計算する。

本シミュレーションではあらかじめ予想される経路を線で描画し、その線に基づき動線量を BIM モデル上に表示する。さらに、動線量が多い線ほど濃い色で表示することでビジュアル的にも動線量を把握できる。看護動線シミュレーションにて算出された距離を基づき、人の歩く速さを 1.25m/s と仮定し想定移動時間を算出している。

4－2 碧南市民病院のシミュレーション分析

4－2－1 最適配置とシミュレーションの結果

碧南市民病院のシミュレーションは東病棟、西病棟ともに各チーム日中看護師、日勤看護師 1 名ずつ計 8 名を分析対象とする。各看護師で業務時間による差が出ないように調査当日に発生していた残業は含めないものとした。また、本調査では患者のプライバシーの観点から病室内の追跡は行っておらず、4 床室では看護師の患者ごと訪問回数について把握できていないため、病室訪問回数と病室内患者の看護必要度の比率から訪問回数を概算して用いている。

図 4-1 に現状の患者配置、図 4-2 に最適化した患者配置を示す。患者名には最適化する際に Excel に入力した患者情報で示したものを用いる[巻末資料 3~6]。図 4-3～4-10 に各病棟の各看護師に対するシミュレーションの結果を示す。ここでは

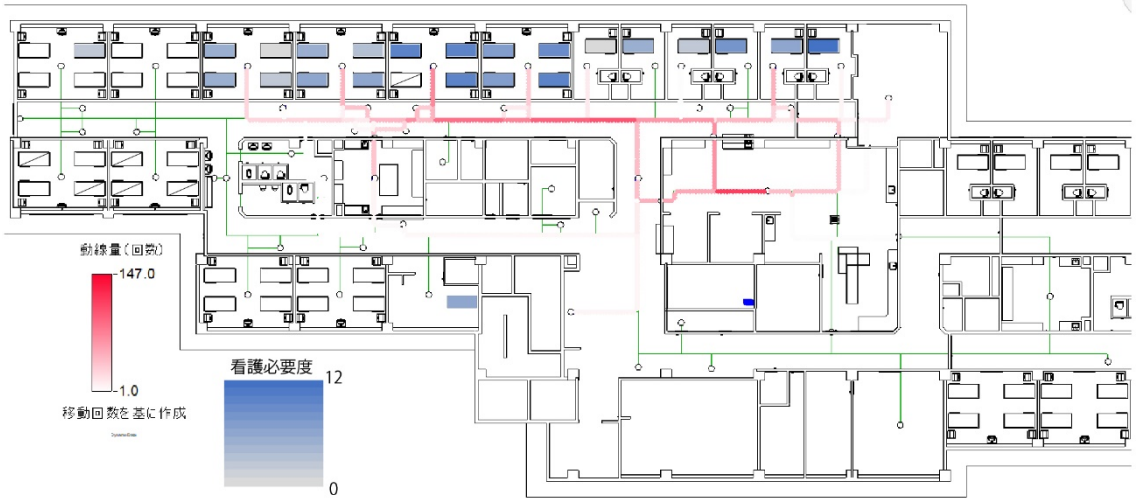
NC 内のみ及び SS 内のみの移動は移動時間・距離には含めず一つの場所としてカウントしており、ある場所からある場所への移動のみを廊下移動時間・距離としている。また対象看護師のチームの受け持ち患者のみ看護必要度を大きいものほど濃い色で、小さいものほど薄く表示している。

碧南市民病院の現状分析は竹内が既発表であるが、最適化後のシミュレーション結果との比較のためここでは概要を示し分析も行う。

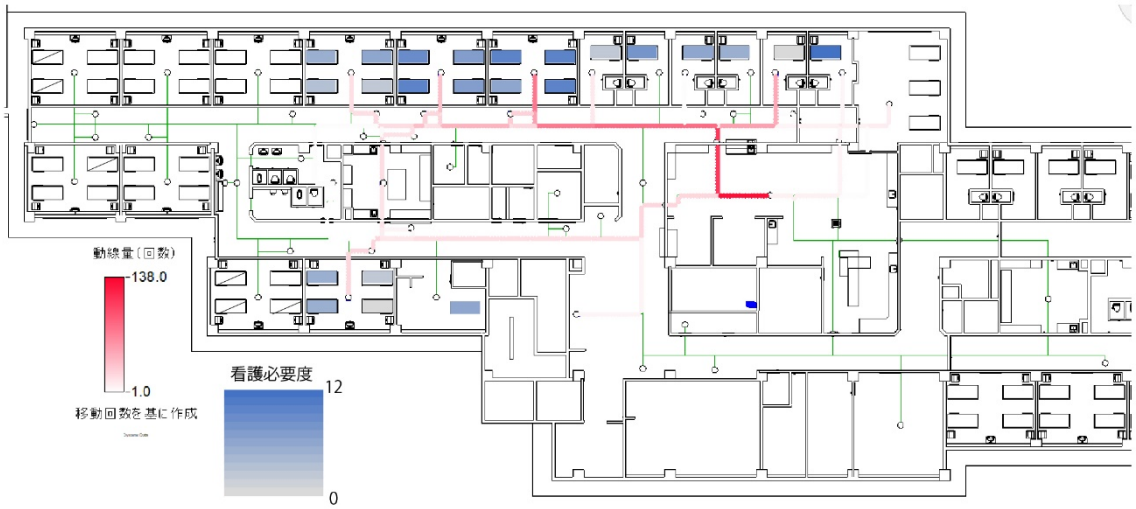
図 4-1　碧南市民病院現状の患者配置

図 4-2　碧南市民病院最適化した患者配置

西病棟　　日中看護師Ｆ　現状配置　（8：30 ～ 20:30）



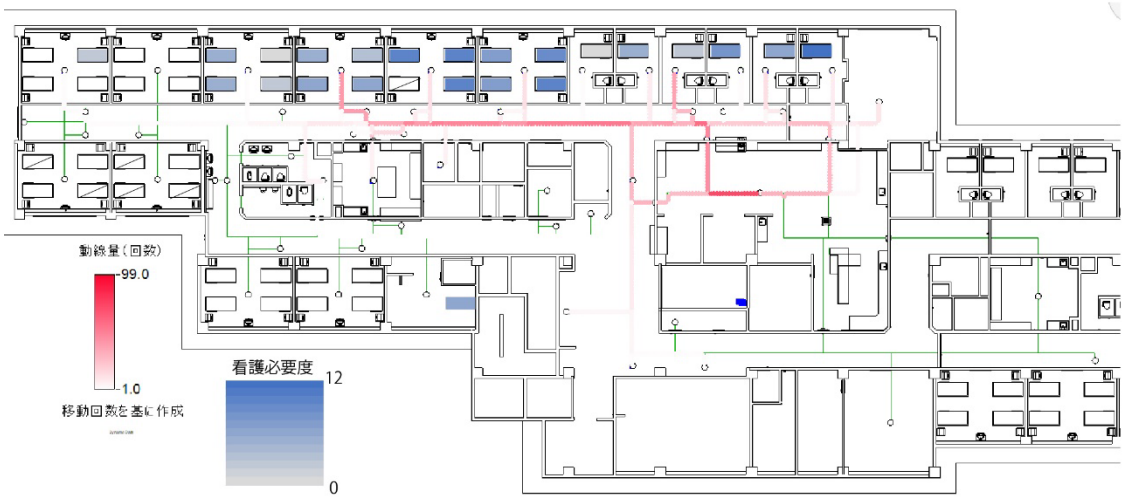
西病棟　　日中看護師Ｆ　最適配置　（8：30 ～ 20:30）



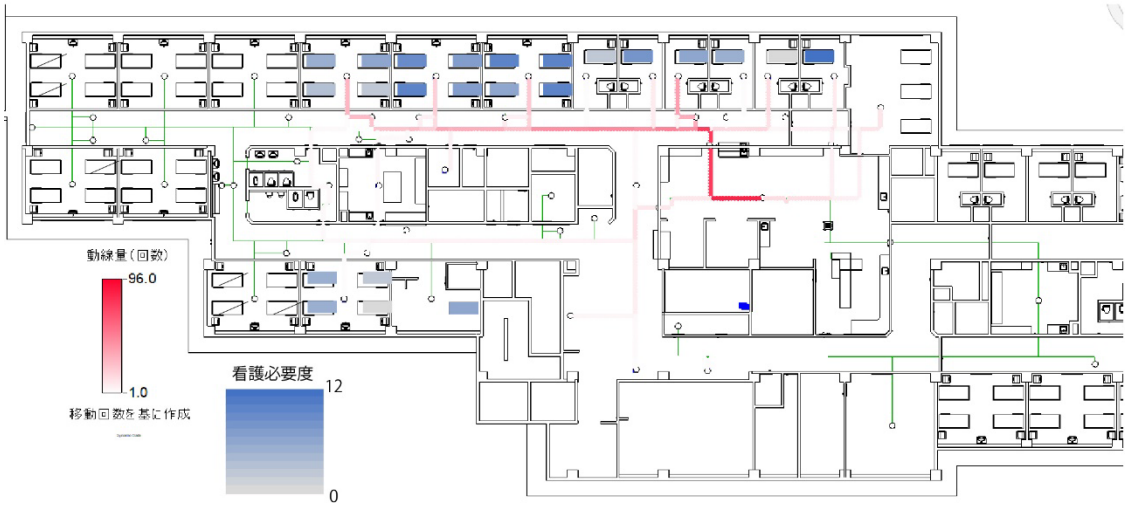
	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	4852.5 m	4066.6 m
看護師総移動時間（秒速 1.25m で換算）	64.7 分	54.2 分

図 4-3　碧南市民病院西病棟-日中看護師 F

西病棟　　日勤看護師 E　現状配置　（8：30 ～ 17:15）



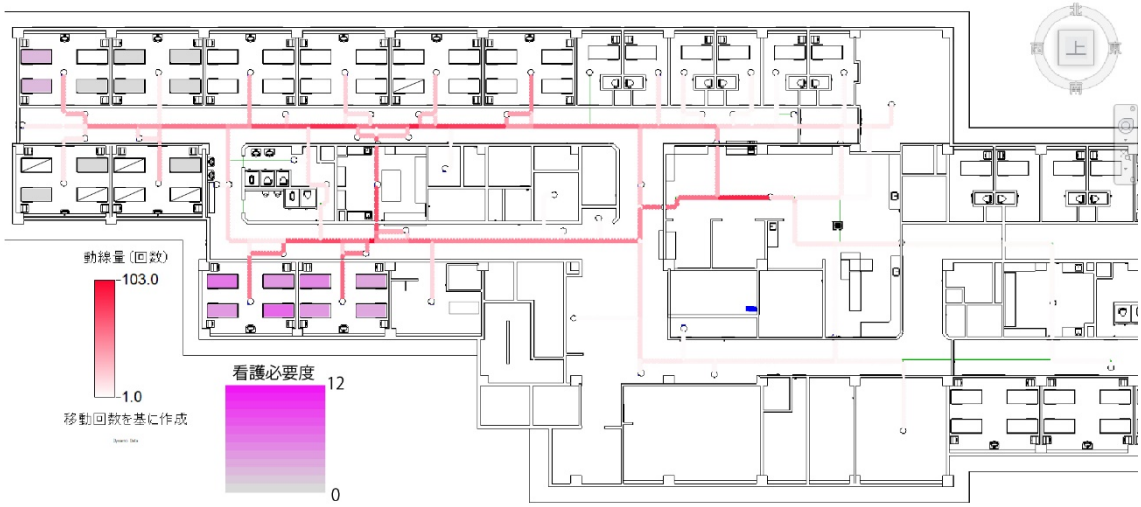
西病棟　　日勤看護師 E　最適配置　（8：30 ～ 17:15）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	3057.5 m	2559.5 m
看護師総移動時間（秒速 1.25m で換算）	40.8 分	34.1 分

図 4-4　碧南市民病院西病棟-日勤看護師 E

西病棟　　日中看護師 D　現状配置　(8：30 ～ 21：30)

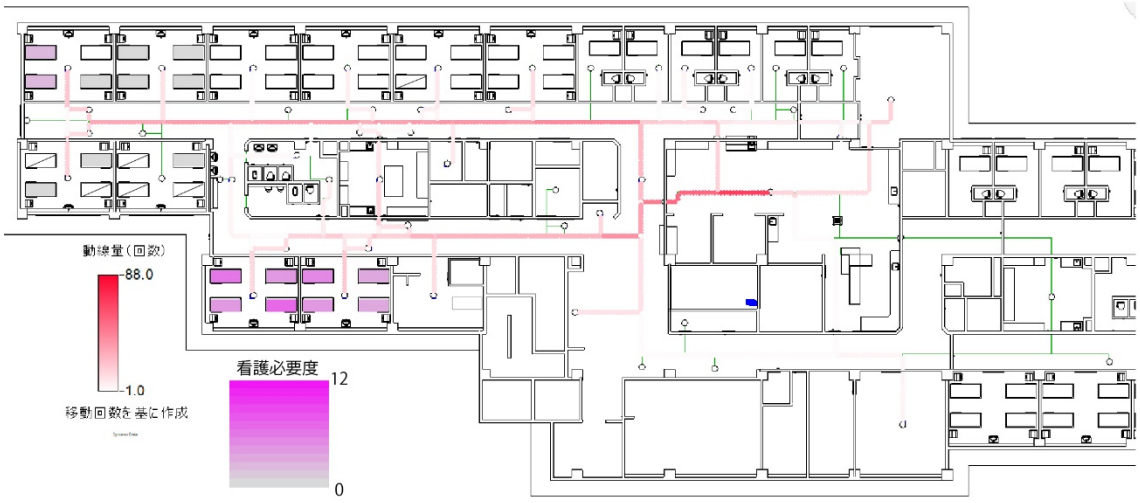


西病棟　　日中看護師 D　最適配置　(8：30 ～ 21：30)

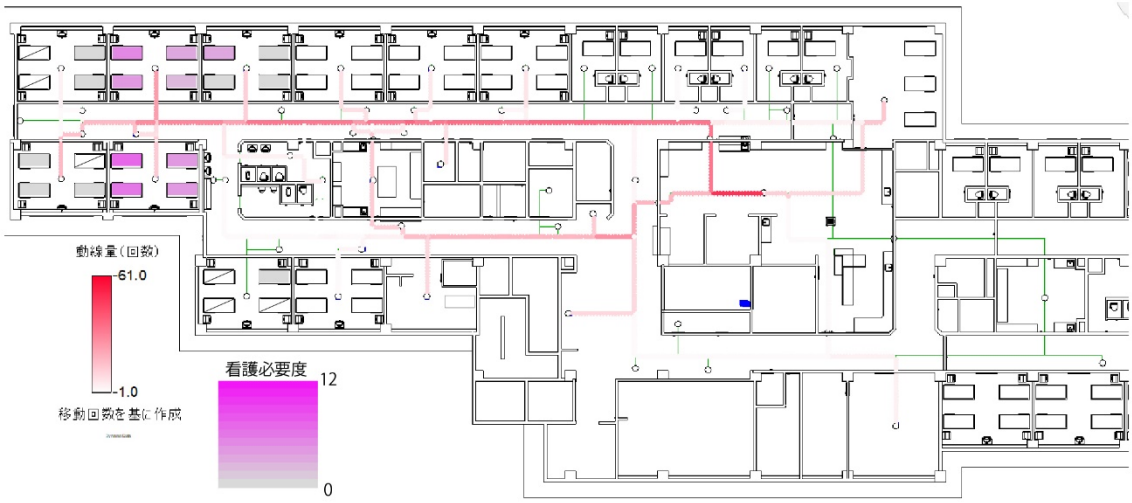
	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	7371.0 m	7516.8 m
看護師総移動時間（秒速 1.25m で換算）	98.3 分	100.2 分

図 4-5　碧南市民病院西病棟-日中看護師 D

西病棟　　日勤看護師 C　現状配置　（8：30 ～ 17:15）

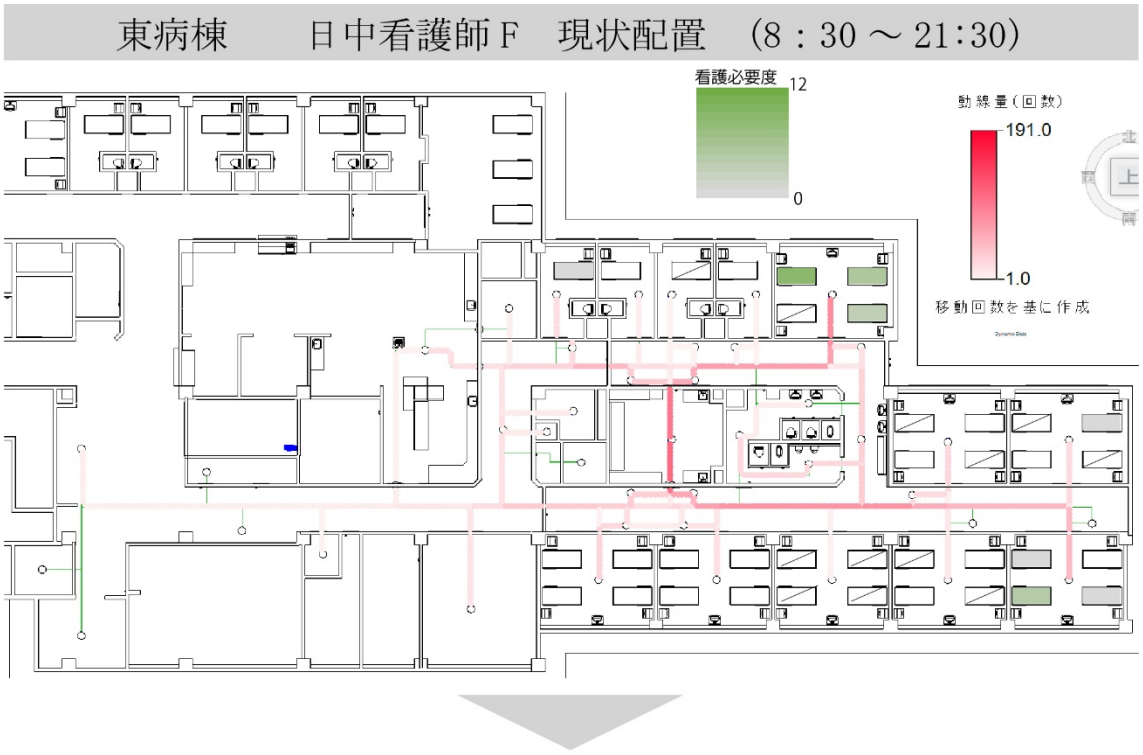


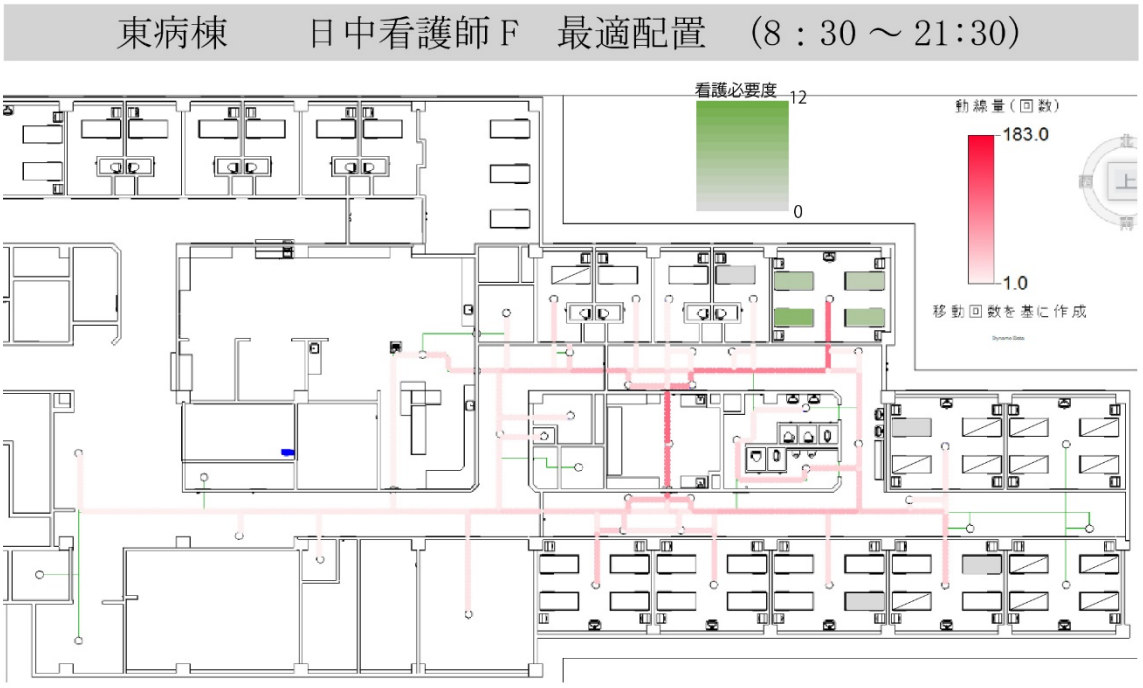
西病棟　　日勤看護師 C　最適配置　（8：30 ～ 17:15）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	3841.8 m	3089.5 m
看護師総移動時間（秒速 1.25m で換算）	51.2 分	41.2 分

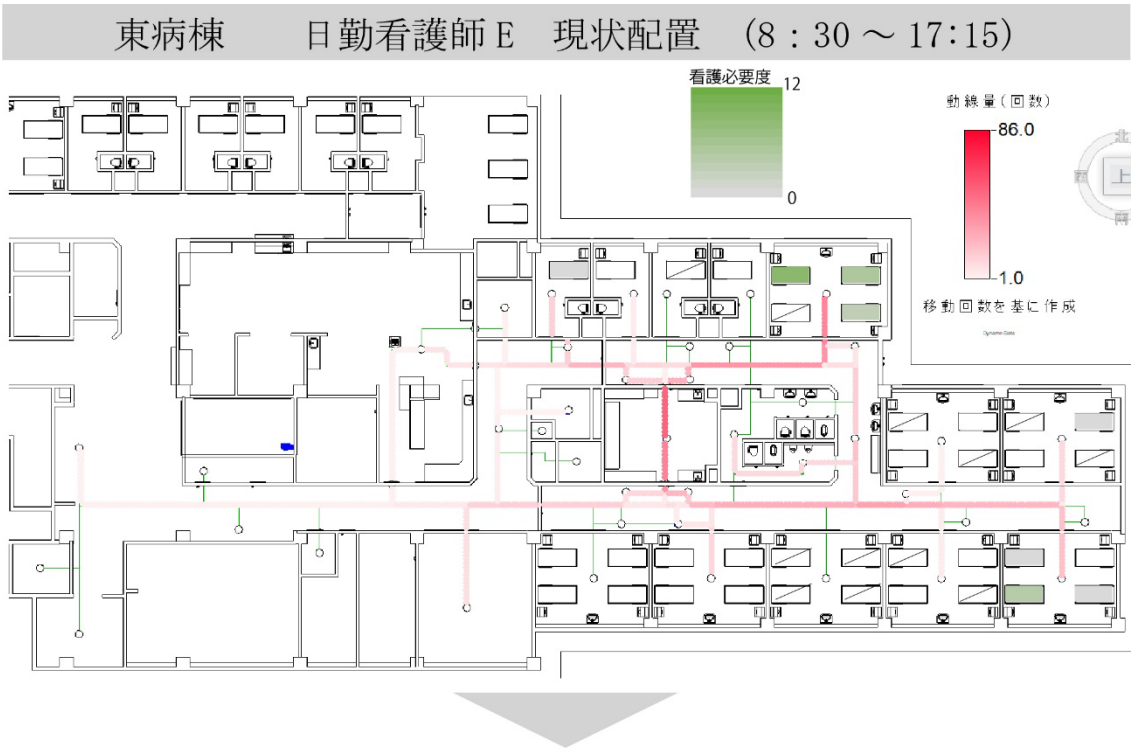
図 4-6　碧南市民病院西病棟-日勤看護師 C

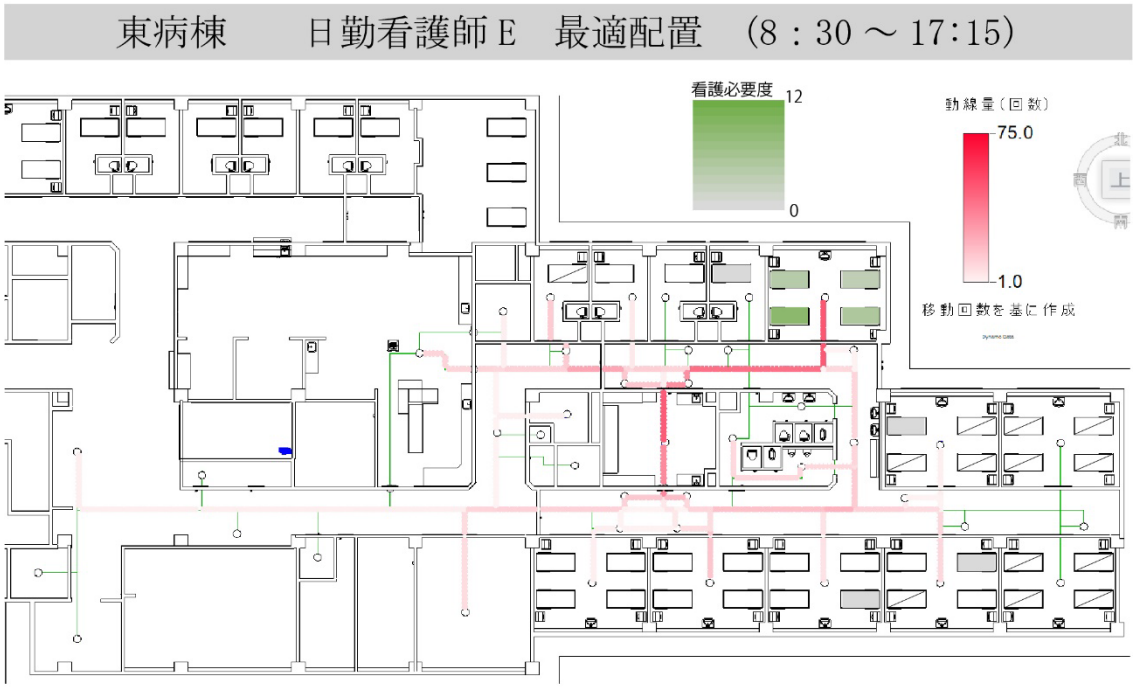




	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	5049.1 m	4418.9 m
看護師総移動時間（秒速 1.25m で換算）	67.3 分	58.9 分

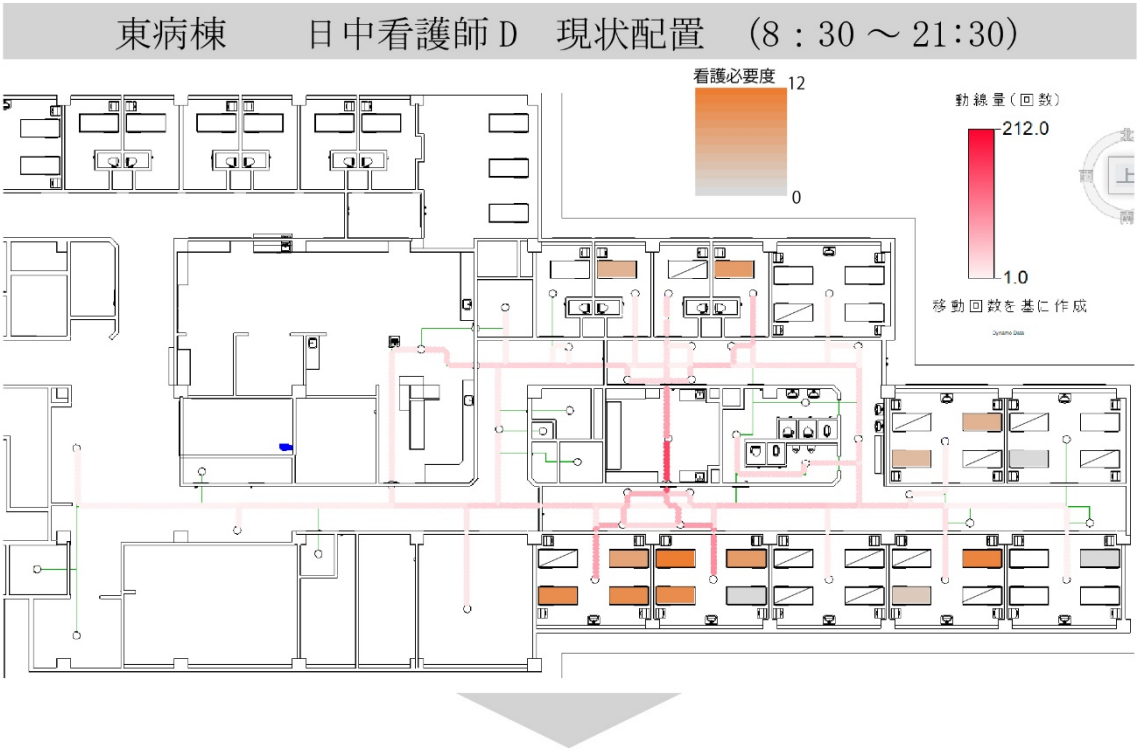
図 4-7　碧南市民病院東病棟-日中看護師 F

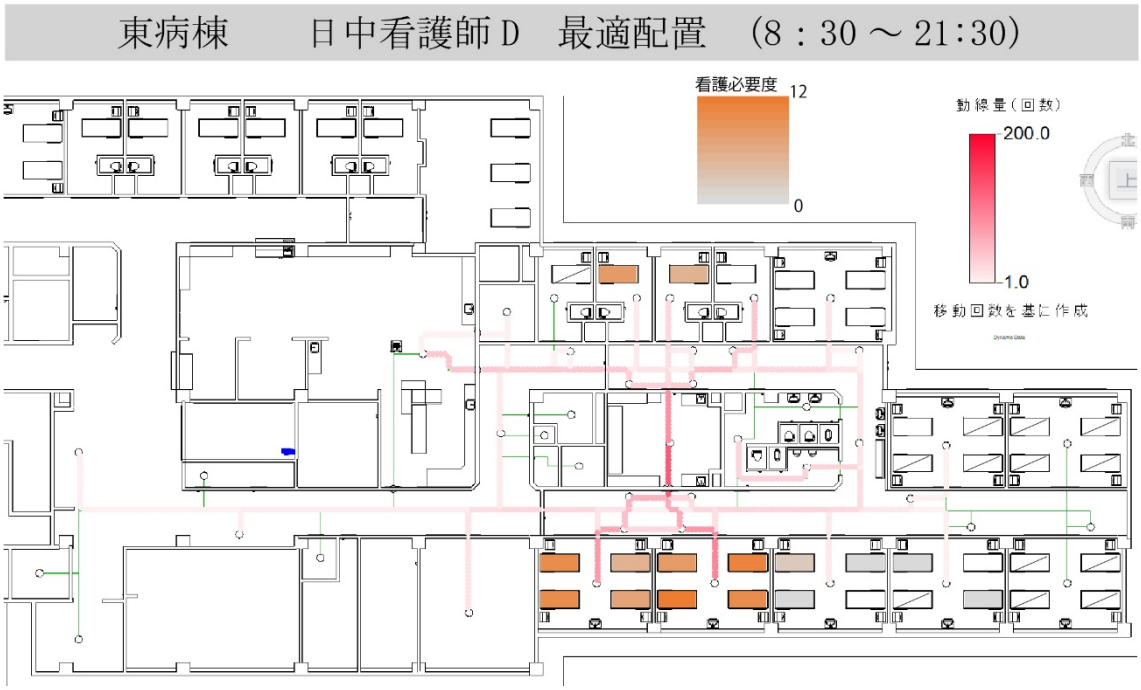




	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	2085.7 m	1777.5 m
看護師総移動時間（秒速 1.25m で換算）	27.8 分	23.7 分

図 4-8　碧南市民病院東病棟-日勤看護師 E

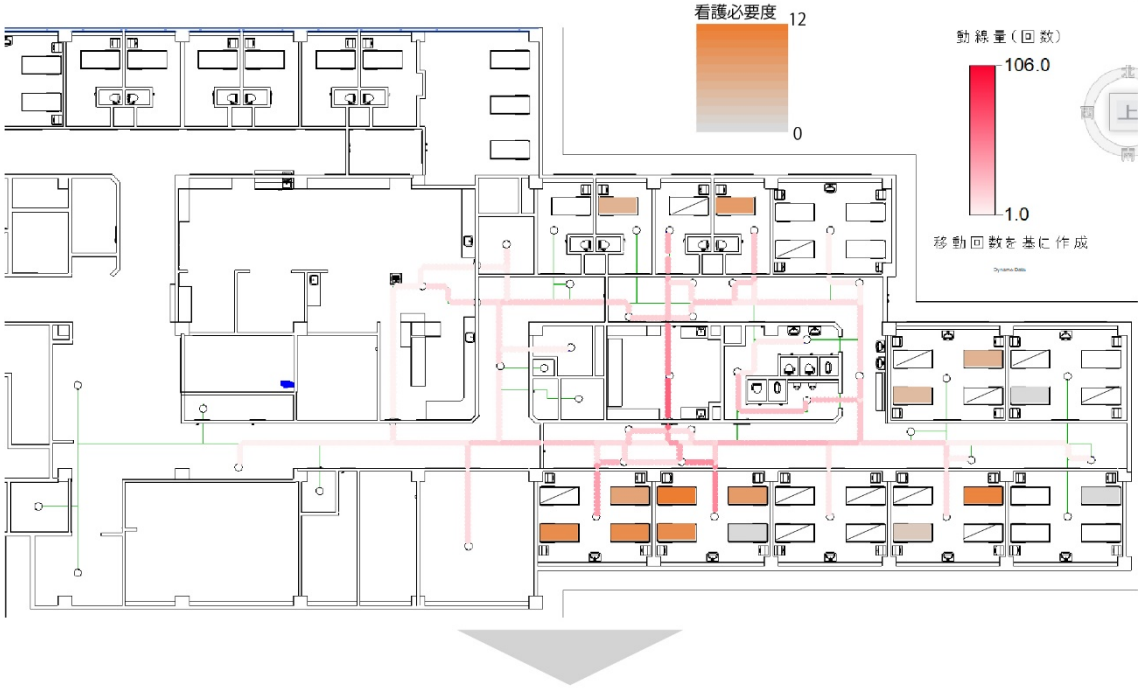




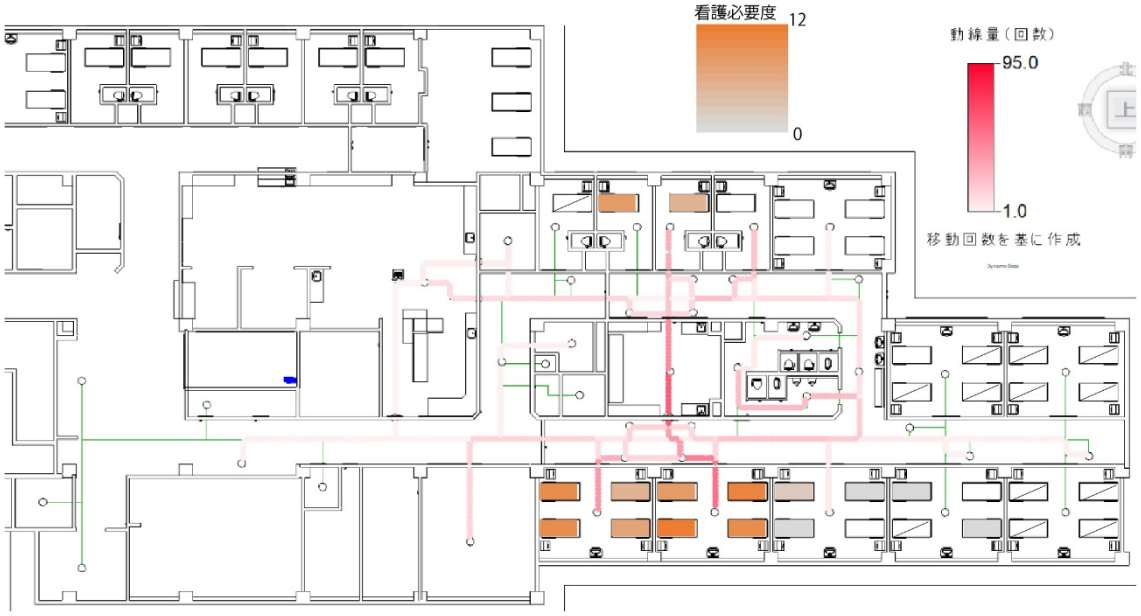
	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	4209.3 m	3741.2 m
看護師総移動時間（秒速 1.25m で換算）	56.1 分	49.9 分

図 4-9　碧南市民病院東病棟-日中看護師 D

東病棟　　日勤看護師 C　現状配置　(8：30 ～ 17:15)



東病棟　　日勤看護師 C　最適配置　(8：30 ～ 17:15)



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	4209.3 m	3741.2 m
看護師総移動時間（秒速 1.25m で換算）	56.1 分	49.9 分

図 4-10　碧南市民病院東病棟-日勤看護師 C

4－2－2　シミュレーションの分析

表 4-1 に看護師の病室訪問回数及びシミュレーションから得た移動時間と移動距離、その結果から得た時間の増減差と距離の増減差、増減率を示す。

現状の西病棟をみると、A チームより B チームの方が看護動線量が多くなっていることが分かる。特に、日勤 C は訪問回数が少ないにもかかわらず日勤 E よりも看護動線量が多くなっている。西病棟では看護拠点を SS としているため、端の病室まで距離があるが A チームの担当患者は SS に近い病室に配置され、B チームの担当患者は遠い病室に配置されている。そのため、チームによる看護動線量の差には担当患者の配置が大きく起因していると推測できる。さらに、日中 D は病室訪問回数が全ての看護師と比較して非常に多くなっている。出力されたシミュレーション結果をから、これは他の看護師よりも担当患者外の病室に頻繁に移動していることによるものだと考えられる。日中 F と比較して移動距離に対する病室訪問回数は大きく違いはないため、SS からの病室への移動よりも病室間の移動が多かったことや SS から遠い病室より比較的近い病室への移動が多かったことが推測される。東病棟をみると日勤には移動距離に対する病室訪問回数に大きく違いがみられなかったが、日中では日中 F が、病室訪問回数が少ないにもかかわらず移動距離が多くなっている。東病棟は西病棟ほどでは無いものの、NC から遠い病室が存在し、そこに A チームの担当患者が集中している。そのため移動距離が多くなったと考えられる。また、西病棟の A チームを除く全ての看護師のシミュレーション結果で担当外の患者が入室している病室への移動頻度が高いことが観察された。

最適化された患者配置は西病棟では、個室病室に入室している患者が全て A

表 4-1　シミュレーション結果

		現状分析			最適化後の分析		比較分析		
		移動時間（分）	移動距離（m）	病室訪問回数（回）	移動時間（分）	移動距離（m）	時間の増減差（分）	距離の増減差（m）	増減率（%）
西病棟	日中F	64.7	4852.5	124	54.2	4066.6	-10.5	-785.9	-16.2
	日勤E	40.8	3057.5	73	34.1	2559.5	-6.7	-498.0	-16.4
	日中D	98.3	7371.0	201	100.2	7516.8	1.9	145.8	1.9
	日勤C	51.2	3841.8	68	41.2	3089.5	-10.0	-752.3	-19.5
東病棟	日中F	67.3	5049.1	133	58.9	4418.9	-8.4	-630.2	-12.5
	日勤E	27.8	2085.7	58	23.7	1777.5	-4.1	-308.2	-14.7
	日中D	56.1	4209.3	143	49.9	3741.2	-6.2	-468.1	-11.1
	日勤C	32.5	2435.4	81	27.0	2028.0	-5.5	-407.4	-16.9

チームの担当だったため、SS からの距離よりも同チームでのまとまりが優先され、A チームが SS から近い病室、B チームが SS から遠い病室に配置されたと考えられる。また、東病棟は B チームに看護必要度が高い患者が多く、その患者が NC から近い病室に配置され、それを基準に他の患者が配置されていると考えられる。この配置は両病棟ともに看護必要度による患者の移動はあったもののチームのまとまりとしては大きな変化はみられなかった。そのため、チームのまとまりという観点からは現状も最適に近い配置がされていたといえる。結果をみるとほとんどの看護師で看護動線量が削減されている。最大で約 785m、時間にして約 10 分の看護動線量が削減された。削減された看護師の増減率を見ると最大で 19.5%削減されている。平均して約 15%削減され、全ての看護師で削減率がかなり高いことがわかる。チームのまとまりとしては最適に近い状態であったが、看護必要度の高さからくる配置の最適化によって大きく看護動線量が削減された。比較的低い東病棟と日中 F と日中 D については病室訪問回数が多いことと担当外の患者が入室している病室への移動頻度が高い点が共通している。一方で、西病棟の日中 D では看護動線量が削減されず、わずかに増加している。この看護師は患者訪問回数と担当外の患者が入室している病室への移動が非常に多かった。このように病室訪問回数と担当外の病室への移動回数が増減率に大きく影響を与えていると考えられる。

4－3 下呂温泉病院のシミュレーション分析

4－3－1 最適配置とシミュレーションの結果

下呂温泉病院のシミュレーションは東病棟、西病棟ともに各チーム長日勤看護師、日勤看護師 1 名ずつ計 8 名を分析対象とする。

図 4-11 に現状の患者配置、図 4-12 に最適化した患者配置を示す。図 4-13～4-20 に各病棟の各看護師に対するシミュレーションの結果を示す。ここでは看護師の SS 内の滞在場所に伴い、スタッフコーナー（主に点滴準備などの作業を行う場所）と SS 中央（SS 内の PC 作業や申し送りなどを行う場所）の 2 点のみを設定し着点としている。

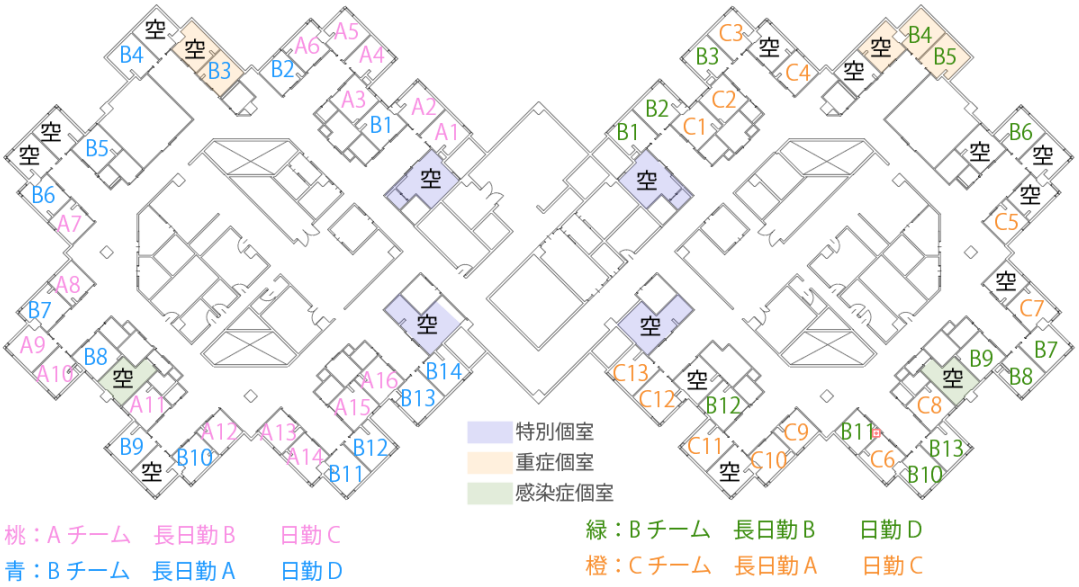


図 4-11　下呂温泉現状の患者配置

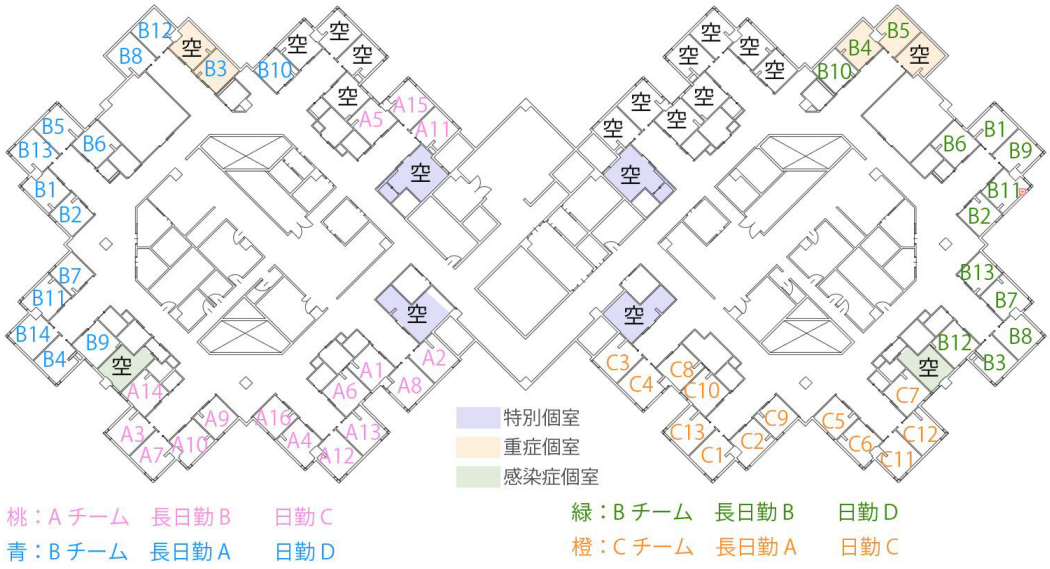
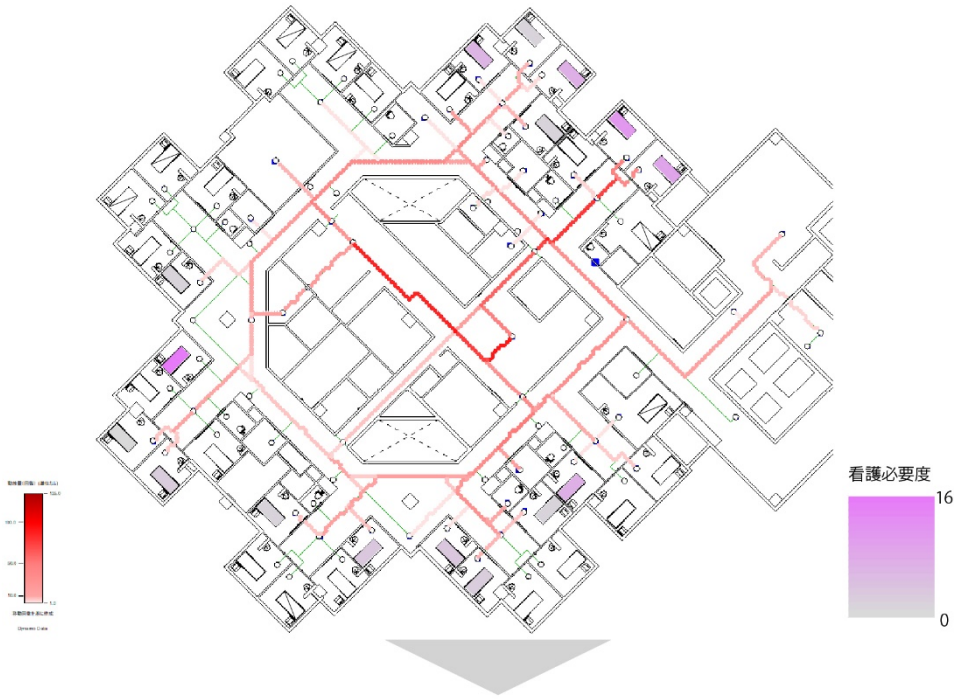
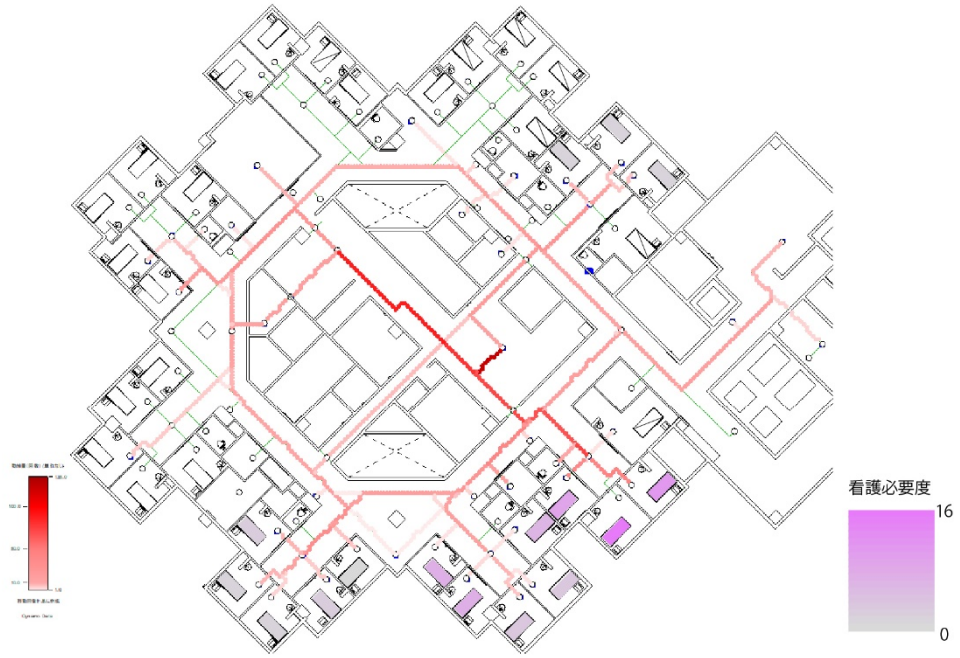


図 4-12　下呂温泉病院最適化した患者配置

西病棟　　長日勤看護師 B　現状配置　(8：30 ～ 21：30)



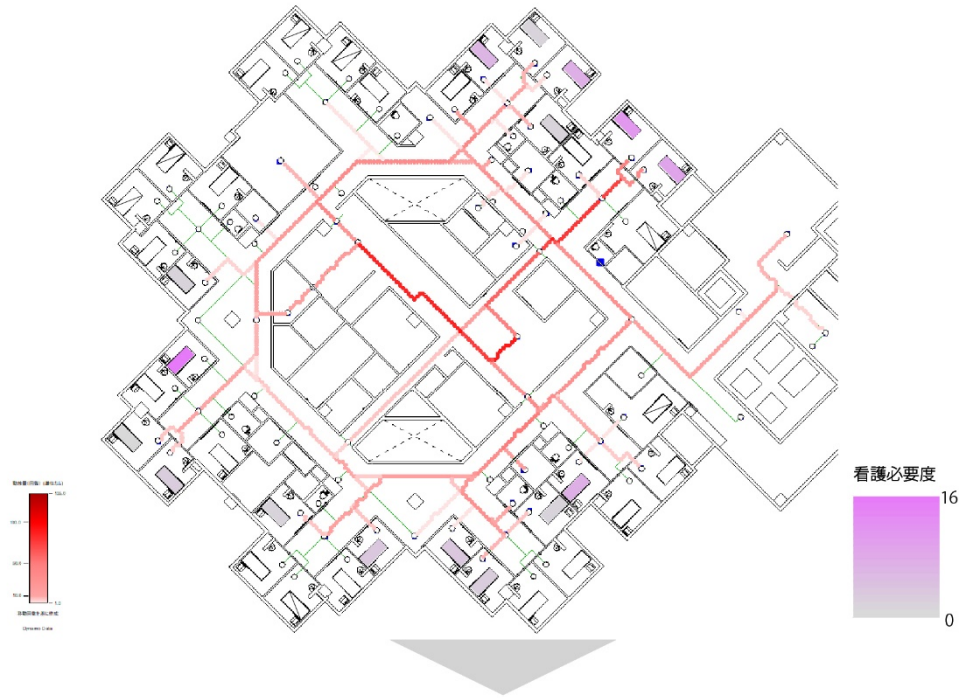
西病棟　　長日勤看護師 B　最適配置　(8：30 ～ 21：30)



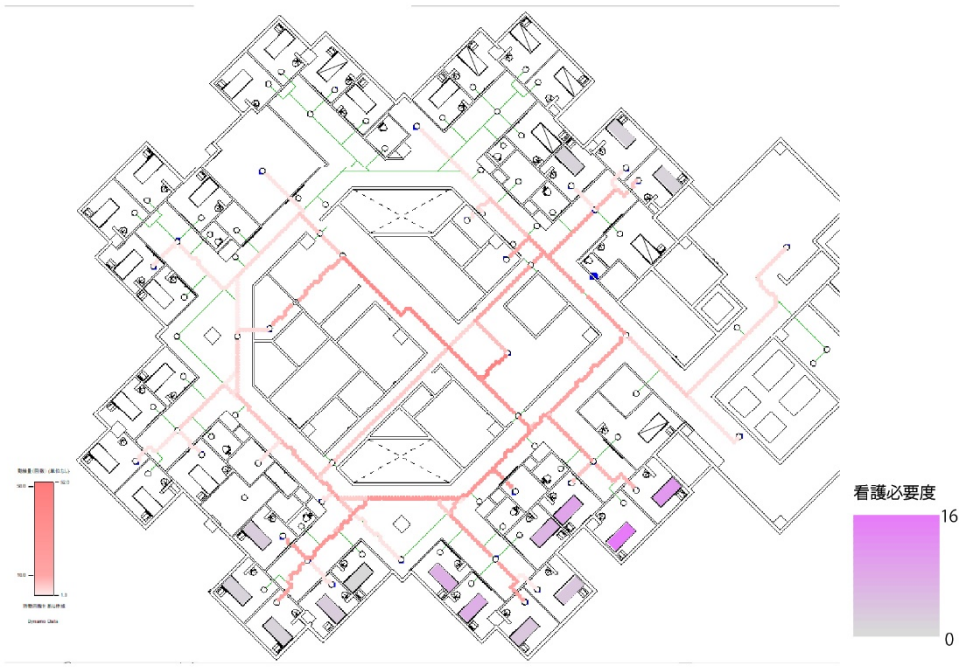
	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	6046.2 m	5216.4 m
看護師総移動時間（秒速 1.25m で換算）	80.6 分	69.6 分

図 4-13　下呂温泉病院西病棟-長日勤看護師 B

西病棟　　日勤看護師 C　現状配置　（8：30 ～ 17:30）



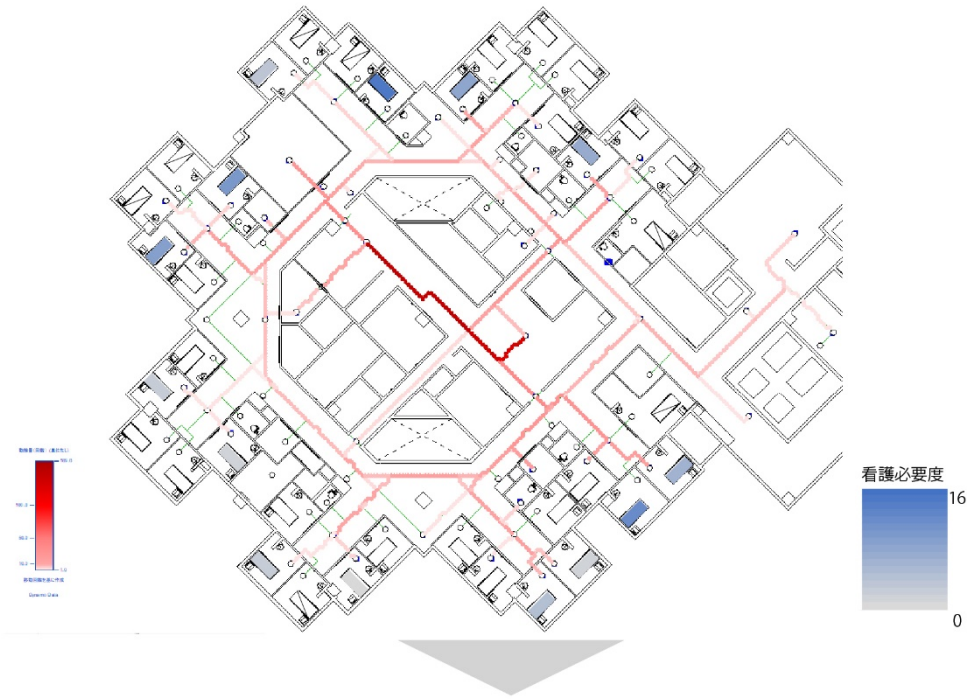
西病棟　　日勤看護師 C　最適配置　（8：30 ～ 17:30）



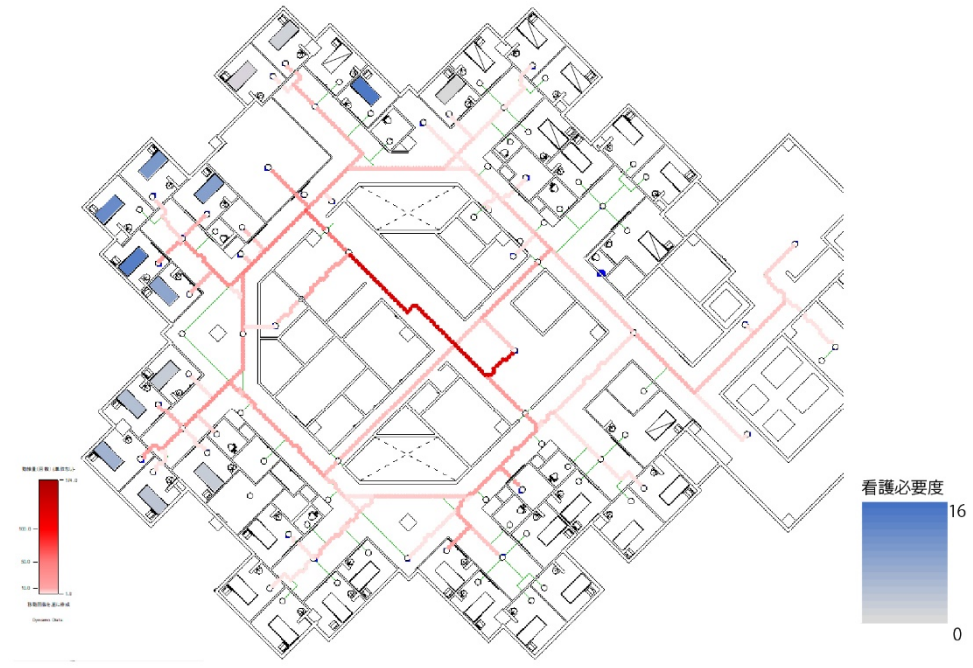
	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	2546.1m	2452.2 m
看護師総移動時間（秒速 1.25m で換算）	33.9 分	32.7 分

図 4-14　下呂温泉病院西病棟-日勤看護師 C

西病棟　　長日勤看護師 A　現状配置　(8：30 ～ 21:30)



西病棟　　長日勤看護師 A　最適配置　(8：30 ～ 21:30)

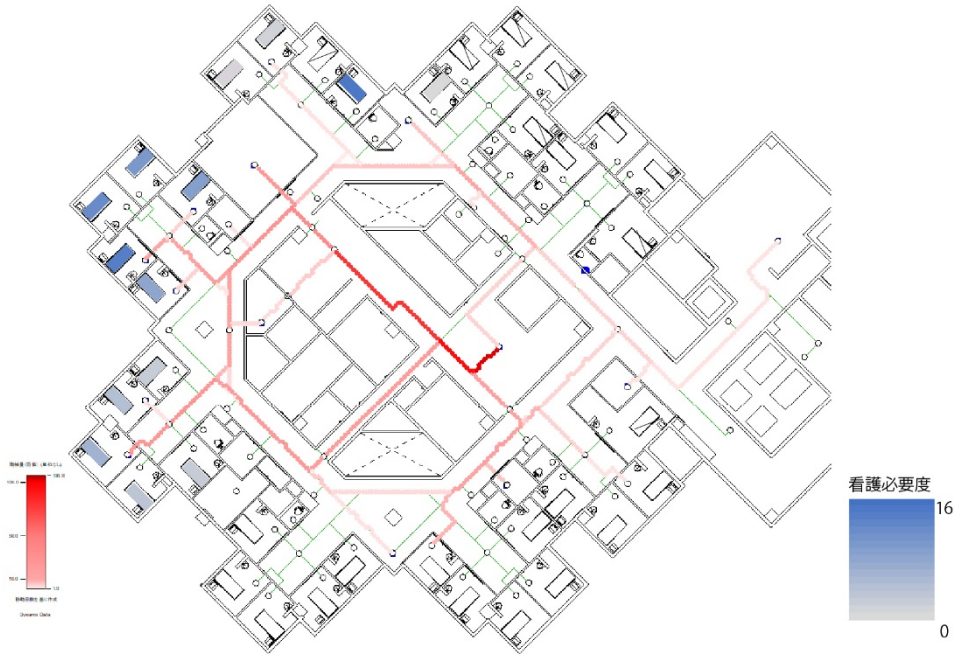


	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	4796.1m	4545.2 m
看護師総移動時間（秒速 1.25m で換算）	63.9 分	60.6 分

図 4-15　下呂温泉病院西病棟-長日勤看護師 A

西病棟　　日勤看護師 D　現状配置　（8：30 ～ 17:30）

西病棟　　日勤看護師 D　最適配置　（8：30 ～ 17:30）

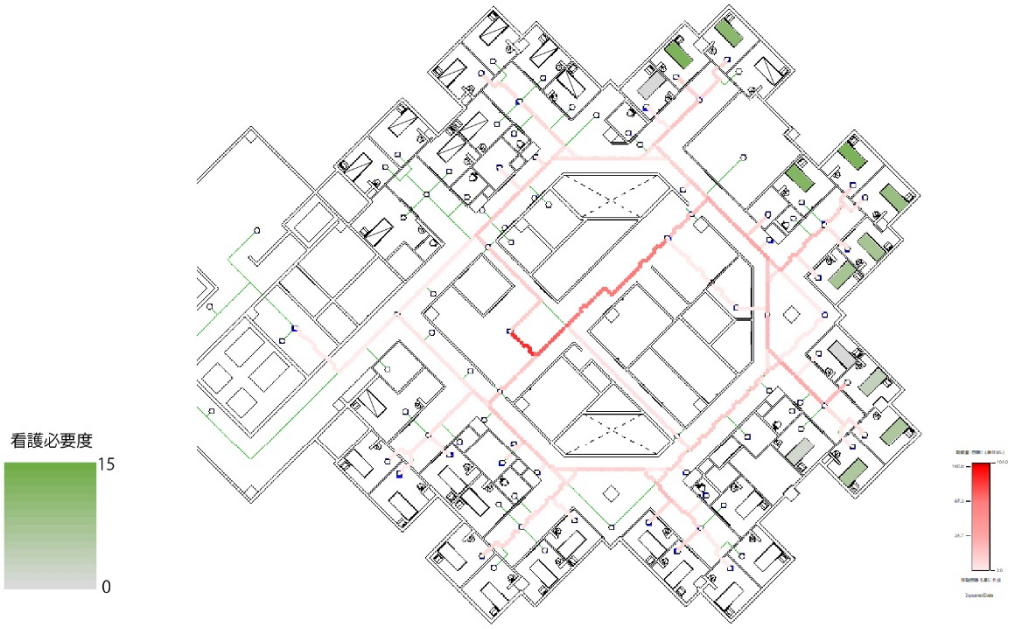


	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	2948.6m	3249.9 m
看護師総移動時間（秒速 1.25m で換算）	39.3 分	43.3 分

図 4-16　下呂温泉病院西病棟-日勤看護師 D

東病棟　　長日勤看護師 B　現状配置　(8：30 ～ 21:30)

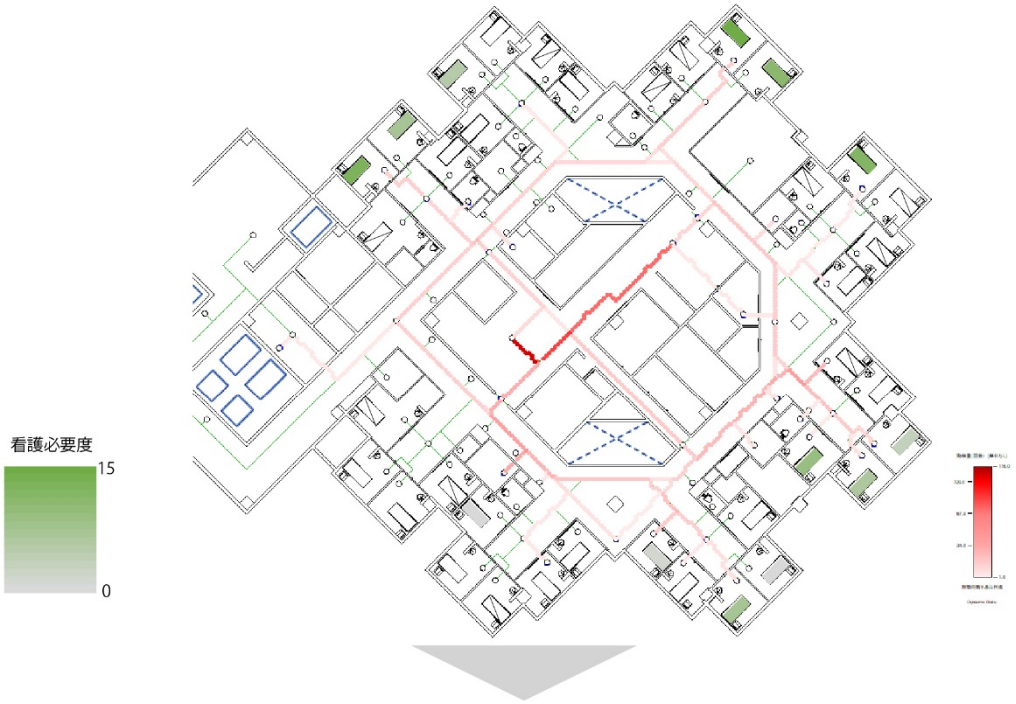
東病棟　　長日勤看護師 B　最適配置　(8：30 ～ 21:30)



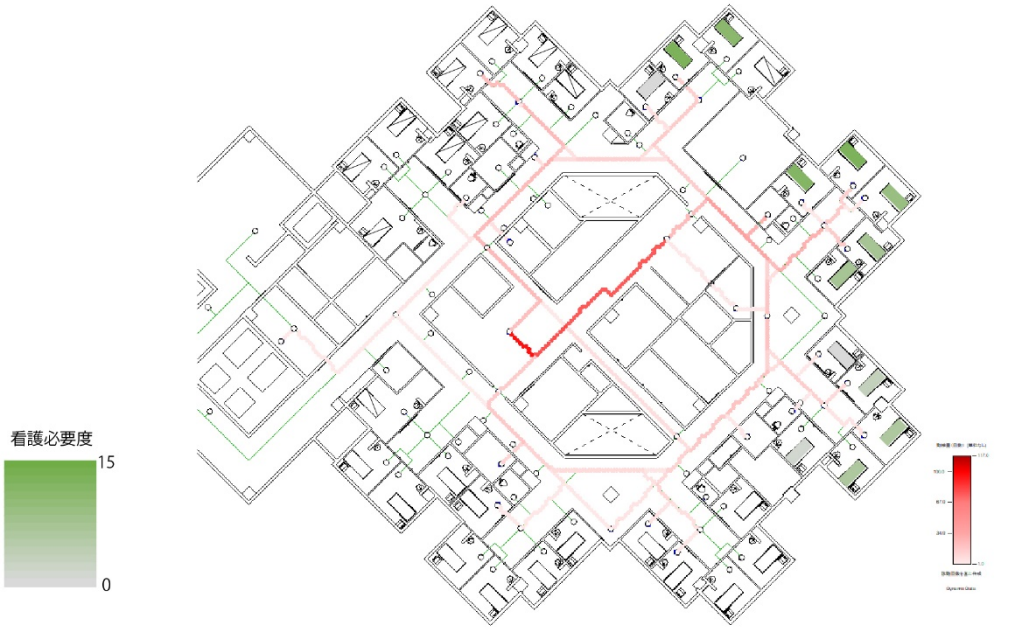
	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	4357.7m	3841.4m
看護師総移動時間（秒速 1.25m で換算）	58.1 分	51.2 分

図 4-17　下呂温泉病院東病棟-長日勤看護師 B

東病棟　　日勤看護師 D　現状配置　(8：30 ～ 17:30)



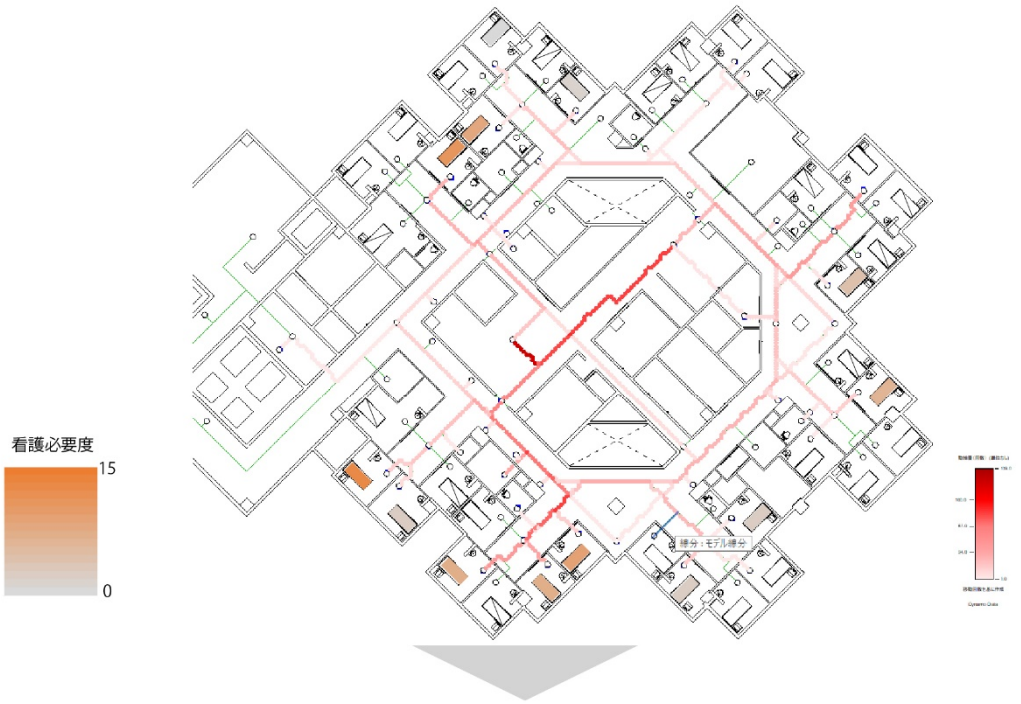
東病棟　　日勤看護師 D　最適配置　(8：30 ～ 17:30)



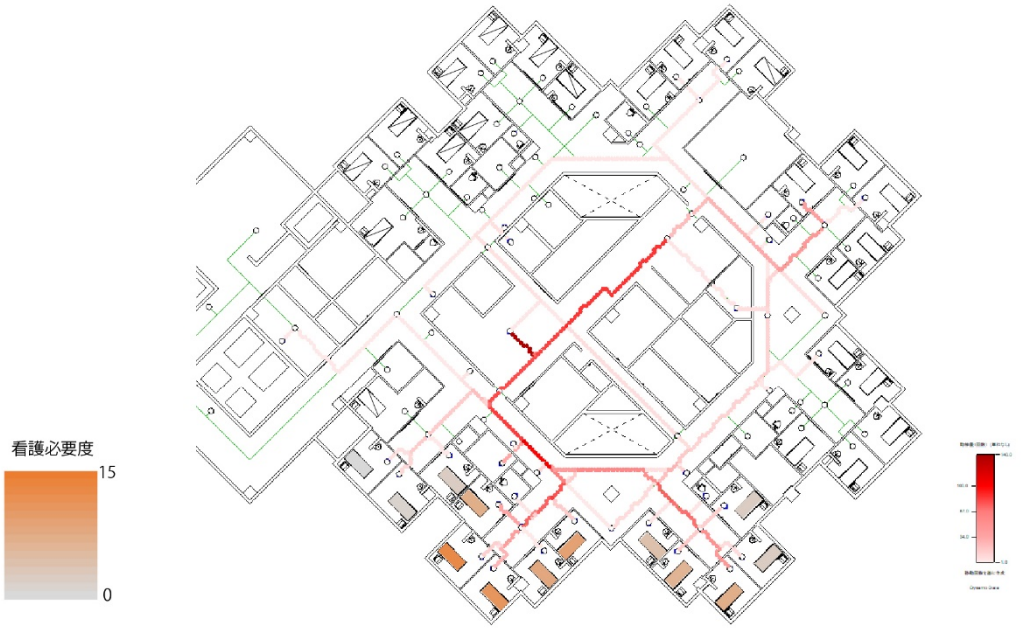
	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	3996.3m	3827.1m
看護師総移動時間（秒速 1.25m で換算）	53.3 分	51.0 分

図 4-18　下呂温泉病院東病棟-日勤看護師 D

東病棟　　長日勤看護師 A　現状配置　(8：30 ～ 21:30)



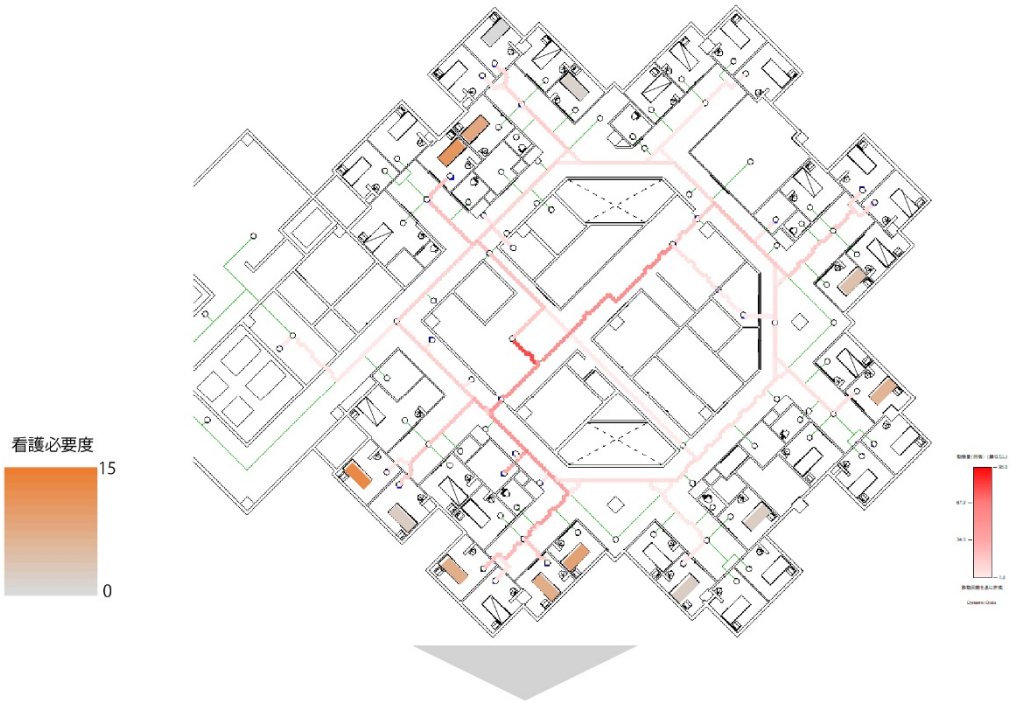
東病棟　　長日勤看護師 A　最適配置　(8：30 ～ 21:30)



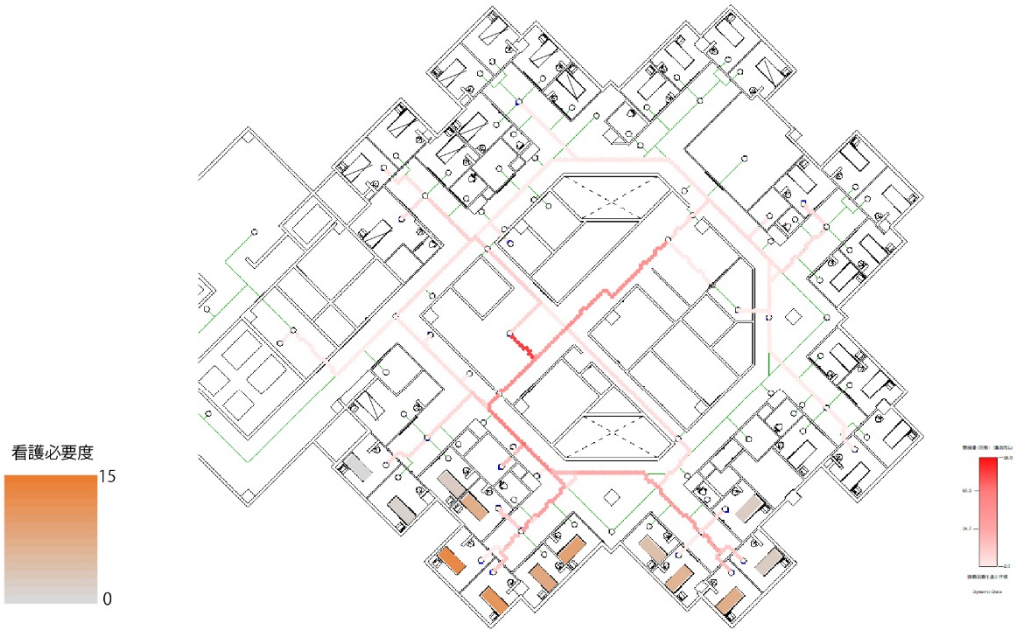
	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	6194.5m	5847.4m
看護師総移動時間（秒速 1.25m で換算）	82.6 分	78.0 分

図 4-19　下呂温泉病院東病棟-長日勤看護師 A

東病棟　　日勤看護師 C　現状配置　(8：30 ～ 17:30)



東病棟　　日勤看護師 C　最適配置　(8：30 ～ 17:30)



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	3611.3m	3390.7m
看護師総移動時間（秒速 1.25m で換算）	48.2 分	45.2 分

図 4-20　下呂温泉病院東病棟-日勤看護師 C

4ー3ー2　シミュレーションの分析

表 4-2 に看護師の病室訪問回数及びシミュレーションから得た移動時間と移動距離、その結果から得た時間の増減差と距離の増減差、増減率を示す。

現状の西病棟と東病棟の看護動線量を比較すると、日勤は東病棟の方が多くなっていることが分かる。長日勤の看護動線量にはほぼ差がないが病室訪問回数に大きく差が生まれた。これにより西病棟の SS での移動が頻繁に行われたことや病室間の移動にユニット間の移動や対称のユニットへ移動するために SS を挟んでの長い距離の移動が頻繁に行われたことなどが推測される。若しくは東病棟ではかなり効率的な看護師の移動が行われていた可能性も考えられる。このように SS を中心に 360 度配置された回廊型の病棟では SS から病室までの距離は大きく変わらないが、対称に配置された病室への移動や物品などの置かれた特定の部屋への移動が看護動線量に大きく影響を与えていると考えられる。また、両病棟ともに長日勤 A・B は看護動線量と病室訪問回数に顕著な差が見受けられた。片方の看護師は看護動線距離が 6km を超える結果となり、非常に長くなったが病室訪問回数も多くなっていることから、これは単純に病室訪問回数が多くなったことによるものと考えるのが妥当である。

患者配置の最適化により、基本的にチームごとに集約された配置が得られ、ほとんどの看護師で看護動線量が削減される結果となった。最大で約 830m、時間にして 11 分の看護動線量が削減された。削減された看護師の増減率をみると、東病棟の長日勤 A を除き病室訪問回数が多くなるに従い、削減率が大きくなっている。ここからも SS から病室への移動や病室間の移動が看護動線量に大きく影響を与えていることがわかる。東病棟の長日勤 A は看護動線量が削減されて

表 4-2　シミュレーション結果

		現状分析			最適化後の分析		比較分析		
		移動時間（分）	移動距離（m）	病室訪問回数(回)	移動時間（分）	移動距離（m）	時間の増減差（分）	距離の増減差（m）	増減率(%)
西病棟	長日勤B	80.6	6046.2	90	69.6	5216.4	-11.0	-829.8	-13.6
	日勤C	33.9	2546.1	36	32.7	2452.2	-1.2	-93.9	-3.5
	長日勤A	63.9	4796.1	47	60.6	4545.2	-3.3	-250.9	-5.2
	日勤D	39.3	2948.6	28	43.3	3249.9	4.0	301.3	10.2
東病棟	長日勤B	58.1	4357.7	73	51.2	3841.4	-6.9	-516.3	-11.9
	日勤D	53.3	3996.3	51	51.0	3827.1	-2.3	-169.2	-4.3
	長日勤A	82.6	6194.5	121	78.0	5847.4	-4.6	-347.1	-5.6
	日勤C	48.2	3611.3	57	45.2	3390.7	-3.0	-220.6	-6.2

いるものの訪問回数に対して削減率は高くない。シミュレーション結果をみると最適化した患者配置では看護動線が多く訪問している病室に集中しており、各病室もまとめて配置されている。しかし、現状の配置の場合も SS から近い病室に看護動線が集中しており、病室間の移動より SS から病室への移動が多かった可能性を含めると前述したように、既にかなり効率的な看護師の移動が行われていたと推測される。さらに、全ての看護師で担当患者ではない病室や空病室への移動が行われているが、東病棟の長日勤 A では比較的その移動が多かったことも影響していると考えられる。一方で、西病棟の日勤 D では看護動線量が大きく増加している。この看護師は病室訪問回数が少なかったことに加え、東病棟の長日勤 A と同様に比較的担当外の患者や空病室への移動が多かった。担当患者の病室への移動回数に対して担当外の病室への移動回数が多いため、両チームでバランスよく配置されていた現状の配置より両チームで偏った配置を行った最適化後で、看護動線量が大きくなったと考えられる。

4－4 両病院のシミュレーション分析のまとめ

4-2 及び 4-3 で、調査対象両病院の現状の配置と最適化後の配置で看護動線シミュレーションを行った。そこで、以下の結果が得られた。

・碧南市民病院

① 碧南市民病院のような看護拠点から病室の距離にばらつきがある病棟では、担当患者の配置と SS から病室の距離に差が看護動線量に大きく差が出る。

② 現状の患者配置はチームのまとまりという観点からは最適に近い配置がされていた。

③ 最適化後の患者配置はチームのまとまりにはあまり変更は見られなかったが、看護必要度により配置が変化した。看護必要度の高さからくる配置の最適化により、ほとんどの看護師で大幅に看護動線量が削減された。

④ 今回行った最適化方法では、病室訪問回数と担当外の病室への移動回数が増減率に大きく影響を与えている。

・下呂温泉病院

① 下呂温泉病院のような SS を中心に 360 度配置された回廊型の病棟では SS から病室までの距離は大きく変わらないが、対称に配置された病室への移動や物品などの置かれた特定の部屋への移動が看護動線量に大きく影響を与えている。

② 最適化後の患者配置は、基本的にチームごとに集約された配置が得られ、ほとんどの看護師で病室訪問回数が多くなるに従い、削減率が高くなった。SS

から病室への移動や病室間の移動が看護動線量に大きく影響を与えていることがわかった。

③ 今回行った最適化方法では、担当外の病室への移動回数が増減率に影響を与えている。

碧南市民病院のような看護拠点から病室までの距離にばらつきのある病棟と、下呂温泉病院のような看護拠点から病室までの距離にあまり差のない病棟では、看護動線量の増減に異なる影響を受けることが確認された。また、最適化後の削減率にも異なる影響を受けている。碧南市民病院では削減された看護師では平均的で大きく削減され、下呂温泉病院では訪問回数が多くなるに従い削減率が大きくなったが、訪問回数の多くない看護師では削減率はかなり少ないものとなった。看護師の業務はナースコールなどで SS から直接病室を訪問し、SS へと戻るといった移動も少なくない。SS から病室への距離は、下呂温泉病院はほとんど差がないため、少ない訪問回数では病室間の移動では看護動線量に差が現れるが、SS と病室での移動では差が現れない。そのため訪問回数に従い削減率が変化し、少ない訪問回数では現状とほとんど変わらない看護動線量になっていると考えられる。

今回行った最適化方法は、SS からの距離・病室間の距離・看護必要度の高低・チームのまとまりを考慮し最適化するものである。両病院でのシミュレーションから今回の最適化方法では担当外の病室や空病室への移動がある場合、看護動線量が増える例も見られた。特に碧南市民病院では、担当外の患者病室への移動が顕著にみられた。

4－5 チームによる相互関係を含めない場合

4－5－1 最適配置とシミュレーションの結果

前節までで行った最適化方法では、担当外の病室や空病室への移動がある場合、看護動線量が増える可能性あることが確認された。SS からの距離・病室間の距離・看護必要度の高低・チームのまとまりを考慮し最適化されていた。しかし、碧南市民病院では西病棟の A チームの看護師以外、下呂温泉病院では全ての看護師で担当外の病室や空病室への移動が発生していた。本節ではチームのまとまりを表す相互関係行列を全て同じ値として最適化を行う。チームのまとまりは考慮せずに最適化した場合の患者配置に変更し、看護動線シミュレーションを行う。図 4-21 に碧南市民病院、図 4-22 に下呂温泉病院、それぞれの最適化後の患者配置を示す。さらに図 4-23～4-30 に各病棟の各看護師に対するシミュレーションの結果を示す。

図 4-21　碧南市民病院最適化した患者配置

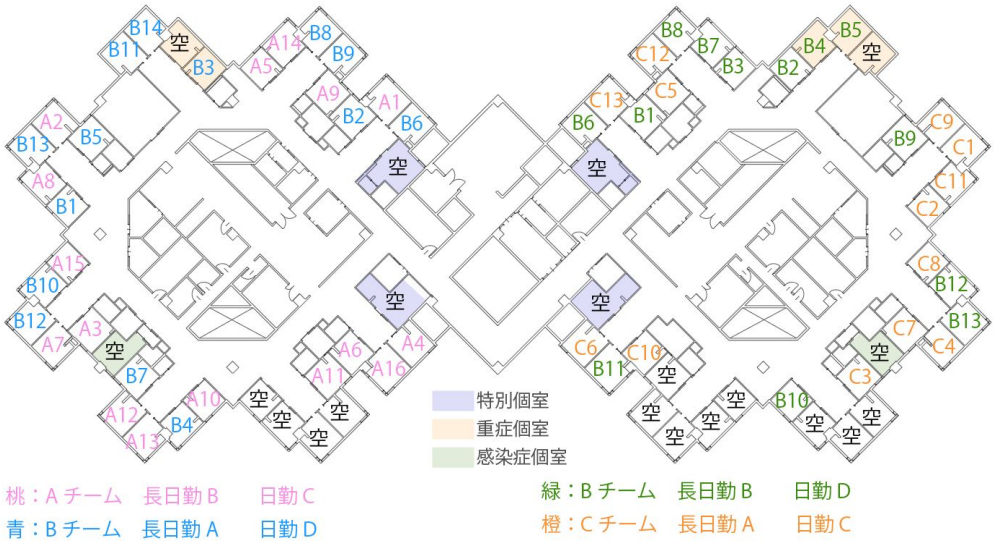
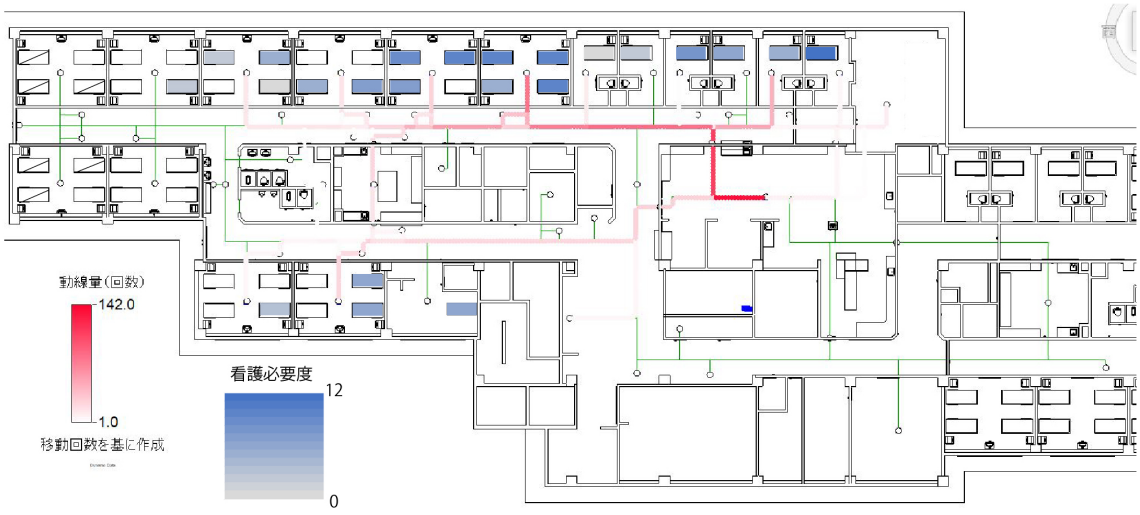


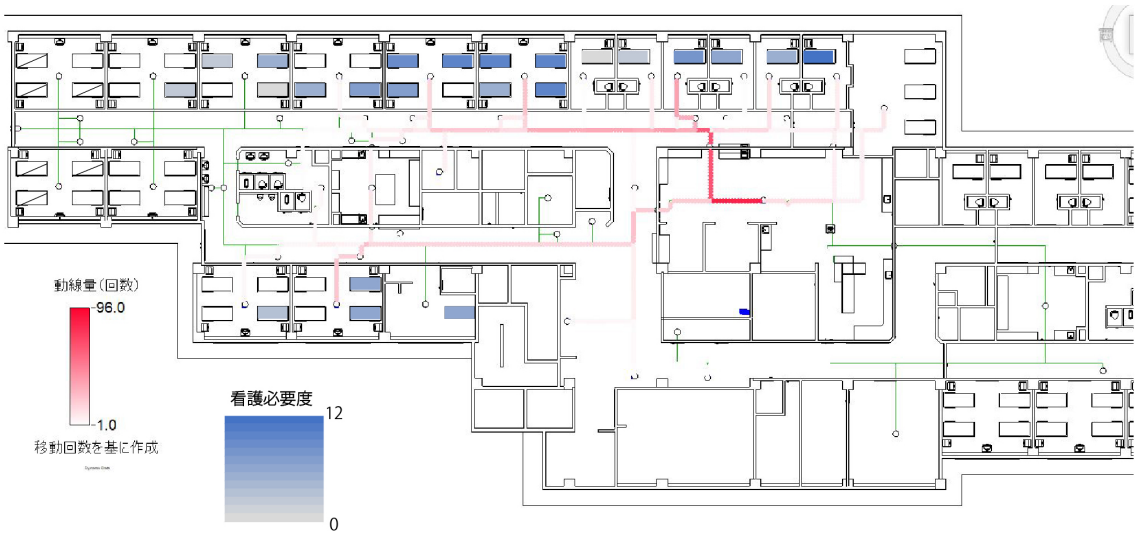
図 4-22　下呂温泉病院最適化した患者配置

西病棟　　日中看護師Ｆ　相互関係無視　（8：30 ～ 21：30）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	4852.5 m	4473.5 m
看護師総移動時間（秒速 1.25m で換算）	64.7 分	59.6 分

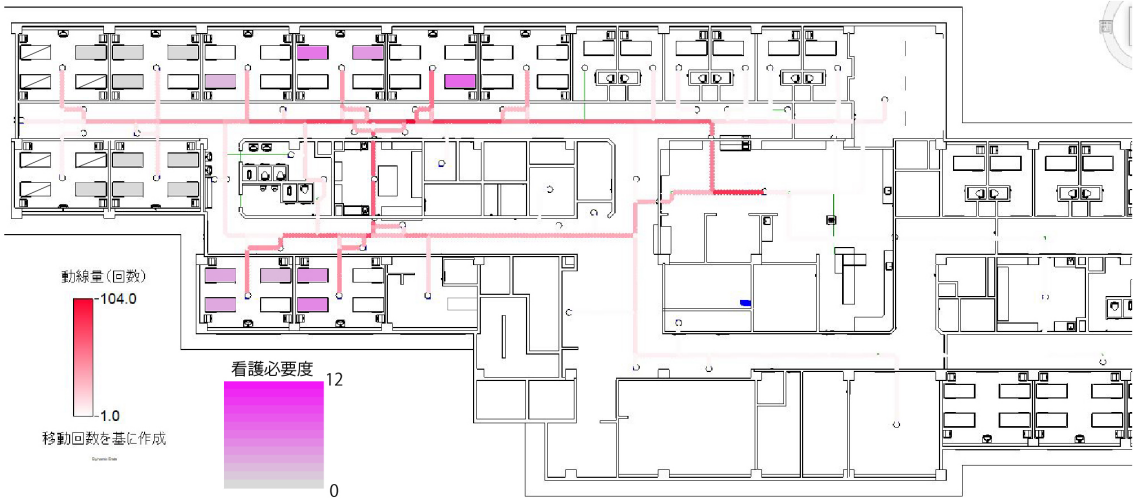
西病棟　　日勤看護師Ｅ　相互関係無視　（8：30 ～ 17：15）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	3057.5 m	2616.8 m
看護師総移動時間（秒速 1.25m で換算）	40.8 分	34.9 分

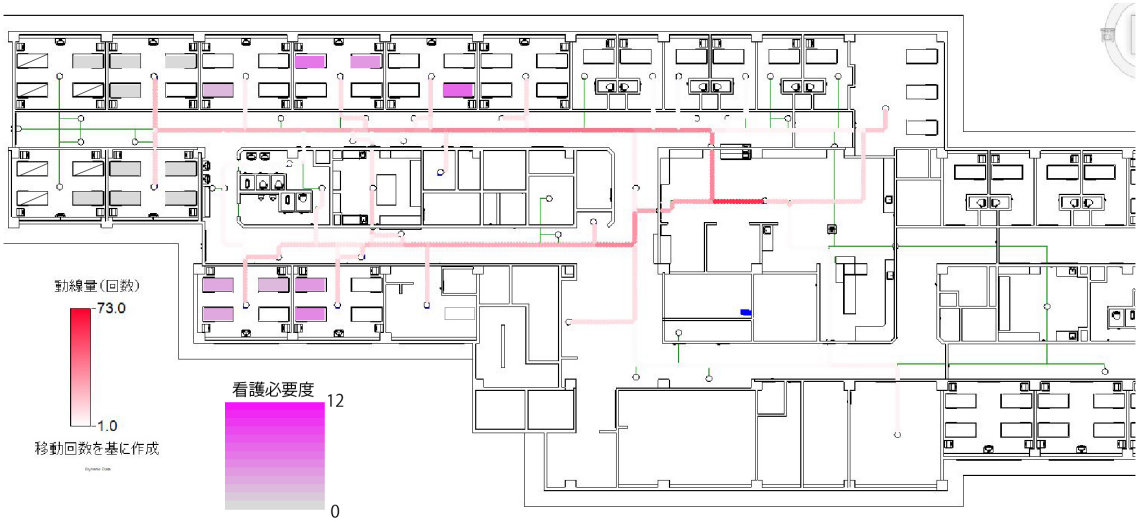
図 4-23　碧南市民病院西病棟-（上：日中看護師 F　下：日勤看護師 E）

西病棟　　日中看護師 D　相互関係無視　（8：30 ～ 21：30）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	7371.0 m	7062.8 m
看護師総移動時間（秒速 1.25m で換算）	98.3 分	94.2 分

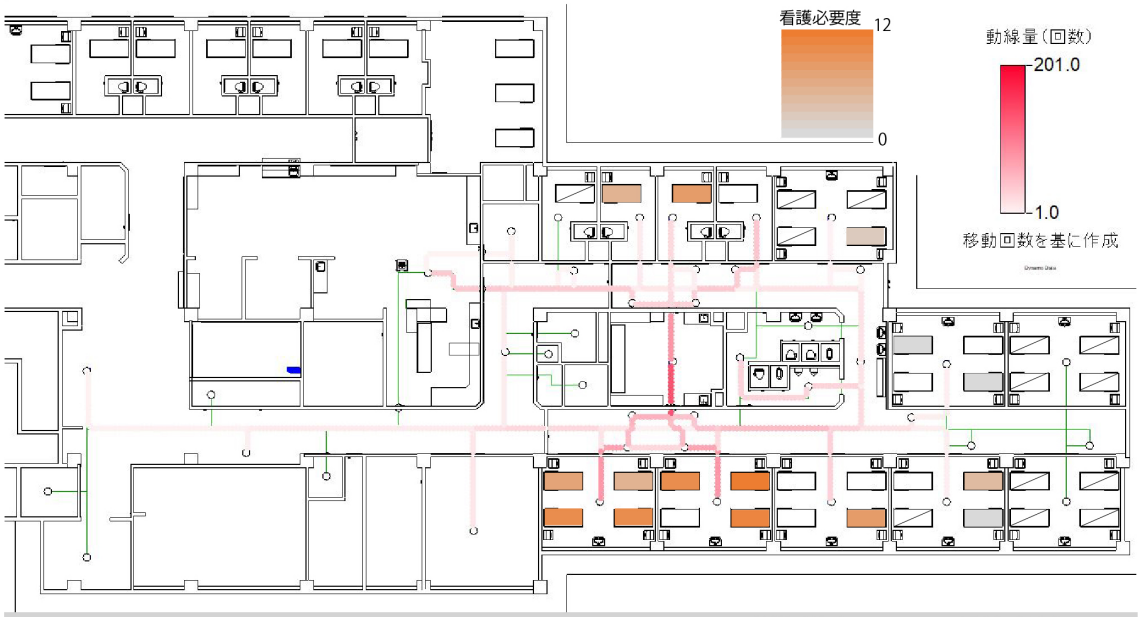
西病棟　　日勤看護師 C　相互関係無視　（8：30 ～ 17：15）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	3841.8 m	3265.8 m
看護師総移動時間（秒速 1.25m で換算）	51.2 分	43.5 分

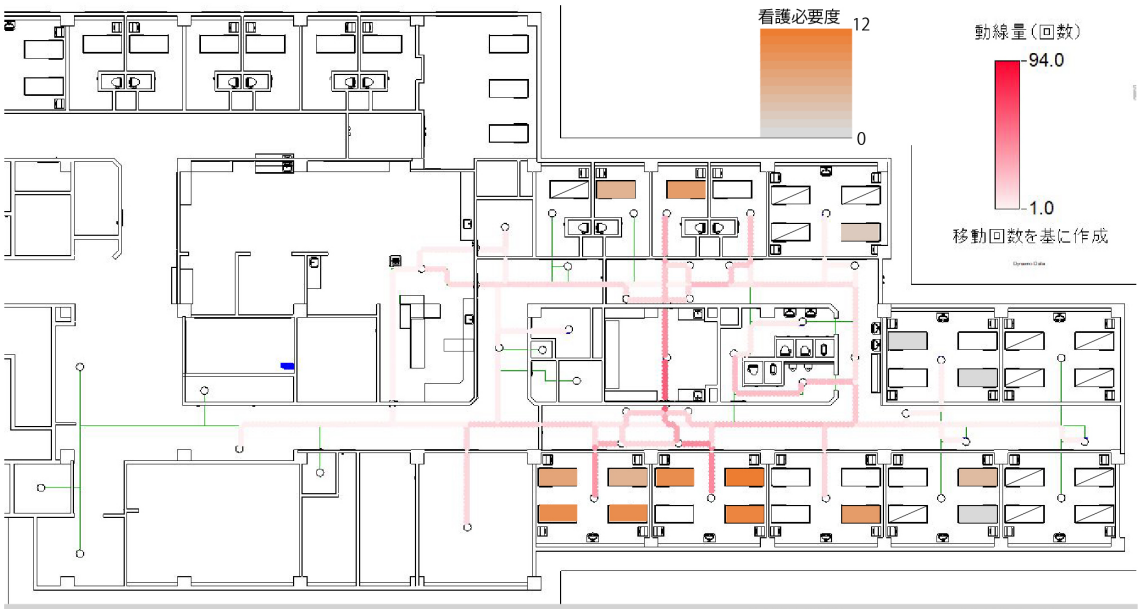
図 4-24　碧南市民病院西病棟-（上：日中看護師 D　下：日勤看護師 C）

東病棟　　日中看護師 D　相互関係無視　（8：30 ～ 21：30）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	4209.3 m	3869.6 m
看護師総移動時間（秒速 1.25m で換算）	56.1 分	51.6 分

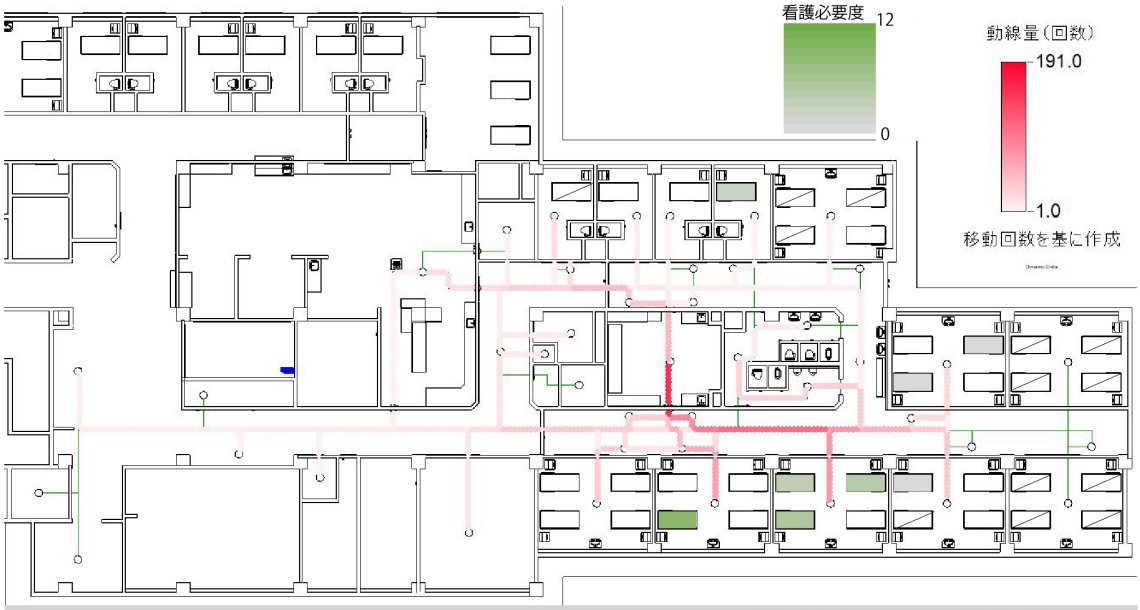
東病棟　　日勤看護師 C　相互関係無視　（8：30 ～ 17：15）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	2435.4 m	2051.8 m
看護師総移動時間（秒速 1.25m で換算）	32.5 分	27.4 分

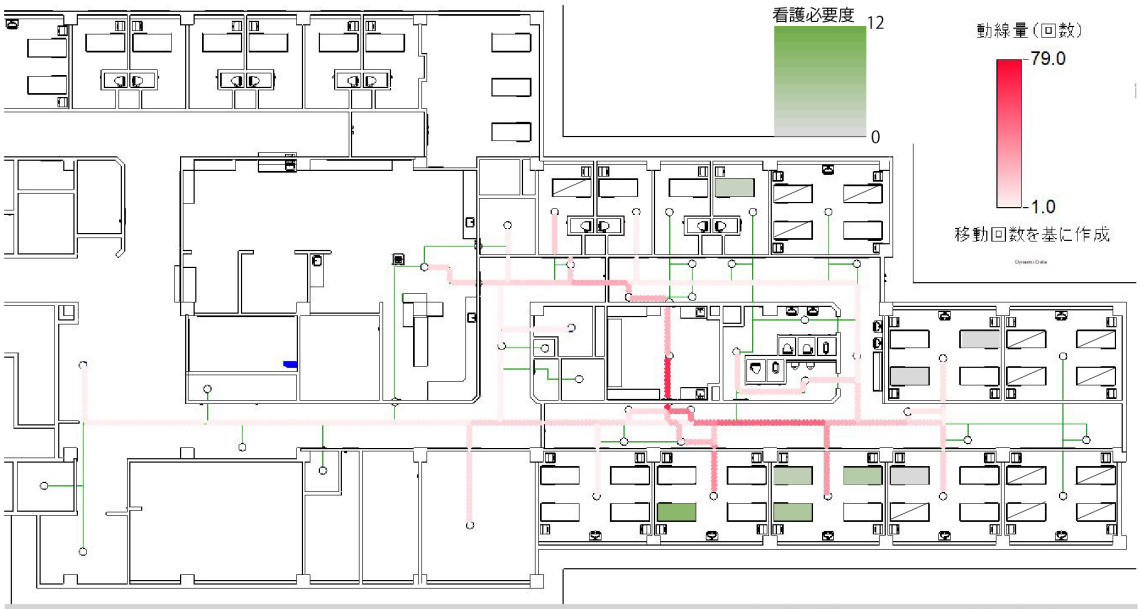
図 4-25　碧南市民病院東病棟-（上：日中看護師 D　下：日勤看護師 C）

東病棟　　日中看護師Ｆ　相互関係無視　（8：30 ～ 21：30）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	5049.1 m	4390.1 m
看護師総移動時間（秒速 1.25m で換算）	67.3 分	58.5 分

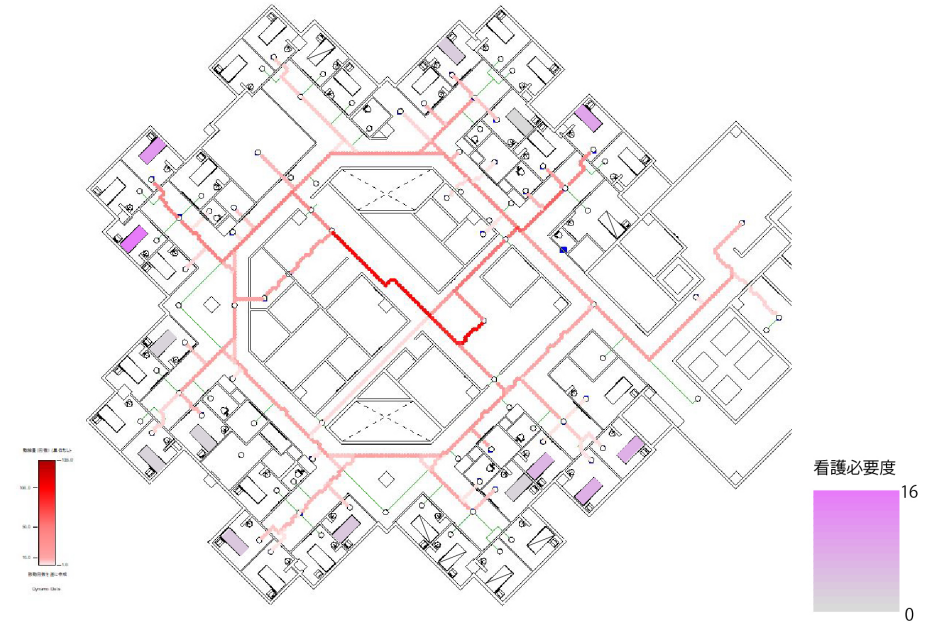
東病棟　　日勤看護師Ｅ　相互関係無視　（8：30 ～ 17：15）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	2085.7 m	1687.7 m
看護師総移動時間（秒速 1.25m で換算）	27.8 分	22.5 分

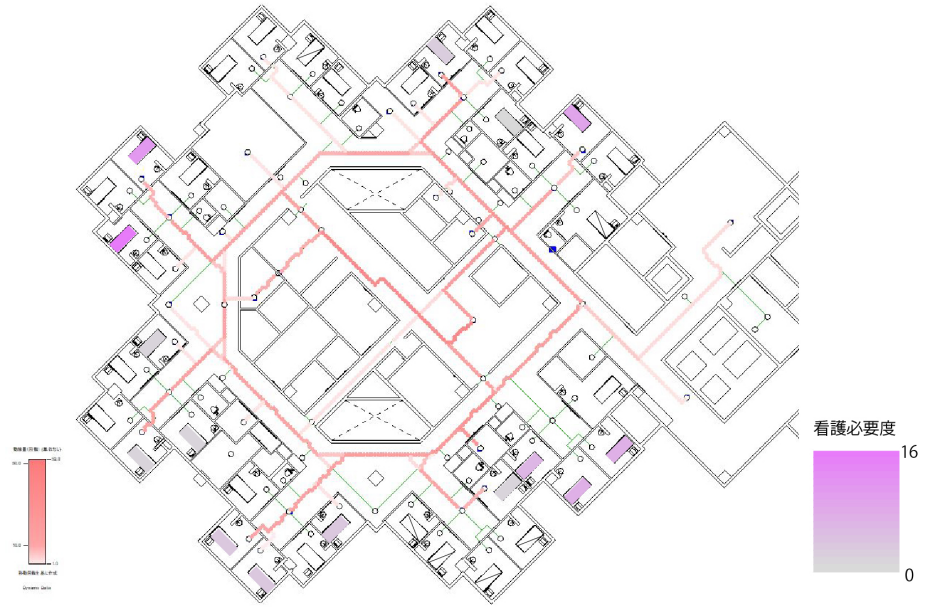
図 4-26　碧南市民病院東病棟-（上：日中看護師 F　下：日勤看護師 E）

西病棟　　長日勤看護師 B　相互関係無視　（8：30 ～ 21：30）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	6046.2 m	5789.9m
看護師総移動時間（秒速 1.25m で換算）	80.6 分	77.2 分

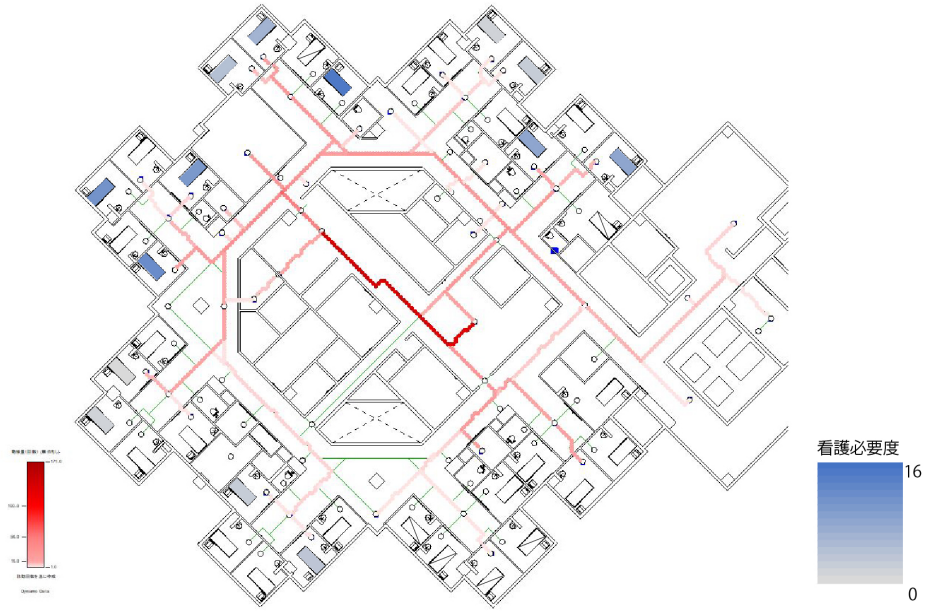
西病棟　　日勤看護師 C　相互関係無視　（8：30 ～ 17：30）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	2546.1 m	2772.0 m
看護師総移動時間（秒速 1.25m で換算）	33.9 分	37.0 分

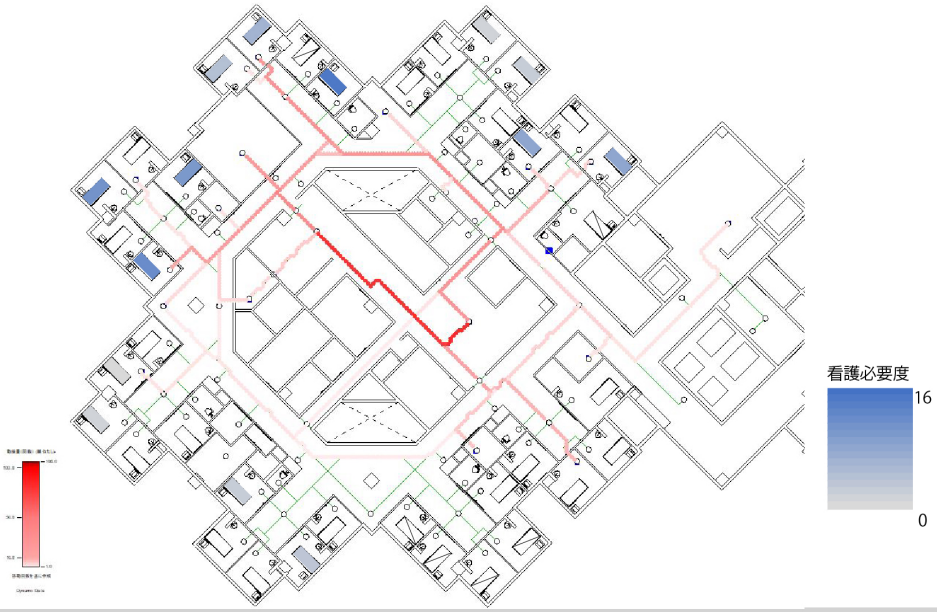
図 4-27　下呂温泉病院西病棟-（上：長日勤看護師 B　下：日勤看護師 C）

西病棟　　長日勤看護師 A　相互関係無視　（8：30 ～ 21：30）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	4796.1 m	4743.1m
看護師総移動時間（秒速 1.25m で換算）	63.9 分	63.2 分

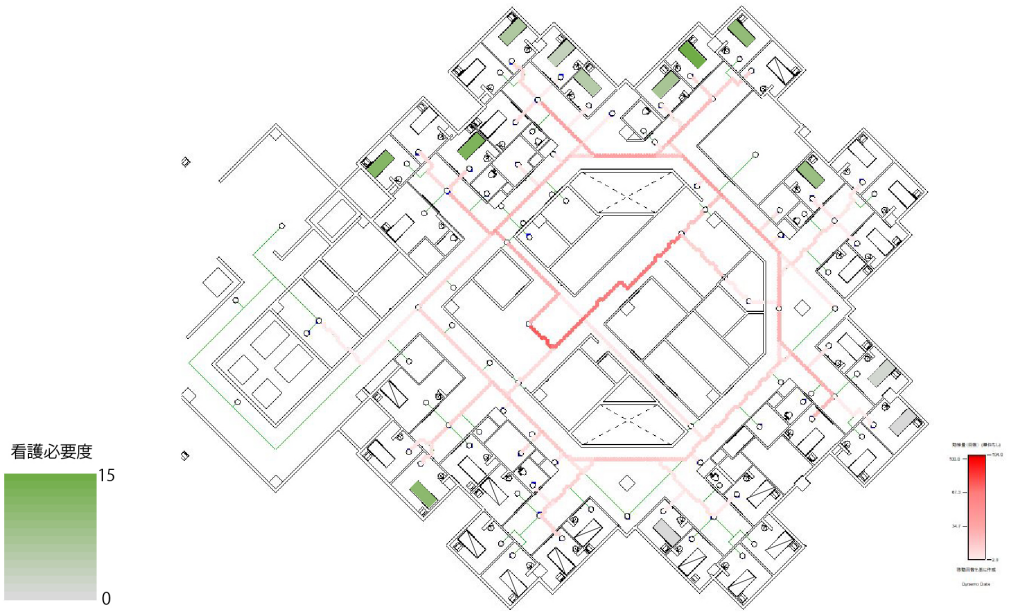
西病棟　　日勤看護師 D　相互関係無視　（8：30 ～ 17：30）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	2948.6 m	3084.5 m
看護師総移動時間（秒速 1.25m で換算）	39.3 分	41.1 分

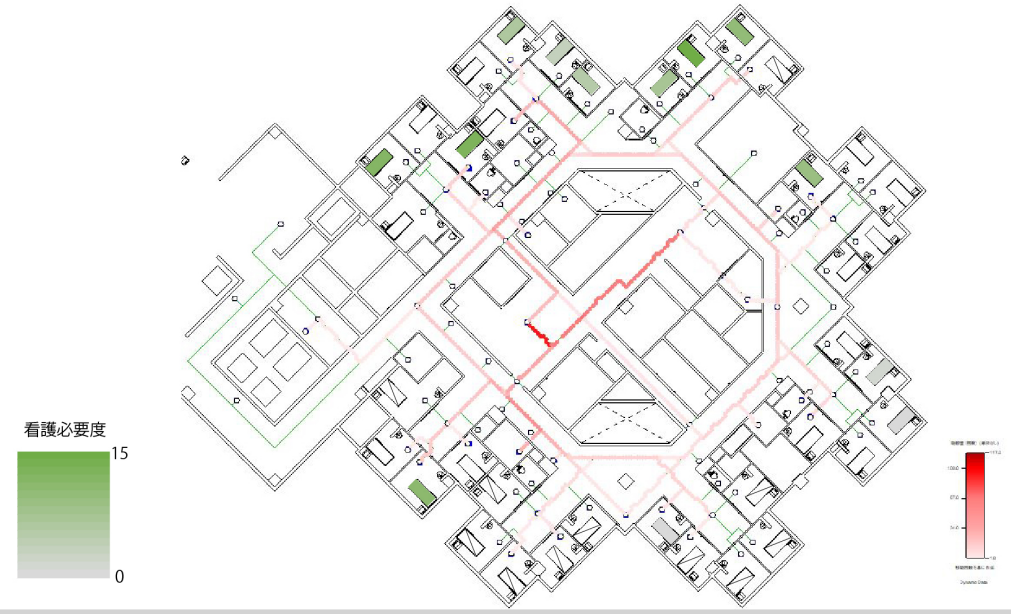
図 4-28　下呂温泉病院西病棟-（上：長日勤看護師 A　下：日勤看護師 D）

東病棟　　長日勤看護師 B　相互関係無視　（8：30 ～ 21：30）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	4357.7 m	4594.8 m
看護師総移動時間（秒速 1.25m で換算）	58.1 分	61.3 分

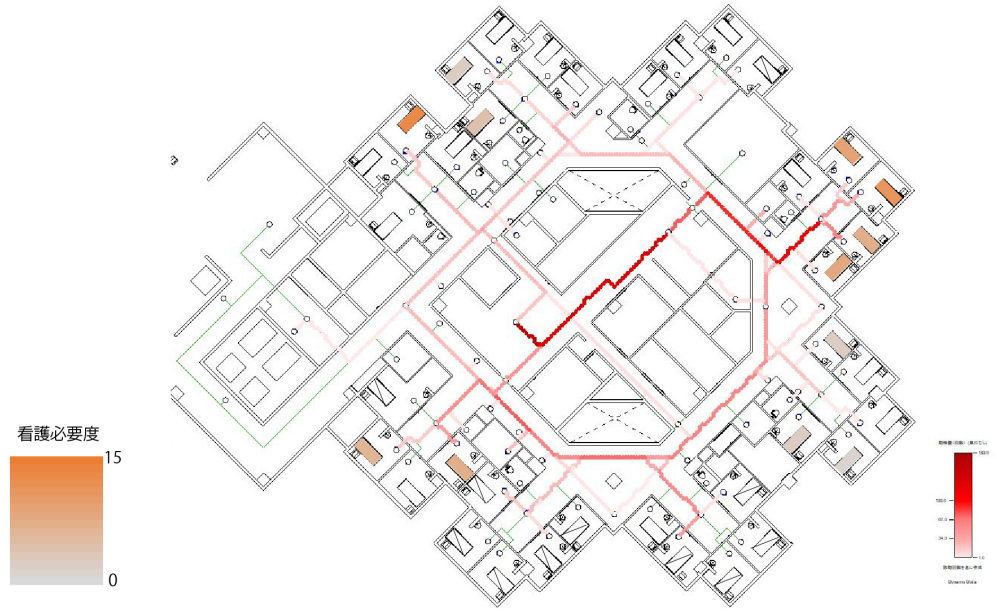
東病棟　　日勤看護師 D　相互関係無視　（8：30 ～ 17：30）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	3996.3 m	3797.5 m
看護師総移動時間（秒速 1.25m で換算）	53.3 分	50.6 分

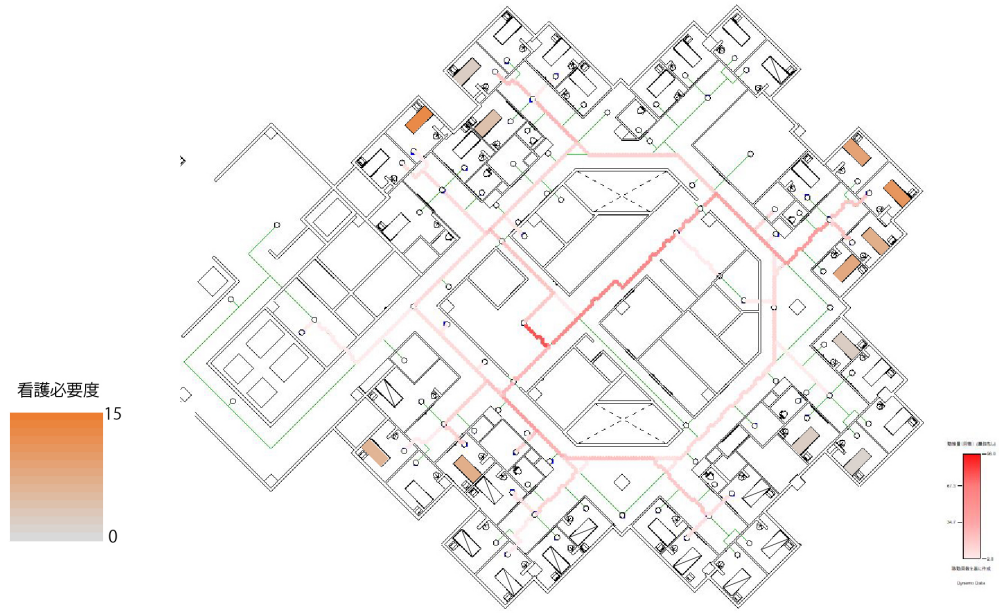
図 4-29　下呂温泉病院東病棟-（上：長日勤看護師 B　下：日勤看護師 D）

東病棟　　長日勤看護師 A　相互関係無視　（8：30 ～ 21：30）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	6194.5 m	9208.7m
看護師総移動時間（秒速 1.25m で換算）	82.6 分	122.8 分

東病棟　　日勤看護師 C　相互関係無視　（8：30 ～ 17：30）



	現状配置	最適配置
看護師総移動距離（看護拠点内の移動は無視）	3611.3 m	4151.8 m
看護師総移動時間（秒速 1.25m で換算）	48.2 分	55.4 分

図 4-30　下呂温泉病院東病棟-（上：長日勤看護師 A　下：日勤看護師 C）

4ー5ー2 シミュレーションの分析

表 4-3 に碧南市民病院のシミュレーション結果から得た、現状配置からの時間の増減差、距離の増減差、増減率を示す。

結果を見ると、全ての看護師で看護動線量が削減されていることが分かる。しかし、西病棟では 4-2 での配置と増減率を比較すると、日中 D 以外は看護動線量が増加した。特に、日中 F は看護動線量の増加が顕著であった。病室訪問回数が多いことや担当外の患者が入室している病室への移動がほとんど無かったことによるものであると考えられる。日中 D は看護動線量が若干削減される結果となったが、ほかの看護師と比較しても増減率はあまり高くない。東病棟では 4-2 での配置の増減率と比較すると、A チームの看護師は看護動線量が減少したが、B チームの看護師は看護動線量が増加した。4-2 での患者配置では個室に A チームの患者が配置されていることから、比較的患者が離れて配置されていた。本節の配置では、個室側の 4 床室に患者があまり配置されず、患者同士が近づいたことにより看護動線量が減少したと考えられる。それにより B チームの患者は離れて配置されたため、看護動線量は増加している。

表 4-3　シミュレーション結果(碧南市民病院)

		病室訪問回数(回)	4-2での削減率	配置変更後の分析		比較分析		
				移動時間（分）	移動距離（m）	時間の増減差（分）	距離の増減差（m）	増減率(%)
西病棟	日中F	124	-16.2	59.6	4473.5	-5.1	-379.0	-7.9
	日勤E	73	-16.4	34.9	2616.8	-5.9	-440.7	-14.5
	日中D	201	1.9	94.2	7062.8	-4.1	-308.2	-4.2
	日勤C	68	-19.5	43.5	3265.8	-7.7	-576.0	-15.0
東病棟	日中F	133	-12.5	58.5	4390.1	-8.8	-659.0	-13.1
	日勤E	58	-14.7	22.5	1687.7	-5.3	-398.0	-19.1
	日中D	143	-11.1	51.6	3869.6	-4.5	-339.7	-8.0
	日勤C	81	-16.9	27.4	2051.8	-5.1	-383.6	-15.7

表 4-4 に下呂温泉病院のシミュレーション結果から得た、現状配置からの時間の増減差、距離の増減差、増減率を示す。

結果を見ると、4-3 の配置における増減率と比較すると、ほとんどの看護師で看護動線量が増加している。また、5 人の看護師で増減率が+の値を取った。西病棟の看護師及び東病棟の A チームは時間の増減差や増減率からも、現状から大きくは増減していない。しかし、東病棟の B チームは看護動線量が大

きく増加している。患者配置やシミュレーション結果をみると、訪問回数が多い病室が、対称のユニットに配置されていることが分かる。それにより、現状の配置より移動距離が長くなったと考えられる。

表 4-4　シミュレーション結果(下呂温泉病院)

		病室訪問回数(回)	4-3での削減率	配置変更後の分析		比較分析		
				移動時間（分）	移動距離（m）	時間の増減差（分）	距離の増減差（m）	増減率(%)
西病棟	長日勤B	90	-13.6	77.2	5789.9	-3.4	-256.3	-4.2
	日勤C	36	-3.5	37.0	2772.0	3.1	225.9	9.1
	長日勤A	47	-5.2	63.2	4743.1	-0.7	-53.0	-1.1
	日勤D	28	10.2	41.1	3084.5	1.8	135.9	4.6
東病棟	長日勤B	73	-11.9	61.3	4594.8	3.2	237.1	5.5
	日勤D	51	-4.3	50.6	3797.5	-2.7	-198.8	-5.1
	長日勤A	121	-5.6	100.1	7508.7	17.5	1314.2	21.2
	日勤C	57	-6.2	55.4	4151.8	7.2	540.5	14.9

4－6 まとめ

本章では 3 章で作成した患者配置の最適化アルゴリズムを用いて、相互関係を含めた場合の患者配置と相互関係を含めない場合の患者配置の最適解を得た。さらに、BIM 上で行う看護動線シミュレーションを用いて、看護動線に関する分析を行った。

碧南市民病院で看護師全体の増減率を平均すると、相互関係行列を含めた配置で-13.2%、含めない配置では-12.2%となり、下呂温泉病院では、相互関係行列を含めた配置で-5.0%、含めない配置で+5.6%となった。相互関係行列を全て同じ値として最適化を行った場合、両病院で全看護師における増減率の平均値は増加した。

碧南市民病院では、どちらの配置でも看護動線量は大きく削減されている。しかし、担当外の病室への移動がほとんどみられない西病棟の A チームを見ると、増減率は増加している。特に、病室訪問回数が多い長日勤 B では、その増加が顕著に表れていた。普段から担当外の病室への移動が行われていない場合、看護動線量はさらに削減されると考えられる。普段から行われている場合でも、看護師全体でみれば、相互関係行列を含めた配置で看護動線量が大きく削減されることがわかった。碧南市民病院のような看護拠点から病室の距離に差がある場合や担当外の病室への移動が多い場合でも相互関係行列を含めた患者配置

の最適化が、看護動線の観点からは非常に有効であることが確認できた。

下呂温泉病院では、相互関係行列を含めた配置では看護動線量は削減されたが、相互関係行列を含めない配置では看護動線量は増加した。しかし、その増減幅は約±5%ほどで、碧南市民病院ほど看護動線量は患者配置の変更による影響を受けないことがわかった。SS からそれぞれの病室までの距離に差がないことによるものだと推測できる。ユニットを活用することで改善はされているものの、全室個室であることから病室間の距離には差がある。そのため、SS から病室への移動と病室間の移動の頻度の差によって看護動線量に差が出ることも考えられる。下呂温泉病院のような SS を中心とした回廊型の病棟でも、相互関係行列を含めた患者配置の最適化が、看護動線の観点からは非常に有効であることが確認できた。

本章で、患者配置の最適化アルゴリズムと BIM を用いたシミュレーションを連携し、複数の看護配置とその看護動線量を得ることができた。本研究では、対象病院における実際の看護動線でシミュレーションを行った。しかし、設計段階でシミュレーションを行う場合、実際の看護動線を用いることはできない。そのため、複数の病棟の看護動線調査を通し、より一般的な看護動線モデルを作成する必要があると考えられる。

第5章

総括

5－1 本論文のまとめ

本論文は、本研究では BIM を用いたシミュレーションの開発及び活用方法の提案を行うことで 、建築計画分野において BIM を有効に活用するための知見を得ること、今後の BIM 活用に関する可能性を検討することを目的とした。

2 章では、米国における病院建築の作品集から、いくつかの病室を取り上げ、色彩計画や設えについて分析を行った。また、その事例を参考に病室パースを作成、それぞれの病室パースについて SD 法による印象評価を行うことで居住者の心理的評価を客観的に把握した。得られた結果の因子分析により①病室空間に対する親近性や居心地からなると解釈できる因子、②身体で感じる感覚と結びついていると解釈できる因子の 2 つの主要因子が抽出された。また、病室の構成要素を選択することで簡易的な病室のデザインを作成することのできるアルゴリズムを作成した。病室デザインについて容易に把握できるだけでなく、照度や空調などの各種シミュレーションを行うモデルを作成する段階を簡略化することができると考えられる。

3 章では、看護師の動線量の観点から見て最適な患者配置を得るために、二次割り当て問題及び遺伝的アルゴリズムを用いたアルゴリズムを作成した。また、病室間の距離・看護師チームの受け持ち患者から仮定した移動頻度・SS から病室の距離・患者の看護必要度を評価値として最適化を行い、4 床室への男女の患者配置についても適応可能であることを示した。さらに、連携が想定される BIM シミュレーションに対しても適用が可能であることを確認した。

4 章では、碧南市民病院及び下呂温泉病院の看護動線調査で得た現状の患者配置における看護動線量と、3 章で得た最適化された患者配置における看護動線量を比較した。そこで、碧南市民病院のような看護拠点から病室までの距離にばらつきのある病棟と、下呂温泉病院のような看護拠点から病室までの距離にあまり差のない病棟では、看護動線量の増減に異なる影響を受けることが確認された。また、その患者配置では、担当外の病室や空病室の移動がある場合、看護動線量が増加する可能性が確認されたため、評価値を異なる設定で最適化し、新たに看護動線量を得た。それらを比較することで、看護動線の観点から、3 章で作成したアルゴリズムを使用し、最適な患者配置を得ることは非常に有効であることが確認できた。

5－2 今後の課題と展望

今後、Dynamo 上に様々なシミュレーションの拡張機能が配布されることで、シミュレーションの作成及びカスタマイズが行いやすい環境が整い、より身近になっていくだろう。設計者が、それらを如何に利用し、如何に共有するかなど分析を深める必要があると考える。

病室のデザイン提案については、病室空間の設計段階において様々な検討・提案が必要になると、BIM を用いたシミュレーションを行うことで設計者・施主の意思疎通を行うことが重要になるだろう。本研究で作成したシミュレーションは、簡易的な病室デザインを構成要素の選択を通して、デザインしていくというものである。これは、病室デザインの提案を自動的に行うという可能性の一端を示したのみである。今後は病室デザインの評価の結果などを通して機械学習をさせていくことで、自動的に病室のデザインを行えるシミュレーションの提案が必要だろう。さらに、規格化できる可能性のある建築を選択・自動化を行うなどの発展について考察する必要がある。

最適化アルゴリズムでは、設定を変えることで、本章で用いた相互関係を含めない場合の患者配置や看護動線が最大になるであろう患者配置を得ることもできる。シミュレーションと合わせて運用することで、設計段階でそれぞれの看護動線量に差が出ないような病棟平面を検討することもできるだろう。設計段階での有効な活用方法を考察し、シミュレーション結果を基に設計を変更・改善するようなプロセスの提案やシミュレーションに用いる、より一般的な看護動線モデルを作成する必要がある。

参考文献

＜第 1 章：参考文献＞

[1] 「BIM その進化と活用」編集委員会. BIM その進化と活用:建築を目指す人、BIM に取り組む人のガイドブック.日韓建設通信新聞社, 2016

[2] 日本建築学会. BIM のかたち:Society5.0 へつながる建築知. 彰国社, 2019

[3] 総務省. 平成 30 年版 情報通信白書のポイント. https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h30/html/nb000000.html (参照:2020-01-30)

[4] 日本建築学会. アルゴリズミック・デザイン. 鹿島出版会, 2009

[5] ノイズ・アーキテクツ. Rhinoceros+Grasshopper 建築デザイン実践ハンドブック 第 2 版. 彰国社, 2014

[6] Autodesk. ユーザー事例:「五感で学ぶ日本海」を具現化する上越市新水族博物館. https://www.autodesk.co.jp/customer-stories/nihon-sekkei (参照:2020-01-30)

[7] 前田建設 建築設計統括部門, BIM PROJECT. https://www.maeda-arch-design.jp/bim.html (参照:2020-01-30)

[8] Autodesk. オートデスクユーザー事例 http://bim-design.com/catalog/img/Nihon-sekkei_Case%20study_ja_web.pdf (参照:2020-01-30)

[9] 藤澤範好, 宮崎隆昌, 中澤公伯. BIM と GIS の連携による日照シミュレーション手法の検討と都市景観デザインへの応用に関する研究. 日本建築学会技術報告集, 第 21 巻 第 47 号, 344-360, 2015

[10]竹澤拓晃, 大西康伸. 応急仮設団地を対象としたビジュアルシミュレーションの実験的活用に関する研究. 日本建築学会大会学術講演梗概集, 2017

[11]正宗尚馬, 大西康伸. 原好佑. BIM を利用した応急仮設住宅の配置計画案の自動作成に関する研究 その 1:配置ルールの設定と配置計画案の自動作成手順の提案. 日本建築学会大会学術講演梗概集, 2019

[12]原好佑, 大西康伸. 正宗尚馬. BIM を利用した応急仮設住宅の配置計画案の自動作成に関する研究 その 1:自動配置プログラムの開発と評価. 日本建築学会大会学術講演梗概集, 2019

[13]福井大学医学部附属病院 看護部. 福井大学医学部附属病院 看護部 PNS の紹介. https://www.hosp.u-fukui.ac.jp/kango/pns/ (参照:2020-01-30)

[14]FM 推進連絡協議会. 公式ガイド ファシリティマネジメント. 日本経済新聞出版社. 2018

＜第 2 章：参考文献＞

[1] Sara O. Marberry and Laurie Zagon. The Power of Color: Creating Healthy Interior Spaces. NY : John Wiley & Sons, Inc. 1995

[2] Sara O. Marberry. Color in the Office: Design Trends from 1950-90. NY : John Wiley & Sons, Inc. 1997

[3] Hessam Ghamari and Cherif M. Amor. The Impact of Color on Healthcare Environments : A Growing Body of Evidence-Based Design. Edra44. 2013

[4] Jain Malkin. Medical and Dental Space Planning 3rd ed. NY : John Wiley & Sons, Inc. 2002

[5] Jain Malkin. Hospital Interior Architecture: Creating Healing Environments for Special Patient Populations. NY. Van Nostrand Reinhold. 2002

[6] Eleanor Lynn Nesmith. Health Care Architecture: Designs for the Future. MA. Rockport Publishers. 1995

[7] 山口恭平, 加藤彰一. BIM を用いた病室デザインの提案に関する考察. 日本建築学会大会学術講演梗概集, 2019

[8] 山口恭平, 加藤彰一. SD 法を用いた病室デザインの評価に関する考察. 日本建築学会東海支部研究報告集, 2020 発表予定

＜第 3 章：参考文献＞

[1] 渡辺玲奈, 良村貞子. 急性期病棟における患者の病床配置と看護必要度との関連:個室・4 床室と中央看護拠点までの距離に関する検討. 看護総合科学研究会誌, 12(1), 15-24, 2009

[2] 谷口元, 今井正次, 加藤彰一, 山本和典, 志田弘二, 柳沢忠. 看護動線量の予測に関する基礎的研究:N 病棟外来病棟への適応. 日本建築学会論文集, 344(0), 116-125, 1984

[3] 竹内貴洋, 山口恭平, 高木碧, 加藤彰一. BIM を用いた看護動線量シミュレーションによる病棟計画及び運営の評価に関する研究. 日本建築学会計画系論文集, 85(767), 33-40, 2020

[4] 竹内貴洋, 加藤彰一. BIM を用いたシミュレーション及び建築計画の教育に関する研究. 三重大学大学院工学研究科建築学専攻修士論文, 2018

[5] 橋本理一郎. 2 次割当て問題に対する近似解放の研究. http://numaf.net/Z9/Z9a/html/THESIS/H14/presen_hashimoto.pdf (参照:2020-01-30)

[6] 立命館大学 光情報通信(左貝)研究室. 遺伝的アルゴリズム. http://www.ritsumei.ac.jp/se/re/sakailab/ga.html (参照:2020-01-30)

[7] 青木義次. プラン作成と遺伝進化とのアナロジー―室配置問題の遺伝進化アルゴリズムによる解法―.日本建築学会計画系論文集，61(481), 151-156, 1996
[8] 村岡直人, 青木義次. 遺伝的アルゴリズムによる平面形状の最適化と設計ノウハウの獲得.日本建築学会計画系論文集，62(497), 111-115, 1997
[9] 岩田伸一郎, 宗本順三, 吉田哲, 阪野明文. 移動コストを評価関数とした廊下パターンと室配置への GA 適用:「An approach to the optimum layout of single-storey buildings」における病院手術等を事例として. 日本建築学会建築系論文集, 64(518), 329-333, 1999
[10]数値解析Ⅱ - 北海学園大学工学部電子情報工学科. 第 16 回　遺伝的アルゴリズム（暫定版）.
http://www.eli.hokkai-s-u.ac.jp/~kikuchi/ma2/chap15.html (参照:2020-01-30)
[11]山口恭平, 加藤彰一. 遺伝的アルゴリズムを用いた患者配置の最適化手法に関する考察: 看護動線シミュレーションを用いた病棟運営の評価手法に関する研究 その１, 日本建築学会東海支部研究報告集, 2020 発表予定

＜第 4 章：参考文献＞

[1] 竹内貴洋, 山口恭平, 高木碧, 加藤彰一. BIM を用いた看護動線量シミュレーションによる病棟計画及び運営の評価に関する研究. 日本建築学会計画系論文集, 85(767), 33-40, 2020
[2] 竹内貴洋, 加藤彰一. BIM を用いたシミュレーション及び建築計画の教育に関する研究. 三重大学大学院工学研究科建築学専攻修士論文, 2018
[3] 山口恭平, 高木碧, 加藤彰一. BIM シミュレーションを用いた評価の実践による考察: 看護動線シミュレーションを用いた病棟運営の評価手法に関する研究 その 2, 日本建築学会東海支部研究報告集, 2020 発表予定

謝辞

修士論文の執筆にあたり、ご指導いただいた加藤先生には深く感謝いたします。学部から 3 年間にわたって大変お世話になりました。設計製図Ⅳ以来、病院建築および医療に関する基礎から専門知識まで、さらには BIM などの建築情報の在り方など様々な知識を得ることができました。また、国際学会をはじめとしたさまざまな場所での発表の機会を頂き、非常に貴重な体験となりました。加藤先生にご指導いただきましたことで、とても有意義な研究生活を送ることができたと考えております。

安井建築設計事務所の篠原様には、病院建築や BIM のシミュレーションに関して、実務の面から細かなご指導をいただきました。今後とも、ご指導ご鞭撻のほど、よろしくお願いいたします。

本研究にあたって、調査にご協力いただいた病院間権者の方々にはご多忙の中、大変お世話になりました。皆様のご協力がなければ本論文は完成しませんでした。今後とも精進してまいります。ご丁寧な対応をいただき、誠にありがとうございました。

加藤研究室の皆様には多大なご支援をいただき、感謝しております。博士課程の加藤先輩、能登先輩にはご自身のお仕事が忙しいにもかかわらず、アドバイスをいただき大変お世話になりました。既に修了された竹内先輩は病院建築や BIM シミュレーション、修士 1 年の高木さんは病院建築を共に研究しており、互いに情報共有しながら活動し、様々な知見を得ることができました。修士 1 年の今井さん、小林君、坂本君には研究に直接の関係はないものの、お互いに情報共有をしながら活動し、私を支えていただきました。学部 4 年の魏さん、桑原さん、西さん、瀧澤君、鷲尾君にはゼミ運営など、様々な面でお世話になりました。ありがとうございました。

最後に、同期の藤田君には調査や論文作成にあたり、様々な場面で助けてもらいました。また、3 年間を通し多くのことを学ばせてもらいました。ありがとうございました。

令和元年 2 月 5 日　　山口 恭平

巻末資料

1. 碧南市民病院―患者配置最適化アルゴリズムのテキストプログラミング

```
import xlrd
import openpyxl
import datetime
import numpy as np
import random as rnd
import matplotlib.pyplot as plt
import copy
from operator import mul



def generator(pop_num):
#初期生成
    xl_bk = xlrd.open_workbook("info_4 階西_碧南.xlsx")
    W = xl_bk.sheet_by_index(5)
    W_ary=[[None for p in range(W.ncols)] for q in range(W.nrows)]
    for y in range(W.nrows):
        for x in range(W.ncols):
            W_ary[y][x]=W.cell(y,x).value

    num = len(W_ary)
    select_num = [i for i in range(num)]

    P = xl_bk.sheet_by_index(0)
    P_ary=[[None for p in range(P.ncols)] for q in range(P.nrows)]

    for y in range(P.nrows):
        for x in range(P.ncols):
            P_ary[y][x]=P.cell(y,x).value

    for i in range(len(P_ary)):
        del P_ary[i][0]
```

```
p_name = P_ary[1] #患者名
for i in range(num):
    if p_name[i] == "":
        p_name[i] = "空"
p_gen = P_ary[2]#患者性別
p_room = P_ary[3]#患者室
p_need = P_ary[4]#患者必要度
p_need_co = copy.deepcopy(p_need)
for i in range(num):
    if p_need[i] == "":
        p_need[i] = 0
    if p_need[i] == 0:
        p_need[i] = 1
p_team = P_ary[5]#患者担当チーム
p_team_co = copy.deepcopy(p_team)
for i in range(num):
    if p_team_co[i] == 1:
        p_team_co[i] = "①"
    if p_team_co[i] == 2:
        p_team_co[i] = "②"
p_pns = P_ary[6]#患者担当 PNS
p_pns_co = copy.deepcopy(p_pns)
for i in range(num):
    if p_pns[i] == "":
        p_pns[i] = "空"

F_ary = [["" for i in range(num)] for j in range(num)]#相関行列作成
for x in range(num):
    for i in range(num):
        if p_team[x] == p_team[i]:
            if p_pns[x] == p_pns[i]:#同 PNS チーム
                F_ary[x][i] = 10
            if p_pns[x] != p_pns[i]:#同チーム別 PNS
                F_ary[x][i] = 5
        if p_team[x] != p_team[i]:#別チーム
```

```
                F_ary[x][i] = 1
            if p_team[i] == "":#空ベッド
                F_ary[x][i] = 0
            if p_team[x] == "":
                F_ary[x][i] = 0
for i in range(len(F_ary)):
    F_ary[i][i] = 0

R = xl_bk.sheet_by_index(1)#特別室
r_unno=[[None for p in range(R.ncols)] for q in range(R.nrows)]
for y in range(R.nrows):
    for x in range(R.ncols):
        r_unno[y][x]=R.cell(y,x).value

for i in range(len(r_unno)):
    del_list = []
    for j in range(len(r_unno[i])):
        if r_unno[i][j] == "":
            del_list.append(j)
    for j in range(len(del_list)):
        del r_unno[i][-1]

p_unno = []#患者特別室
for i in range(len(r_unno)):
    p_unno1 = []
    for j in range(num):
        if r_unno[i][0] == p_room[j]:
            p_unno1.append(j)
    p_unno.append(p_unno1)

for i in range(len(r_unno)):
    del r_unno[i][0]
for i in range(len(r_unno)):
    for j in range(len(r_unno[i])):
        r_unno[i][j] = int(r_unno[i][j])
```

```
p_normal = copy.deepcopy(select_num)#患者普通
r_normal = copy.deepcopy(select_num)#室普通
p_unno_f = sum(p_unno,[])
r_unno_f = sum(r_unno,[])
p_unoc = [i for i, x in enumerate(p_pns) if x == '空']
for i in range(len(p_unno_f)):
    p_normal.remove(p_unno_f[i])
for i in range(len(p_unoc)):
    p_normal.remove(p_unoc[i])
for i in range(len(r_unno_f)):
    r_normal.remove(r_unno_f[i])
r_normal = [r_normal[i:i + 4] for i in range(0,len(r_normal), 4)]

p_gend = []#患者性別
p_gen_m = []
p_gen_w = []
p_gen_u = []
for i in range(num):
    if p_gen[i] == "男":
        p_gen_m.append(i)
    elif p_gen[i] == "女":
        p_gen_w.append(i)
    elif p_gen[i] == "":
        p_gen_u.append(i)
p_gend.append(p_gen_m)
p_gend.append(p_gen_w)
p_gend.append(p_gen_u)

S = xl_bk.sheet_by_index(4)#SS 距離
R_type=[[None for p in range(S.ncols)] for q in range(S.nrows)]
for y in range(S.nrows):
    for x in range(S.ncols):
        R_type[y][x]=S.cell(y,x).value
del R_type[0:2]
```

```
del R_type[0][0]
del R_type[1][0]
SS_dis = R_type[0]
r_ty = R_type[1]

select_pa = [i for i in range(num)]

p_unno1 = sum(p_unno, [])#性別から消去
for i in range(len(p_unno1)):
    try:
        p_gen_m.remove(p_unno1[i])
    except ValueError:
        pass
    try:
        p_gen_w.remove(p_unno1[i])
    except ValueError:
        pass

for i in range(len(p_unno)):#特別病室配置
    for y in range(len(p_unno[i])):
        select_pa.remove(p_unno[i][y])
    if len(r_unno[i]) != len(p_unno[i]):
        for j in range(len(r_unno[i])-len(p_unno[i])):
            u = rnd.choice(p_gen_u)
            p_unno[i].append(u)
            select_pa.remove(u)
            p_gen_u.remove(u)
del r_ty[0:4]

p_gen_m1 = copy.deepcopy(p_gen_m)
p_gen_w1 = copy.deepcopy(p_gen_w)
p_gen_u1 = copy.deepcopy(p_gen_u)
all_ang_end = []
while len(all_ang_end) <= pop_num-1:#初期値生成
    p_unnorma11 = []
```

```
all_ang_end1 = []
for i in range(len(p_unno)):
    rnd_p = rnd.sample(p_unno[i],len(p_unno[i]))
    p_unnorma11.append(rnd_p)
p_unnorma11 = sum(p_unnorma11, [])
all_ang_end1.append(p_unnorma11)

p_gen_mu = []
p_gen_wu = []
select_pa1 = copy.deepcopy(select_pa)
p_gen_mu.extend([p_gen_m1,p_gen_u1])
p_gen_mu = sum(p_gen_mu, [])
p_gen_wu.extend([p_gen_w1,p_gen_u1])
p_gen_wu = sum(p_gen_wu, [])

for x in range(len(r_normal)-1):
    count = []
    for i in range(4):
        sel = rnd.choice(select_pa1)
        count.append(sel)
        if sel not in p_gen_u:
            break
    p_gen_mu = list(set(p_gen_mu) - set(count))
    p_gen_wu = list(set(p_gen_wu) - set(count))
    select_pa1 = list(set(select_pa1) - set(count))
    count1 = copy.deepcopy(count)

    if p_gen[count[-1]] == "男":
        key = 4-len(count)
        try:
            sel1 = rnd.sample(p_gen_mu, key)
            for z in range(key):
                count1.append(sel1[z])
        except ValueError:
            break
```

```
                else:
                    key = 4-len(count)
                    try:
                        sel2 = rnd.sample(p_gen_wu, key)
                        for z in range(key):
                            count1.append(sel2[z])
                    except ValueError:
                        break
                p_gen_mu = list(set(p_gen_mu) - set(count1))
                p_gen_wu = list(set(p_gen_wu) - set(count1))
                select_pa1 = list(set(select_pa1) - set(count1))
                all_ang_end1.append(count1)
            all_ang_end1 = sum(all_ang_end1, [])
            if p_gen_mu == []:
                for i in range(len(p_gen_wu)):
                    all_ang_end1.append(p_gen_wu[i])
            elif p_gen_wu == []:
                for i in range(len(p_gen_mu)):
                    all_ang_end1.append(p_gen_mu[i])
            elif len(p_gen_mu) == 4 and list(set(p_gen_wu) & set(p_gen_u)) ==
p_gen_wu:
                for i in range(len(p_gen_mu)):
                    all_ang_end1.append(p_gen_mu[i])
            elif len(p_gen_wu) == 4 and list(set(p_gen_mu) & set(p_gen_u)) ==
p_gen_mu:
                for i in range(len(p_gen_wu)):
                    all_ang_end1.append(p_gen_wu[i])


            if len(all_ang_end1) == num:
                all_ang_end.append(all_ang_end1)


        return W_ary, F_ary, all_ang_end, p_name, p_gen, p_need_co, p_team_co,
p_pns_co, p_normal, p_unno, r_normal, r_unno, p_gen_m, p_gen_w, p_need, SS__dis,
num, p_gen_m1, p_gen_w1, p_gen_u1
```

```
def evaluate(W_ary, F_ary, all_ang_end, p_name, p_gen, p_need_co, p_team_co,
p_pns_co, p_normal, p_unno, r_normal, r_unno, p_gen_m, p_gen_w, p_need, SS__dis,
num, p_gen_m1, p_gen_w1, p_gen_u1):
    #要素ごとの積で評価
    new1_F_ary = [] #相関行列入れ替え
    for x in range(pop_num):
        mid1_F_ary = []
        for y in range(len(F_ary)):
            mid2_F_ary = []
            for z in range(len(F_ary)):
                mid2_F_ary.append(F_ary[y][all_ang_end[x][z]])
            mid1_F_ary.append(mid2_F_ary)
        new1_F_ary.append(mid1_F_ary)

    new_F_ary = []
    for x in range(pop_num):
        mid3_F_ary = []
        for y in range(pop_num):
            mid4_F_ary = []
            for z in range(len(F_ary)):
                mid4_F_ary.append(new1_F_ary[y][all_ang_end[x][z]])
            mid3_F_ary.append(mid4_F_ary)
        new_F_ary.append(mid3_F_ary[x])

    new_p_need = []#必要度入れ替え
    for x in range(pop_num):
        mid1_p_need = []
        for y in range(len(p_need)):
            mid1_p_need.append(p_need[all_ang_end[x][y]])
        new_p_need.append(mid1_p_need)

    W = sum(W_ary, [])#計算
    W_mat = np.array(W).reshape(len(W_ary), len(W_ary))

    F_mat = []
```

```
    for i in range(pop_num):
        F = sum(new_F_ary[i], [])
        F = np.array(F).reshape(len(F_ary), len(F_ary))
        F_mat.append(F)

    old_evaluate_value = []
    for i in range(pop_num):
        WF = W_mat * F_mat[i]
        old_evaluate_value.append(sum(WF))

    num = len(F_ary)
    select_num = [i for i in range(num)]

    new_evaluate_value = []
    for i in range(pop_num):
        new_evaluate_value1 = []
        for j in range(num):

new_evaluate_value1.append(old_evaluate_value[i][j]*new_p_need[i][j]*SS_dis[j])
        new_evaluate_value.append(new_evaluate_value1)

    evaluate_value = []
    for i in range(pop_num):
        evaluate_value.append(sum(new_evaluate_value[i]))

    index = evaluate_value.index(min(evaluate_value))
    index = all_ang_end[index]

    arrange_pa = []
    for i in range(len(index)):
        index1 = index[i]
        arrange_pa.append(p_name[index1])

    arrange_gen = []
    for i in range(len(index)):
```

```
            index1 = index[i]
            arrange_gen.append(p_gen[index1])

        arrange_need = []
        for i in range(len(index)):
            index1 = index[i]
            arrange_need.append(p_need_co[index1])

        arrange_team = []
        for i in range(len(index)):
            index1 = index[i]
            arrange_team.append(p_team_co[index1])

        arrange_pns = []
        for i in range(len(index)):
            index1 = index[i]
            arrange_pns.append(p_pns_co[index1])
        return evaluate_value, index, arrange_pa, arrange_gen, arrange_need,
arrange_team, arrange_pns

def selection(all_ang_end, evaluate_value, elite_select_num):
        #ルーレット選択とエリート選択
        all_ang_end_cut = []
        evaluate_value_cut = sorted(evaluate_value)
        for i in range(pop_num):
            value = evaluate_value_cut[i]
            index = evaluate_value.index(value)
            all_ang_end_cut.append(all_ang_end[index])
        n = int(pop_num /3*2)
        evaluate_value_cut = evaluate_value_cut[:n]
        all_ang_end_cut = all_ang_end_cut[:n]

        nega_evaluate_value = []
        for i in range(n):
            nega_evaluate_value.append(-evaluate_value_cut[i])
```

```
    nmax = 10    #適応度マッピング値
    nmin = 1
    fitting = ((nmax-nmin)*(min(nega_evaluate_value)-
nega_evaluate_value))/((min(nega_evaluate_value)-max(nega_evaluate_value))+nmin)
    fitting += 1
    prob = fitting/sum(fitting)

    select_pop = []
    elite_pop = []

    while True:
        select = rnd.choices(all_ang_end_cut,weights = prob)
        select_pop.append(select)
        if len(select_pop) >= (pop_num - elite_select_num):
            break
    select_pop = sum(select_pop, [] )

    sort_evaluate_value = copy.deepcopy(evaluate_value)
    sort_evaluate_value.sort(reverse=False)
    for i in range(elite_select_num):
        value = sort_evaluate_value[i]
        index = evaluate_value.index(value)
        elite_pop.append(all_ang_end[index])
    return select_pop, elite_pop

def crossover(pop_num, select_pop, crossover_prob, elite_select_num, num, p_gen_m1,
p_gen_w1, p_gen_u1):
    #順序交叉

    next_pop = []
    while len(next_pop) <= (pop_num - elite_select_num - 2):
        select_pop_cho = copy.deepcopy(select_pop)
        cross_pop = []
        for i in range(2):
            pop = rnd.choice(select_pop_cho)
```

```
        cross_pop.append(pop)
    pop_1 = cross_pop[0]
    pop_2 = cross_pop[1]
    check_prob = rnd.randint(0, 100)
    if check_prob <= crossover_prob:
        st_pop1 = []
        st_pop2 = []
        st_pop1.append(pop_1[0:7])
        st_pop2.append(pop_2[0:7])
        del pop_1[0:7]
        del pop_2[0:7]
        rpop_1 = [pop_1[idx:idx + 4] for idx in range(0,len(pop_1), 4)]
        rpop_2 = [pop_2[idx:idx + 4] for idx in range(0,len(pop_2), 4)]

        cut_index1 = rnd.randint(0, len(rpop_1)-1)
        cut_index2 = rnd.randint(cut_index1+1, len(rpop_1))
        if cut_index1 == 0 and cut_index2 == 8:
            cut_index1 = 1
            cut_index2 = 7

        re1 = (sum(rpop_1[cut_index1:cut_index2],[]))
        re2 = (sum(rpop_2[cut_index1:cut_index2],[]))
        re_1 = []
        re_2 = []
        re1_m = []
        re1_w = []
        re2_m = []
        re2_w = []

        for i in range(len(pop_2)):
            if pop_2[i] not in re1:
                re_1.append(pop_2[i])

        for i in range(len(pop_1)):
            if pop_1[i] not in re2:
```

```
            re_2.append(pop_1[i])

re1_ar = []
for i in range(len(rpop_2)):
    sel = []
    for j in range(len(re_1)):
        if re_1[j] in rpop_2[i]:
            sel.append(re_1[j])
    re1_ar.append(sel)

re2_ar = []
for i in range(len(rpop_1)):
    sel = []
    for j in range(len(re_2)):
        if re_2[j] in rpop_1[i]:
            sel.append(re_2[j])
    re2_ar.append(sel)



for i in range(len(re1_ar)):
    for j in range(len(re1_ar[i])):
        if re1_ar[i][j] in p_gen_m1:
            re1_m.append(re1_ar[i])
            break
        elif re1_ar[i][j] in p_gen_w1:
            re1_w.append(re1_ar[i])
            break
    else:
        sel = rnd.choice([re1_m, re1_w])
        sel.append(re1_ar[i])
re1_m = sum(re1_m,[])
re1_w = sum(re1_w,[])

for i in range(len(re2_ar)):
    for j in range(len(re2_ar[i])):
```

```
                if re2_ar[i][j] in p_gen_m1:
                    re2_m.append(re2_ar[i])
                    break
                elif re2_ar[i][j] in p_gen_w1:
                    re2_w.append(re2_ar[i])
                    break
        else:
            sel = rnd.choice([re2_m, re2_w])
            sel.append(re2_ar[i])
re2_m = sum(re2_m,[])
re2_w = sum(re2_w,[])



if len(re1_m) % 4 != 0:
    if len(re1_m) % 4 > len(re1_w) % 4:
        sel = re1_w
    elif len(re1_m) % 4 < len(re1_w) % 4:
        sel = re1_m
    else:
        sel = rnd.choice([re1_m, re1_w])
    a = list(set(p_gen_u1)-(set(p_gen_u1) - set(sel)))
    b = len(sel) % 4
    try:
        c = rnd.sample(a,b)
    except ValueError:
        if sel == re1_w:
            sel = re1_m
        else:
            sel = re1_w
        a = list(set(p_gen_u1)-(set(p_gen_u1) - set(sel)))
        b = len(sel) % 4
        c = rnd.sample(a,b)
    for i in range(len(c)):
        if sel == re1_m:
            sel1 = rnd.randint(0,len(re1_w))
```

```
                    re1_w.insert(sel1,c[i])
                    sel.remove(c[i])
                elif sel == re1_w:
                    sel1 = rnd.randint(0,len(re1_m))
                    re1_m.insert(sel1,c[i])
                    sel.remove(c[i])

    if len(re2_m) % 4 != 0:
        if len(re2_m) % 4 > len(re2_w) % 4:
            sel = re2_w
        elif len(re2_m) % 4 < len(re2_w) % 4:
            sel = re2_m
        else:
            sel = rnd.choice([re2_m, re2_w])
        a = list(set(p_gen_u1)-(set(p_gen_u1) - set(sel)))
        b = len(sel) % 4
        try:
            c = rnd.sample(a,b)
        except ValueError:
            if sel == re2_w:
                sel = re2_m
            else:
                sel = re2_w
            a = list(set(p_gen_u1)-(set(p_gen_u1) - set(sel)))
            b = len(sel) % 4
            c = rnd.sample(a,b)
        for i in range(len(c)):
            if sel == re2_m:
                sel1 = rnd.randint(0,len(re2_w))
                re2_w.insert(sel1,c[i])
                sel.remove(c[i])
            elif sel == re2_w:
                sel1 = rnd.randint(0,len(re2_m))
                re2_m.insert(sel1,c[i])
                sel.remove(c[i])
```

```
            re1_m = [re1_m[idx:idx + 4] for idx in range(0,len(re1_m), 4)]
            re1_w = [re1_w[idx:idx + 4] for idx in range(0,len(re1_w), 4)]
            re2_m = [re2_m[idx:idx + 4] for idx in range(0,len(re2_m), 4)]
            re2_w = [re2_w[idx:idx + 4] for idx in range(0,len(re2_w), 4)]
            re_re1 = [re1[idx:idx + 4] for idx in range(0,len(re1), 4)]
            re_re2 = [re2[idx:idx + 4] for idx in range(0,len(re2), 4)]



            re1_mw = re1_m + re1_w
            re2_mw = re2_m + re2_w
            rnd.shuffle(re1_mw)
            rnd.shuffle(re2_mw)

            samp = [i for i in range(len(rpop_1))]
            samp1 = samp[cut_index1:cut_index2]
            samp = list(set(samp)-set(samp1))

            for i in range(len(samp)):
                re_re1.insert(samp[i],re1_mw[i])
                re_re2.insert(samp[i],re2_mw[i])

            ls_pop1 = st_pop1 + re_re1
            ls_pop2 = st_pop2 + re_re2
            ls_pop1 = sum(ls_pop1, [])
            ls_pop2 = sum(ls_pop2, [])

            if len(ls_pop1) == num:
                next_pop.append(ls_pop1)
            if len(ls_pop2) == num:
                next_pop.append(ls_pop2)
        else:
            next_pop.append(pop_1)
            next_pop.append(pop_2)
    return next_pop
```

```
def mutation(mutation_prob, next_pop, p_gen_m1, p_gen_w1):
    #突然変異
    last_pop = copy.deepcopy(next_pop)
    for i in range(len(last_pop)):
        check_prob = rnd.randint(0, 100)
        if check_prob <= mutation_prob:
            sl = rnd.choice([p_gen_m1, p_gen_w1])
            sl_index = rnd.sample(sl, 2)
            a = last_pop[i].index(sl_index[0])
            b = last_pop[i].index(sl_index[1])
            c = last_pop[i][a]
            d = last_pop[i][b]
            last_pop[i][a] = d
            last_pop[i][b] = c
    return last_pop

def export(arrange_pa, arrange_gen, arrange_need, arrange_team, arrange_pns,
min_eva):

    wb = openpyxl.load_workbook("info_4 階西_碧南.xlsx")
    ws_1 = wb["患者配置"]
    for i in range(len(arrange_pa)):
        ws_1.cell(row=2, column=(i+1), value = arrange_pa[i])
    for i in range(len(arrange_gen)):
        ws_1.cell(row=3, column=(i+1), value = arrange_gen[i])
    for i in range(len(arrange_need)):
        ws_1.cell(row=4, column=(i+1), value = arrange_need[i])
    for i in range(len(arrange_team)):
        ws_1.cell(row=5, column=(i+1), value = arrange_team[i])
    for i in range(len(arrange_pns)):
        ws_1.cell(row=6, column=(i+1), value = arrange_pns[i])

    ws_2 = wb["グラフ"]
    for i in range(len(min_eva)):
```

```
            ws_2.cell(row=2, column=(i+1), value = min_eva[i])

        now = datetime.datetime.now().strftime("%y%m%d") + "_4階西" + ".xlsx"
        wb.save(filename = now)

        return

pop_num =32       #個体数
generation_num = 10000        #世代数

elite_select_num = 2       #エリート選択数
crossover_prob = 100        #交叉確率
mutation_prob = 10     #突然変異確率

generation_count = 1
W_ary, F_ary, all_ang_end, p_name, p_gen, p_need_co, p_team_co, p_pns_co,
p_normal, p_unno, r_normal, r_unno, p_gen_m, p_gen_w, p_need, SS__dis, num,
p_gen_m1, p_gen_w1, p_gen_u1 = generator(pop_num)

min_eva = []

while generation_count <= generation_num:
        evaluate_value, index, arrange_pa, arrange_gen, arrange_need, arrange_team,
arrange_pns = evaluate(W_ary, F_ary, all_ang_end, p_name, p_gen, p_need_co,
p_team_co, p_pns_co, p_normal, p_unno, r_normal, r_unno, p_gen_m, p_gen_w,
p_need, SS__dis, num, p_gen_m1, p_gen_w1, p_gen_u1)
        select_pop, elite_pop = selection(all_ang_end, evaluate_value, elite_select_num)

        print(generation_count, "世代")
        print("最良値:", min(evaluate_value))

        print("------------------------------")

        min_eva.append(min(evaluate_value))
```

```
    next_pop = crossover(pop_num, select_pop, crossover_prob, elite_select_num,
num, p_gen_m1, p_gen_w1, p_gen_u1)
    last_pop = mutation(mutation_prob, next_pop, p_gen_m1, p_gen_w1)
    nextgene_pop = last_pop + elite_pop
    all_ang_end = nextgene_pop
    generation_count +=1


export(arrange_pa, arrange_gen, arrange_need, arrange_team, arrange_pns, min_eva)
```

2. 下呂温泉病院―患者配置最適化アルゴリズムのテキストプログラミング

```
import xlrd
import openpyxl
import datetime
import numpy as np
import random as rnd
import matplotlib.pyplot as plt
import copy
from operator import mul



def generator(pop_num):
#初期生成
    xl_bk = xlrd.open_workbook("info_4 階西_碧南.xlsx")
    W = xl_bk.sheet_by_index(5)
    W_ary=[[None for p in range(W.ncols)] for q in range(W.nrows)]
    for y in range(W.nrows):
        for x in range(W.ncols):
            W_ary[y][x]=W.cell(y,x).value

    num = len(W_ary)
    select_num = [i for i in range(num)]

    P = xl_bk.sheet_by_index(0)
    P_ary=[[None for p in range(P.ncols)] for q in range(P.nrows)]

    for y in range(P.nrows):
        for x in range(P.ncols):
            P_ary[y][x]=P.cell(y,x).value

    for i in range(len(P_ary)):
        del P_ary[i][0]
```

```
p_name = P_ary[1] #患者名
for i in range(num):
    if p_name[i] == "":
        p_name[i] = "空"
p_gen = P_ary[2]#患者性別
p_room = P_ary[3]#患者室
p_need = P_ary[4]#患者必要度
p_need_co = copy.deepcopy(p_need)
for i in range(num):
    if p_need[i] == "":
        p_need[i] = 0
    if p_need[i] == 0:
        p_need[i] = 1
p_team = P_ary[5]#患者担当チーム
p_team_co = copy.deepcopy(p_team)
for i in range(num):
    if p_team_co[i] == 1:
        p_team_co[i] = "①"
    if p_team_co[i] == 2:
        p_team_co[i] = "②"
p_pns = P_ary[6]#患者担当 PNS
p_pns_co = copy.deepcopy(p_pns)
for i in range(num):
    if p_pns[i] == "":
        p_pns[i] = "空"

F_ary = [["" for i in range(num)] for j in range(num)]#相関行列作成
for x in range(num):
    for i in range(num):
        if p_team[x] == p_team[i]:
            if p_pns[x] == p_pns[i]:#同 PNS チーム
                F_ary[x][i] = 10
            if p_pns[x] != p_pns[i]:#同チーム別 PNS
                F_ary[x][i] = 5
        if p_team[x] != p_team[i]:#別チーム
```

```
                F_ary[x][i] = 1
            if p_team[i] == "":#空ベッド
                F_ary[x][i] = 0
            if p_team[x] == "":
                F_ary[x][i] = 0
for i in range(len(F_ary)):
    F_ary[i][i] = 0

R = xl_bk.sheet_by_index(1)#特別室
r_unno=[[None for p in range(R.ncols)] for q in range(R.nrows)]
for y in range(R.nrows):
    for x in range(R.ncols):
        r_unno[y][x]=R.cell(y,x).value

for i in range(len(r_unno)):
    del_list = []
    for j in range(len(r_unno[i])):
        if r_unno[i][j] == "":
            del_list.append(j)
    for j in range(len(del_list)):
        del r_unno[i][-1]

p_unno = []#患者特別室
for i in range(len(r_unno)):
    p_unno1 = []
    for j in range(num):
        if r_unno[i][0] == p_room[j]:
            p_unno1.append(j)
    p_unno.append(p_unno1)

for i in range(len(r_unno)):
    del r_unno[i][0]
for i in range(len(r_unno)):
    for j in range(len(r_unno[i])):
        r_unno[i][j] = int(r_unno[i][j])
```

```
p_normal = copy.deepcopy(select_num)#患者普通
r_normal = copy.deepcopy(select_num)#室普通
p_unno_f = sum(p_unno,[])
r_unno_f = sum(r_unno,[])
p_unoc = [i for i, x in enumerate(p_pns) if x == '空']
for i in range(len(p_unno_f)):
    p_normal.remove(p_unno_f[i])
for i in range(len(p_unoc)):
    p_normal.remove(p_unoc[i])
for i in range(len(r_unno_f)):
    r_normal.remove(r_unno_f[i])
r_normal = [r_normal[i:i + 4] for i in range(0,len(r_normal), 4)]

p_gend = []#患者性別
p_gen_m = []
p_gen_w = []
p_gen_u = []
for i in range(num):
    if p_gen[i] == "男":
        p_gen_m.append(i)
    elif p_gen[i] == "女":
        p_gen_w.append(i)
    elif p_gen[i] == "":
        p_gen_u.append(i)
p_gend.append(p_gen_m)
p_gend.append(p_gen_w)
p_gend.append(p_gen_u)

S = xl_bk.sheet_by_index(4)#SS 距離
R_type=[[None for p in range(S.ncols)] for q in range(S.nrows)]
for y in range(S.nrows):
    for x in range(S.ncols):
        R_type[y][x]=S.cell(y,x).value
del R_type[0:2]
```

```
del R_type[0][0]
del R_type[1][0]
SS_dis = R_type[0]
r_ty = R_type[1]

select_pa = [i for i in range(num)]

p_unno1 = sum(p_unno, [])#性別から消去
for i in range(len(p_unno1)):
    try:
        p_gen_m.remove(p_unno1[i])
    except ValueError:
        pass
    try:
        p_gen_w.remove(p_unno1[i])
    except ValueError:
        pass

for i in range(len(p_unno)):#特別病室配置
    for y in range(len(p_unno[i])):
        select_pa.remove(p_unno[i][y])
    if len(r_unno[i]) != len(p_unno[i]):
        for j in range(len(r_unno[i])-len(p_unno[i])):
            u = rnd.choice(p_gen_u)
            p_unno[i].append(u)
            select_pa.remove(u)
            p_gen_u.remove(u)
del r_ty[0:4]

p_gen_m1 = copy.deepcopy(p_gen_m)
p_gen_w1 = copy.deepcopy(p_gen_w)
p_gen_u1 = copy.deepcopy(p_gen_u)
all_ang_end = []
while len(all_ang_end) <= pop_num-1:#初期値生成
    p_unnorma11 = []
```

```
all_ang_end1 = []
for i in range(len(p_unno)):
    rnd_p = rnd.sample(p_unno[i],len(p_unno[i]))
    p_unnorma11.append(rnd_p)
p_unnorma11 = sum(p_unnorma11, [])
all_ang_end1.append(p_unnorma11)

p_gen_mu = []
p_gen_wu = []
select_pa1 = copy.deepcopy(select_pa)
p_gen_mu.extend([p_gen_m1,p_gen_u1])
p_gen_mu = sum(p_gen_mu, [])
p_gen_wu.extend([p_gen_w1,p_gen_u1])
p_gen_wu = sum(p_gen_wu, [])

for x in range(len(r_normal)-1):
    count = []
    for i in range(4):
        sel = rnd.choice(select_pa1)
        count.append(sel)
        if sel not in p_gen_u:
            break
    p_gen_mu = list(set(p_gen_mu) - set(count))
    p_gen_wu = list(set(p_gen_wu) - set(count))
    select_pa1 = list(set(select_pa1) - set(count))
    count1 = copy.deepcopy(count)

    if p_gen[count[-1]] == "男":
        key = 4-len(count)
        try:
            sel1 = rnd.sample(p_gen_mu, key)
            for z in range(key):
                count1.append(sel1[z])
        except ValueError:
            break
```

```
            else:
                key = 4-len(count)
                try:
                    sel2 = rnd.sample(p_gen_wu, key)
                    for z in range(key):
                        count1.append(sel2[z])
                except ValueError:
                    break
            p_gen_mu = list(set(p_gen_mu) - set(count1))
            p_gen_wu = list(set(p_gen_wu) - set(count1))
            select_pa1 = list(set(select_pa1) - set(count1))
            all_ang_end1.append(count1)
        all_ang_end1 = sum(all_ang_end1, [])
        if p_gen_mu == []:
            for i in range(len(p_gen_wu)):
                all_ang_end1.append(p_gen_wu[i])
        elif p_gen_wu == []:
            for i in range(len(p_gen_mu)):
                all_ang_end1.append(p_gen_mu[i])
        elif len(p_gen_mu) == 4 and list(set(p_gen_wu) & set(p_gen_u)) ==
p_gen_wu:
            for i in range(len(p_gen_mu)):
                all_ang_end1.append(p_gen_mu[i])
        elif len(p_gen_wu) == 4 and list(set(p_gen_mu) & set(p_gen_u)) ==
p_gen_mu:
            for i in range(len(p_gen_wu)):
                all_ang_end1.append(p_gen_wu[i])

        if len(all_ang_end1) == num:
            all_ang_end.append(all_ang_end1)

    return W_ary, F_ary, all_ang_end, p_name, p_gen, p_need_co, p_team_co,
p_pns_co, p_normal, p_unno, r_normal, r_unno, p_gen_m, p_gen_w, p_need, SS__dis,
num, p_gen_m1, p_gen_w1, p_gen_u1
```

```
def evaluate(W_ary, F_ary, all_ang_end, p_name, p_gen, p_need_co, p_team_co,
p_pns_co, p_normal, p_unno, r_normal, r_unno, p_gen_m, p_gen_w, p_need, SS__dis,
num, p_gen_m1, p_gen_w1, p_gen_u1):
    #要素ごとの積で評価
    new1_F_ary = [] #相関行列入れ替え
    for x in range(pop_num):
        mid1_F_ary = []
        for y in range(len(F_ary)):
            mid2_F_ary = []
            for z in range(len(F_ary)):
                mid2_F_ary.append(F_ary[y][all_ang_end[x][z]])
            mid1_F_ary.append(mid2_F_ary)
        new1_F_ary.append(mid1_F_ary)

    new_F_ary = []
    for x in range(pop_num):
        mid3_F_ary = []
        for y in range(pop_num):
            mid4_F_ary = []
            for z in range(len(F_ary)):
                mid4_F_ary.append(new1_F_ary[y][all_ang_end[x][z]])
            mid3_F_ary.append(mid4_F_ary)
        new_F_ary.append(mid3_F_ary[x])

    new_p_need = []#必要度入れ替え
    for x in range(pop_num):
        mid1_p_need = []
        for y in range(len(p_need)):
            mid1_p_need.append(p_need[all_ang_end[x][y]])
        new_p_need.append(mid1_p_need)

    W = sum(W_ary, [])#計算
    W_mat = np.array(W).reshape(len(W_ary), len(W_ary))

    F_mat = []
```

```
    for i in range(pop_num):
        F = sum(new_F_ary[i], [])
        F = np.array(F).reshape(len(F_ary), len(F_ary))
        F_mat.append(F)

    old_evaluate_value = []
    for i in range(pop_num):
        WF = W_mat * F_mat[i]
        old_evaluate_value.append(sum(WF))

    num = len(F_ary)
    select_num = [i for i in range(num)]

    new_evaluate_value = []
    for i in range(pop_num):
        new_evaluate_value1 = []
        for j in range(num):

new_evaluate_value1.append(old_evaluate_value[i][j]*new_p_need[i][j]*SS_dis[j])
        new_evaluate_value.append(new_evaluate_value1)

    evaluate_value = []
    for i in range(pop_num):
        evaluate_value.append(sum(new_evaluate_value[i]))

    index = evaluate_value.index(min(evaluate_value))
    index = all_ang_end[index]

    arrange_pa = []
    for i in range(len(index)):
        index1 = index[i]
        arrange_pa.append(p_name[index1])

    arrange_gen = []
    for i in range(len(index)):
```

```
            index1 = index[i]
            arrange_gen.append(p_gen[index1])

      arrange_need = []
      for i in range(len(index)):
            index1 = index[i]
            arrange_need.append(p_need_co[index1])

      arrange_team = []
      for i in range(len(index)):
            index1 = index[i]
            arrange_team.append(p_team_co[index1])

      arrange_pns = []
      for i in range(len(index)):
            index1 = index[i]
            arrange_pns.append(p_pns_co[index1])
      return evaluate_value, index, arrange_pa, arrange_gen, arrange_need,
arrange_team, arrange_pns

def selection(all_ang_end, evaluate_value, elite_select_num):
      #ルーレット選択とエリート選択
      all_ang_end_cut = []
      evaluate_value_cut = sorted(evaluate_value)
      for i in range(pop_num):
            value = evaluate_value_cut[i]
            index = evaluate_value.index(value)
            all_ang_end_cut.append(all_ang_end[index])
      n = int(pop_num /3*2)
      evaluate_value_cut = evaluate_value_cut[:n]
      all_ang_end_cut = all_ang_end_cut[:n]

      nega_evaluate_value = []
      for i in range(n):
            nega_evaluate_value.append(-evaluate_value_cut[i])
```

```
    nmax = 10    #適応度マッピング値
    nmin = 1
    fitting = ((nmax-nmin)*(min(nega_evaluate_value)-
nega_evaluate_value))/((min(nega_evaluate_value)-max(nega_evaluate_value))+nmin)
    fitting += 1
    prob = fitting/sum(fitting)

    select_pop = []
    elite_pop = []

    while True:
        select = rnd.choices(all_ang_end_cut,weights = prob)
        select_pop.append(select)
        if len(select_pop) >= (pop_num - elite_select_num):
            break
    select_pop = sum(select_pop, [] )

    sort_evaluate_value = copy.deepcopy(evaluate_value)
    sort_evaluate_value.sort(reverse=False)
    for i in range(elite_select_num):
        value = sort_evaluate_value[i]
        index = evaluate_value.index(value)
        elite_pop.append(all_ang_end[index])
    return select_pop, elite_pop

def crossover(pop_num, select_pop, crossover_prob, elite_select_num, num, p_gen_m1,
p_gen_w1, p_gen_u1):
    #順序交叉

    next_pop = []
    while len(next_pop) <= (pop_num - elite_select_num - 2):
        select_pop_cho = copy.deepcopy(select_pop)
        cross_pop = []
        for i in range(2):
            pop = rnd.choice(select_pop_cho)
```

```
        cross_pop.append(pop)
    pop_1 = cross_pop[0]
    pop_2 = cross_pop[1]
    check_prob = rnd.randint(0, 100)
    if check_prob <= crossover_prob:
        st_pop1 = []
        st_pop2 = []
        st_pop1.append(pop_1[0:7])
        st_pop2.append(pop_2[0:7])
        del pop_1[0:7]
        del pop_2[0:7]
        rpop_1 = [pop_1[idx:idx + 4] for idx in range(0,len(pop_1), 4)]
        rpop_2 = [pop_2[idx:idx + 4] for idx in range(0,len(pop_2), 4)]

        cut_index1 = rnd.randint(0, len(rpop_1)-1)
        cut_index2 = rnd.randint(cut_index1+1, len(rpop_1))
        if cut_index1 == 0 and cut_index2 == 8:
            cut_index1 = 1
            cut_index2 = 7

        re1 = (sum(rpop_1[cut_index1:cut_index2],[]))
        re2 = (sum(rpop_2[cut_index1:cut_index2],[]))
        re_1 = []
        re_2 = []
        re1_m = []
        re1_w = []
        re2_m = []
        re2_w = []

        for i in range(len(pop_2)):
            if pop_2[i] not in re1:
                re_1.append(pop_2[i])

        for i in range(len(pop_1)):
            if pop_1[i] not in re2:
```

```
            re_2.append(pop_1[i])

re1_ar = []
for i in range(len(rpop_2)):
    sel = []
    for j in range(len(re_1)):
        if re_1[j] in rpop_2[i]:
            sel.append(re_1[j])
    re1_ar.append(sel)

re2_ar = []
for i in range(len(rpop_1)):
    sel = []
    for j in range(len(re_2)):
        if re_2[j] in rpop_1[i]:
            sel.append(re_2[j])
    re2_ar.append(sel)



for i in range(len(re1_ar)):
    for j in range(len(re1_ar[i])):
        if re1_ar[i][j] in p_gen_m1:
            re1_m.append(re1_ar[i])
            break
        elif re1_ar[i][j] in p_gen_w1:
            re1_w.append(re1_ar[i])
            break
    else:
        sel = rnd.choice([re1_m, re1_w])
        sel.append(re1_ar[i])
re1_m = sum(re1_m,[])
re1_w = sum(re1_w,[])

for i in range(len(re2_ar)):
    for j in range(len(re2_ar[i])):
```

```
                if re2_ar[i][j] in p_gen_m1:
                    re2_m.append(re2_ar[i])
                    break
                elif re2_ar[i][j] in p_gen_w1:
                    re2_w.append(re2_ar[i])
                    break
        else:
            sel = rnd.choice([re2_m, re2_w])
            sel.append(re2_ar[i])
    re2_m = sum(re2_m,[])
    re2_w = sum(re2_w,[])



    if len(re1_m) % 4 != 0:
        if len(re1_m) % 4 > len(re1_w) % 4:
            sel = re1_w
        elif len(re1_m) % 4 < len(re1_w) % 4:
            sel = re1_m
        else:
            sel = rnd.choice([re1_m, re1_w])
        a = list(set(p_gen_u1)-(set(p_gen_u1) - set(sel)))
        b = len(sel) % 4
        try:
            c = rnd.sample(a,b)
        except ValueError:
            if sel == re1_w:
                sel = re1_m
            else:
                sel = re1_w
            a = list(set(p_gen_u1)-(set(p_gen_u1) - set(sel)))
            b = len(sel) % 4
            c = rnd.sample(a,b)
        for i in range(len(c)):
            if sel == re1_m:
                sel1 = rnd.randint(0,len(re1_w))
```

```
                re1_w.insert(sel1,c[i])
                sel.remove(c[i])
            elif sel == re1_w:
                sel1 = rnd.randint(0,len(re1_m))
                re1_m.insert(sel1,c[i])
                sel.remove(c[i])

    if len(re2_m) % 4 != 0:
        if len(re2_m) % 4 > len(re2_w) % 4:
            sel = re2_w
        elif len(re2_m) % 4 < len(re2_w) % 4:
            sel = re2_m
        else:
            sel = rnd.choice([re2_m, re2_w])
        a = list(set(p_gen_u1)-(set(p_gen_u1) - set(sel)))
        b = len(sel) % 4
        try:
            c = rnd.sample(a,b)
        except ValueError:
            if sel == re2_w:
                sel = re2_m
            else:
                sel = re2_w
            a = list(set(p_gen_u1)-(set(p_gen_u1) - set(sel)))
            b = len(sel) % 4
            c = rnd.sample(a,b)
        for i in range(len(c)):
            if sel == re2_m:
                sel1 = rnd.randint(0,len(re2_w))
                re2_w.insert(sel1,c[i])
                sel.remove(c[i])
            elif sel == re2_w:
                sel1 = rnd.randint(0,len(re2_m))
                re2_m.insert(sel1,c[i])
                sel.remove(c[i])
```

```
            re1_m = [re1_m[idx:idx + 4] for idx in range(0,len(re1_m), 4)]
            re1_w = [re1_w[idx:idx + 4] for idx in range(0,len(re1_w), 4)]
            re2_m = [re2_m[idx:idx + 4] for idx in range(0,len(re2_m), 4)]
            re2_w = [re2_w[idx:idx + 4] for idx in range(0,len(re2_w), 4)]
            re_re1 = [re1[idx:idx + 4] for idx in range(0,len(re1), 4)]
            re_re2 = [re2[idx:idx + 4] for idx in range(0,len(re2), 4)]



            re1_mw = re1_m + re1_w
            re2_mw = re2_m + re2_w
            rnd.shuffle(re1_mw)
            rnd.shuffle(re2_mw)

            samp = [i for i in range(len(rpop_1))]
            samp1 = samp[cut_index1:cut_index2]
            samp = list(set(samp)-set(samp1))

            for i in range(len(samp)):
                  re_re1.insert(samp[i],re1_mw[i])
                  re_re2.insert(samp[i],re2_mw[i])

            ls_pop1 = st_pop1 + re_re1
            ls_pop2 = st_pop2 + re_re2
            ls_pop1 = sum(ls_pop1, [])
            ls_pop2 = sum(ls_pop2, [])

            if len(ls_pop1) == num:
                  next_pop.append(ls_pop1)
            if len(ls_pop2) == num:
                  next_pop.append(ls_pop2)
      else:
            next_pop.append(pop_1)
            next_pop.append(pop_2)
return next_pop
```

```
def mutation(mutation_prob, next_pop, p_gen_m1, p_gen_w1):
    #突然変異
    last_pop = copy.deepcopy(next_pop)
    for i in range(len(last_pop)):
        check_prob = rnd.randint(0, 100)
        if check_prob <= mutation_prob:
            sl = rnd.choice([p_gen_m1, p_gen_w1])
            sl_index = rnd.sample(sl, 2)
            a = last_pop[i].index(sl_index[0])
            b = last_pop[i].index(sl_index[1])
            c = last_pop[i][a]
            d = last_pop[i][b]
            last_pop[i][a] = d
            last_pop[i][b] = c
    return last_pop

def export(arrange_pa, arrange_gen, arrange_need, arrange_team, arrange_pns,
min_eva):

    wb = openpyxl.load_workbook("info_4 階西_碧南.xlsx")
    ws_1 = wb["患者配置"]
    for i in range(len(arrange_pa)):
        ws_1.cell(row=2, column=(i+1), value = arrange_pa[i])
    for i in range(len(arrange_gen)):
        ws_1.cell(row=3, column=(i+1), value = arrange_gen[i])
    for i in range(len(arrange_need)):
        ws_1.cell(row=4, column=(i+1), value = arrange_need[i])
    for i in range(len(arrange_team)):
        ws_1.cell(row=5, column=(i+1), value = arrange_team[i])
    for i in range(len(arrange_pns)):
        ws_1.cell(row=6, column=(i+1), value = arrange_pns[i])

    ws_2 = wb["グラフ"]
    for i in range(len(min_eva)):
```

```
            ws_2.cell(row=2, column=(i+1), value = min_eva[i])

        now = datetime.datetime.now().strftime("%y%m%d") + "_4階西" + ".xlsx"
        wb.save(filename = now)

        return

pop_num =32       #個体数
generation_num = 10000        #世代数

elite_select_num = 2       #エリート選択数
crossover_prob = 100        #交叉確率
mutation_prob = 10     #突然変異確率

generation_count = 1
W_ary, F_ary, all_ang_end, p_name, p_gen, p_need_co, p_team_co, p_pns_co,
p_normal, p_unno, r_normal, r_unno, p_gen_m, p_gen_w, p_need, SS__dis, num,
p_gen_m1, p_gen_w1, p_gen_u1 = generator(pop_num)

min_eva = []

while generation_count <= generation_num:
        evaluate_value, index, arrange_pa, arrange_gen, arrange_need, arrange_team,
arrange_pns = evaluate(W_ary, F_ary, all_ang_end, p_name, p_gen, p_need_co,
p_team_co, p_pns_co, p_normal, p_unno, r_normal, r_unno, p_gen_m, p_gen_w,
p_need, SS__dis, num, p_gen_m1, p_gen_w1, p_gen_u1)
        select_pop, elite_pop = selection(all_ang_end, evaluate_value, elite_select_num)

        print(generation_count, "世代")
        print("最良値:", min(evaluate_value))

        print("------------------------------")

        min_eva.append(min(evaluate_value))
```

```
    next_pop = crossover(pop_num, select_pop, crossover_prob, elite_select_num,
num, p_gen_m1, p_gen_w1, p_gen_u1)
    last_pop = mutation(mutation_prob, next_pop, p_gen_m1, p_gen_w1)
    nextgene_pop = last_pop + elite_pop
    all_ang_end = nextgene_pop
    generation_count +=1


export(arrange_pa, arrange_gen, arrange_need, arrange_team, arrange_pns, min_eva)
```

表 3-1　患者情報

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
患者名	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AN	AM	AO	AP	AQ	AR	AS	AT	AU	AV	AW	AX
性別	男	男	男	男	男	男	男	男	女	女	女	女	男	男	男	男	男	男	男	女	女	女	女	男
特別室																								個
看護必要度B	2	2	0	0	0	0	0	0	5	0	5	2	5	3	6	6	10	10	10	7	9	7	10	0
受け持ちチーム	2	1	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ペア	B	A	B	B	B	B	B	B	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A

	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
患者名	AY	AZ	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BN	BM	BO						
性別	男	男	女	男	男	男	男	女	女	女	女	女	男	男	男	男	女						
特別室	個	個	個	個	感												特						
看護必要度B	5	2	8	6	12	0	0	0	6	4	4	7	5	3	4	3	6						
受け持ちチーム	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	1						
ペア	A	A	A	A	A	B	B	B	B	B	B	B	B	B	B	B	A						

表 3-2　病室情報

個	0	1	2	3	4
感	5				
特	6				

表 3-3　SS と病室間の距離情報

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
SS-病室　距離	14	10	9	9	10	14	22	48	48	48	48	42	42	42	42	36	36	36	36	30	30	30	30	24

	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
SS-病室　距離	24	24	24	18	18	18	18	48	48	48	48	42	42	42	42	34	34	34	34	28	28	28	28

表 4-1　患者情報

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
患者名	A	B	C	D	E	F	G	H	I	J	K	L	N	M	O	P	Q	R	S	T	U	V	W	
性別	男	女	男	女	女	女	男	男	男	男	男	男	男	女	女	女	女	女	女	女	女	女	女	
特別室	個	個	個																					
看護必要度B	0	5	8	9	5	3	5	4	0	0	7	10	10	12	8	10	0	11	2	0	0	4	0	
受け持ちチーム	1	2	2	1	1	1	2	2	1	2	2	2	2	2	2	2	2	2	2	1	2	1	1	
ペア	A	C	B	A	A	A	C	C	A	C	B	B	B	B	C	C	B	B	C	A	C	A	A	

	24	25	26	27	28	29	30	31	32	33	34	35
患者名												
性別												
特別室												
看護必要度B												
受け持ちチーム												
ペア												

表 4-2　病室情報

個	0	1	2	3

表 4-3　SS と病室間の距離情報

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
SS-病室　距離	10	6	5	10	13	13	13	13	20	20	20	20	25	25	25	25	25	25	25	25	20	20	20	20

	24	25	26	27	28	29	30	31	32	33	34	35
SS-病室　距離	13	13	13	13	8	8	8	8	9	9	9	9

表 5-1　患者情報

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
患者名	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	B1	B2	B3	B4	B5	B6	B7	B8
特別室																			重					
看護必要度	9	11	1	7	1	6	1	16	0	2	1	3	3	2	1	7	14	8	15	3	10	9	2	1
受け持ちチーム	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2

	24	25	26	27	28	29	30	31	32	33	34	35	36	37
患者名	B9	B10	B11	B12	B13	B14								
特別室														
看護必要度	2	0	4	1	12	6								
受け持ちチーム	2	2	2	2	2	2								

表 5-2　病室情報

特	0	1
重	2	3
感	4	

表 5-3　SS と病室間の距離情報

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
SS-病室　距離	17	18	24	24	19	17	17	17	26	26	26	26	26	24	24	21	21	21	21	21	30	30	30	30

	24	25	26	27	28	29	30	31	32	33	34	35	36	37
SS-病室　距離	30	24	24	24	24	24	24	24	24	24	24	18	18	18

表 6-1　患者情報

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
患者名	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
特別室				重	重																			
看護必要度	13	7	5	15	11	12	3	6	9	0	7	1	0	11	8	0	1	4	6	2	2	9	7	7
受け持ちチーム	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2

	24	25	26	27	28	29	30	31	32	33	34	35	36	37
患者名	C12	C13												
特別室														
看護必要度	2	13												
受け持ちチーム	2	2												

表 6-2　病室情報

特	0	1	
重	2	3	4
感	5		

表 6-3　SS と病室間の距離情報

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
SS-病室　距離	17	18	24	24	19	17	17	17	26	26	26	26	26	24	24	21	21	21	21	21	30	30	30	30

	24	25	26	27	28	29	30	31	32	33	34	35	36	37
SS-病室　距離	30	24	24	24	24	24	24	24	24	24	24	18	18	18