UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Civil, Arquitetura e Urbanismo

VERLEY HENRY CÔCO JÚNIOR

O PROCESSO COLABORATIVO ENTRE ARQUITETOS E ALGORITMOS NA CONCEPÇÃO ARQUITETÔNICA:

Um estudo sobre a geração automatizada de layouts de mobiliário como ferramenta projetual

CAMPINAS

2021

VERLEY HENRY CÔCO JÚNIOR

O PROCESSO COLABORATIVO ENTRE ARQUITETOS E ALGORITMOS NA CONCEPÇÃO ARQUITETÔNICA:

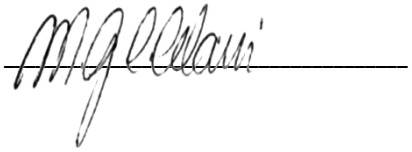
Um estudo sobre a geração automatizada de layouts de mobiliário como ferramenta projetual

Dissertação de Mestrado apresentada a Faculdade de Engenharia Civil, Arquitetura e Urbanismo da Unicamp, para obtenção do título de Mestre em Arquitetura, Tecnologia e Cidade, na área de Arquitetura, Tecnologia e Cidade.

Orientadora: Profa. Dra. Maria Gabriela Caffarena Celani

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO VERLEY HENRY CÔCO JÚNIOR E ORIENTADO PELA PROFA. DRA. MARIA GABRIELA CAFFARENA CELANI.

ASSINATURA DA ORIENTADORA



CAMPINAS

2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

C647p

Côco Júnior, Verley Henry, 1993-
O processo colaborativo entre arquitetos e algoritmos na concepção arquitetônica : um estudo sobre a geração automatizada de layouts de mobiliário como ferramenta projetual / Verley Henry Côco Júnior. – Campinas, SP : [s.n.], 2021.

Orientador: Maria Gabriela Caffarena Celani.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Civil, Arquitetura e Urbanismo.

1. Layout. 2. Algoritmos. 3. Projeto arquitetônico. 4. Modelagem de Informação da Construção. 5. Arquitetura e tecnologia. I. Celani, Maria Gabriela Caffarena, 1967-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Civil, Arquitetura e Urbanismo. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: The collaborative process between architects and algorithms in architectural design : a study on automed generation of furniture layouts as a design tool
Palavras-chave em inglês:
Layout
Algorithms
Architectural project
Building information modeling
Architecture and technology
Área de concentração: Arquitetura, Tecnologia e Cidade
Titulação: Mestre em Arquitetura, Tecnologia e Cidade
Banca examinadora:
Maria Gabriela Caffarena Celani [Orientador]
Márcio Minto Fabricio
David Moreno Sperling
Data de defesa: 24-05-2021
Programa de Pós-Graduação: Arquitetura, Tecnologia e Cidade

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-5185-9195
- Currículo Lattes do autor: https://orcid.org/0000-0002-5185-9195

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA CIVIL, ARQUITETURA E URBANISMO

O PROCESSO COLABORATIVO ENTRE ARQUITETOS E ALGORITMOS NA CONCEPÇÃO ARQUITETÔNICA:
Um estudo sobre a geração automatizada de layouts de mobiliário como ferramenta projetual

VERLEY HENRY CÔCO JÚNIOR

Dissertação de Mestrado aprovada pela Banca Examinadora, constituída por:

Profa. Dra. Maria Gabriela Caffarena Celani

Presidente e Orientadora/FEC, Unicamp

Prof. Dr. Márcio Minto Fabricio

IAU/USP

Prof. Dr. David Moreno Sperling

IAU/USP

A Ata da defesa com as respectivas assinaturas dos membros encontra-se no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 24 de maio de 2021

Aos meus pais.

Agradecimentos

À Prof. Dra. Gabi Celani que tem ensinado a mim e a tantos alunos o verdadeiro papel da docência. Sou grato pelo exemplo sincero, orientação humana e incentivo verdadeiro ao longo destes nove anos desde o início da minha graduação na Unicamp, entre disciplinas de graduação, monitorias, estágios, iniciações científicas, intercâmbios e boas conversas. Meus sinceros agradecimentos pela amizade e companheirismo.

Ao Prof. Dr. Cláudio Ferreira, Prof. Dr. Márcio Fabricio e Prof. Dr. David Sperling por aceitarem o convite de participar das bancas de qualificação e de defesa. Agradeço por se dedicarem à leitura desta dissertação e pelos comentários preciosos.

Ao meu pai, Verley Côco, que acreditou no valor da educação e com grande esforço abriu caminho para a minha formação acadêmica e profissional. Sou grato pelas conversas sobre lógica de programação e pelo entusiasmo ao ver cada novo resultado dos algoritmos. Meus agradecimentos pelo amor e cuidado.

À minha mãe, Eliana Côco, que viu que eu poderia superar obstáculos e conhecer a mim mesmo, dando-me asas para concluir este mestrado. Agradeço pela leitura desta dissertação e pelas sugestões de melhoria, foram contribuições de grande valor. Sou grato pela escuta ativa, pelos conselhos, amor e carinho.

Ao meu irmão, André Côco, que me ensina o significado de amizade e companheirismo. Meus sinceros agradecimentos pela colaboração na modelagem dos apartamentos que serviram para o desenvolvimento desta dissertação. Agradeço pela empatia, interesse sincero em ajudar e pelo entusiasmo.

À minha noiva, Karen Galvão, que ao longo destes anos tem sido sempre presente em uma relação de cumplicidade. Sou grato pelas trocas de conhecimento, conversas sobre metodologia de pesquisa e disposição em contribuir. Meus sinceros agradecimentos por vibrar comigo a cada vitória e por me lembrar das coisas que realmente importam na vida.

Aos amigos que apoiaram este trabalho de alguma forma, entre eles: Prof. Dra. Regina Tirello, Prof. Dr. Roberto Naboni, Prof. Dr. Felipe Tavares, Lucas Lima, Gabriel Galvão, Pedro Lazari, Amadeo Galdino, Caio Castriotto, Raquel Leite, Filipe Campos, Robson Canuto, Marcela Noronha, Thiago Gonzaga, Wilson Barbosa e Guilherme Giantini. Meus sinceros agradecimentos à equipe de funcionários e corpo docente da Faculdade de Engenharia Civil, Arquitetura e Urbanismo da Unicamp. Por fim, sou grato a Deus por ter me dado esta oportunidade acadêmica e por ter colocado estas pessoas no meu caminho.

Costuma-se dizer que uma pessoa não entende realmente algo até que ensine a outra pessoa. Na verdade, uma pessoa não entende algo até depois de ensiná-lo a um computador, ou seja, expressá-lo como um algoritmo.

(KNUTH, 1974, s.n., tradução nossa)

RESUMO

Apesar dos grandes avanços científicos e tecnológicos na área de Projeto Auxiliado por Computador (CAD), muitos processos repetitivos persistem no desenvolvimento de projetos de Arquitetura, mesmo em etapas que não envolvem tomada de decisão complexa. Entretanto, uma mudança de paradigma, já prevista e ensaiada por pioneiros da área, como Charles Eastman e Nicholas Negroponte, favoreceria uma simbiose entre arquitetos e máquinas em uma relação de colaboração. Esta dissertação parte do princípio de que a Modelagem da Informação da Construção (BIM) associada a uma abordagem algorítmica favoreceria uma desejada sistematização no desenvolvimento de projetos arquitetônicos desde a fase de concepção, por meio da automatização dessas tarefas. Dentro deste contexto, este trabalho teve como objetivo principal sistematizar processos de definição de layout de mobília, por meio de um fluxo de informações que integre a implementação de algoritmos em BIM. Para tanto, foi realizada uma revisão da literatura sobre a classificação de problemas de projeto, metodologias algorítmicas aplicáveis à temática trabalhada e sua implementação em BIM bem como o desenvolvimento e avaliação de uma ferramenta de geração de layouts de mobília. Adotou-se como metodologia o Design Science Research (DSR) e como método de implementação o desenvolvimento incremental de software. Os resultados apontam para a possibilidade de uma contribuição para as etapas de concepção mediante uma integração concreta com algoritmos, abrindo inclusive novas oportunidades para a atuação profissional e uma integração com Inteligência Artificial (IA) em trabalhos futuros.

Palavras-chave: geração automatizada de layouts; processo algorítmico; projeto arquitetônico; modelagem da informação da construção

ABSTRACT

Despite the significant scientific and technological advances in the field of Computer Aided Design (CAD), many repetitive processes persist in the development of architectural projects, even in stages that do not involve complex decision-making. However, a paradigm shift, already foreseen and rehearsed by pioneers in the area, such as Charles Eastman and Nicholas Negroponte, would favor a symbiosis between architects and machines in a collaborative relationship. This master thesis assumes that Building Information Modeling (BIM) associated with an algorithmic approach would favor a desired systematization in the development of architectural projects from the conception stage, through the automation of these tasks. Within this context, the main objective of this work was to systematize furniture layout definition processes, through a flow of information that integrates the implementation of algorithms in BIM. Therefore, a literature review was carried out on the classification of design problems, algorithmic methodologies applicable to the studied topic and its implementation in BIM, as well as the development and evaluation of a furniture layout generation tool. Furthermore, the Design Science Research (DSR) methodology was adopted, and the incremental software development method was used. Results indicate a potential contribution to the concept stages through a concrete integration with algorithms, including opening new opportunities for professional performance and an integration with Artificial Intelligence (AI) in future works.

Keywords: algorithmic process; architectural project; building information modeling

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AEC Arquitetura, Engenharia e Construção

BIM Modelagem da Informação da Construção (*Building Information Modeling*)

CAD Projeto Assistido por Computador (*Computer-Aided Design*)

CNC *Computer Numeric Control*

DSR *Design Science Research*

GAN Redes Neurais Generativas Adversárias

IA Inteligência Artificial

LPV Linguagem de Programação Visual

MSL Mapeamento Sistemático da Literatura

SP *Space Planning*

SUMÁRIO

1 INTRODUÇÃO

As tecnologias digitais têm proporcionado profundas mudanças socioeconômicas nas últimas décadas. Elas permitiram a geração e a troca de informações como nunca houve antes na história humana (ALPAYDIN, 2016) bem como o surgimento de novos modelos de negócio com um elevado nível de personalização em massa (CASTAGNINO *et. al*, 2018; SCHWAB, 2016). Desta maneira, a quarta revolução industrial vem sendo construída pelo rompimento das fronteiras entre o que é virtual, físico e biológico (SCHWAB, 2016). Apesar dessas transformações ocorrerem de forma acelerada em diversas áreas, a construção civil continuou operando como nos últimos cinquenta anos, isto é, com uma forte dependência de trabalhos manuais mesmo em tarefas sistematizáveis, tecnologia mecânica e modelos de operação e negócio já estabelecidos. A consequência é um ganho menor de produtividade em relação a outros setores industriais nas últimas décadas (CASTAGNINO *et. al*, 2018; KIERAN; TIMBERLAKE, 2003). Produtividade pode ser definida como "a taxa na qual uma empresa ou país produz mercadorias, geralmente avaliada em relação ao número de pessoas e à quantidade de materiais necessários para produzir as mercadorias" (CAMBRIDGE DICTIONARY, 2020,s.n, tradução nossa). De fato, seu aumento tem sido buscado por arquitetos desde a segunda metade do século XX com a implementação de ferramentas de Projeto Assistido por Computador (CAD) e Modelagem da Informação da Construção (BIM) (CELANI *et al.*, 2015).

Apesar deste "atraso" produtivo em relação aos demais setores industriais, como a indústria automobilística e aeronáutica (KIERAN; TIMBERLAKE, 2003), a construção civil tem grande relevância e impacto social, econômico e ambiental. Estima-se que 6% do PIB mundial está relacionado a este setor que emprega mais de 100 milhões de pessoas (CASTAGNINO *et. al*, 2018). Em 2018, por exemplo, o ambiente construído urbano foi responsável por 75% das emissões de gases estufa mundiais, sendo que os edifícios representaram sozinhos 39% (ARCHITECTURE 2030, 2018). Além disso, o setor é responsável por um grande descarte de materiais - 40% dos resíduos sólidos nos Estados Unidos, por exemplo (WEF, 2016). Assim, o ambiente construído é uma área estratégica que demanda uma revisão de conceitos e abordagens para maior controle do ciclo de vida dos edifícios. Novos mecanismos de sensores, máquinas inteligentes e novos softwares integrados ao BIM podem conduzir a uma maior digitalização da construção. Estima-se que uma digitalização em grande escala de construções não residenciais nos próximos dez anos proporcionaria uma economia global anual de US$ 0,7 trilhão a US$ 1,2 trilhão nas fases de

engenharia e construção e de US$ 0,3 trilhão a US$ 0,5 trilhão na fase de operações, isto é, de 13% a 21% e de 10% a 17%, respectivamente (GERBERT *et. al*, 2016). Para essa digitalização, porém, se faz necessária uma revisão de métodos já na fase de projeto, cujo equilíbrio de recursos humanos e materiais dentro de um cronograma e orçamento é estratégico para estruturação do setor. Uma má gestão e processos ineficientes nessa etapa resultam em informações incompatíveis, redundantes ou perdidas para a etapa de construção, ocasionando atrasos, extrapolação de orçamento e grande geração de resíduos. A definição de processos automatizáveis dentro desse fluxo de trabalho pode contribuir para maior controle sobre as informações de projeto.

Mudanças estratégicas no setor podem ser alcançadas por meio das tecnologias associadas à quarta revolução industrial. Elas abrangem desde a fase de projeto e construção até a operação e demolição, favorecendo o surgimento de ambientes mais colaborativos e integrados (CASTAGNINO *et. al*, 2018; GERBERT *et. al*, 2016; WEF, 2016). O BIM, por exemplo, se apresenta como tecnologia central para as demais devido a sua capacidade de armazenar, gerar e comunicar de forma mais controlada que metodologias tradicionais (OESTERREICH; TEUTEBERG, 2016). Gerbert *et. al* (2016, s.n., tradução nossa) afirmam que "as ferramentas de software integradas ao BIM conferem uma série de benefícios - como gerar e avaliar automaticamente alternativas de projeto, oferecer suporte à engenharia de valor e aprimorar análises de custo-benefício, projeto-fabricação e sustentabilidade". Concomitantemente, *big data e machine learning (*aprendizado de máquina) podem ser utilizados para organização e análise de grande quantidade de informações para cada uma das fases do empreendimento. Modelos tridimensionais, por sua vez, podem ser gerados para diferentes fins de simulação. Desempenho energético, detecção de conflitos ou faseamento da construção são exemplos de análises que podem ser obtidas por meio da prototipagem da construção. Esses modelos digitais podem ser representados por realidade virtual e aumentada inclusive pela integração com dispositivos móveis. Além dessas áreas de interface de usuário e software, há uma dissolução das fronteiras entre o meio digital e físico pelo avanço das tecnologias de manufatura aditiva e digitalização bem como pela maior autonomia de máquinas por controle numérico (CNC) diante das novas tecnologias de sensores e processamento de informação. Quando essas ferramentas são aplicadas na Arquitetura, elas abrem horizontes para projetos com desenvolvimento colaborativo e mais robusto, uma construção com compartilhamento de dados em tempo real, integração e melhor coordenação entre os envolvidos e, por fim, uma melhora na operação e manutenção do edifício (GERBERT *et. al*, 2016). Dentro deste novo cenário mais automatizado e

colaborativo, fazem-se necessárias discussões metodológicas sobre o projeto arquitetônico diante do avanço da Inteligência Artificial (IA).

Dentro deste contexto, diversas pesquisas vêm sendo realizadas sobre o arranjo automatizado dos elementos geométricos do projeto, compondo o campo do conhecimento denominado de *Space Planning* (SP). Entretanto, esse não é um tema recente na Arquitetura. Ele surge com o movimento do *Design Methods* na década de 1960 e se tornou um campo de pesquisa em si, com uso de técnicas sofisticadas de Inteligência Artificial, podendo ser aplicado desde o layout de móveis até o arranjo espacial de áreas de uma cidade.

O layout de mobiliário pode ser uma ferramenta importante para a tomada de decisões desde as etapas preliminares do projeto de edificações (DAS *et. al*, 2016). O seu arranjo é essencial para o entendimento da escala dos ambientes, da circulação, do conforto térmico e acústico, bem como para a disposição de pontos de elétrica, hidráulica e possíveis interferências com a estrutura. Assim, sua sistematização por meio de algoritmos pode contribuir para o processo de tomada de decisões subjetivas (KJØLAAS, 2000).

Apesar da discussão sobre o uso de algoritmos no processo de projeto ter sido iniciada há décadas, como se pode observar, por exemplo, em textos de Gropius (*apud* Rocha, 2004), Eastman (1968a, 1968b), Negroponte (1970) e Mitchell (1975), até hoje essa metodologia não foi implementada em larga escala nos escritórios de Arquitetura. Como resultado, nota-se claramente uma falta de produtividade nas áreas de Arquitetura, Engenharia e Construção (AEC), se comparada às áreas que já absorveram essas tecnologias. Em geral, existe um desconhecimento por boa parte dos profissionais da AEC sobre os benefícios dessas abordagens, cujo uso se restringe a um grupo restrito de grandes escritórios de Arquitetura e construtoras. Além disso, ainda existe um certo receio de que automatizar o processo de projeto poderia eliminar o processo criativo. A consequência direta disso é a subutilização da capacidade computacional e sua utilização simplesmente como ferramenta de desenho ou de modelagem, isto é, um nível pouco ambicioso de utilização (MITCHELL, 1975). Deixa-se de lado o potencial generativo e de análise, discutidos, por exemplo, por Eastman (1968), Gropius (ROCHA, 2004), Simon (1996), Stiny e Mitchell (1978) e Mitchell (1975).

Entretanto, é possível que a atual conjuntura de rápido desenvolvimento tecnológico favoreça e até mesmo exija uma maior integração de algoritmos ao longo do processo de projeto. Como destacado anteriormente, o BIM é um paradigma estratégico para que a construção civil se beneficie de outras tecnologias da quarta revolução industrial (GERBERT

et. al, 2016). Devido às suas características, entre elas a capacidade de atribuir, manipular e acessar informações associadas a modelos com semântica, a utilização do BIM pode ser uma oportunidade para a "algoritmização" dos processos em AEC.

Deste modo, esta dissertação tem como objetivo geral contribuir para a sistematização de processos de definição de layout de mobília por meio de sistemas generativos implementados em ambiente BIM, proporcionando um aumento da produtividade e a possibilidade de se considerar múltiplas alternativas desde as etapas iniciais de projeto.

Seus objetivos específicos são:

- Realizar uma revisão da literatura sobre classificação de problemas, algoritmos para automatização de layouts e a construção de algoritmos em BIM;
- Comparar métodos para automatização de layouts segundo as classes de problemas obtidas na Revisão da literatura;
- Desenvolver um algoritmo em BIM para a automatização de layouts de dormitórios de apartamentos, como prova de conceito;
- Testar o algoritmo desenvolvido em comparação com procedimentos convencionais como meio para avaliação.

Embora a temática aqui tratada pudesse estender-se para áreas correlatas ou de possíveis aplicações, como o ensino de Arquitetura, a relação entre arquitetos e seus clientes, Neurociência aplicada à Arquitetura e até mesmo um desenvolvimento prático de um sistema de Inteligência Artificial, por uma questão de limitação do tempo disponível no mestrado, optou-se por não abordar, neste momento, esses assuntos.

A partir dos objetivos elencados, esta pesquisa pode ser classificada como prescritiva e, quanto aos seus procedimentos, como construtiva e incremental, seguindo o delineamento do *Design Science Research* (LACERDA *et al.*, *2013;* DRESCH; LACERDA; ANTUNES JR., 2015*)*.

Esta dissertação está estruturada em (2) fundamentação teórica, na qual são abordados os temas de (2.1) classificação de problemas, (2.2) algoritmos para a automatização de layouts na Arquitetura e (2.3) a construção de algoritmos em BIM. Na sequência, os (3) materiais e métodos são apresentados, seguidos do (4) desenvolvimento

dos algoritmos criados. Nas últimas seções, (5) os procedimentos de avaliação são apresentados, seguidos da (6) conclusão e (7) referências.

2 FUNDAMENTAÇÃO TEÓRICA

> Considerarei apenas a terceira alternativa e tratarei o problema [do processo de design] como a associação íntima de duas espécies diferentes (homem e máquina), dois processos diferentes (design e computação) e dois sistemas inteligentes (o arquiteto e a máquina de arquitetura). Em virtude de atribuir inteligência a um artefato ou artificial, a parceria não é entre senhor e escravo, mas sim entre dois associados que têm um potencial e um desejo de autoaperfeiçoamento.
>
> (NEGROPONTE, 1970, s.n., tradução nossa)[1]

Diante do crescimento das aplicações de Inteligência Artificial nas mais diversas áreas do conhecimento, há um questionamento sobre quais seriam as fronteiras entre as tarefas a serem feitas por seres humanos ou por máquinas. Essa discussão, que gravita em torno do dualismo entre inteligência humana e artificial, acompanha o desenvolvimento dos computadores desde a criação do primeiro sistema de Inteligência Artificial, por Warren McCulloch e Walter Pitts (1943) (RUSSEL; NORVIG, 1995).

No campo da Arquitetura, não é diferente. Diversos estudos vêm sendo realizados para a obtenção de fluxos de trabalho mais eficientes, com melhor gestão da informação, maior capacidade de geração, de avaliação e de fabricação (CALIXTO; CELANI, 2015; DU *et al.*, 2020; OESTERREICH; TEUTEBERG, 2016; SÖNMEZ, 2018). Entretanto, a concepção e o projeto arquitetônico são inerentemente complexos pela multiplicidade de objetivos, requisitos, limitações e produtos desenvolvidos em um processo não linear (CELANI *et al.,* 2015; SÖNMEZ, 2018). Assim, com a atual quarta revolução industrial em curso, o aprofundamento das aplicações de automatização e aumento da capacidade computacional (ALPAYDIN, 2016), cabe a discussão sobre qual será o papel dos arquitetos nesta nova conjuntura e quais as possíveis relações entre eles e a automatização de processos na fase de concepção.

Um profundo processo de transformação tecnológica também ocorreu na segunda metade do século XX, desde o surgimento do primeiro sistema CAD, o Sketchpad, criado por Sutherland em 1963 (SUTHERLAND, 1963) e a popularização da utilização do computador como ferramenta de projeto nos escritórios de Arquitetura nas décadas seguintes (CELANI, 2002). Já na década de 1960, diversos autores se preocuparam em discutir como as metodologias computacionais poderiam ser aplicadas no projeto de

[1] No original: *I shall consider only the third alternative and shall treat the problem [of design process] as the intimate association of two dissimilar species (man and machine), two dissimilar processes (design and computation), and two intelligent systems (the architect and the architecture machine). By virtue of ascribing intelligence to an artifact or the artificial, the partnership is not one of master and slave but rather of two associates that have a potential and a desire for self- improvement.*

Arquitetura. Um exemplo disso foi a conferência "A*rchitecture and Computer*" realizada em Boston em 1964, marco da *Design Methods* (ROCHA, 2004), movimento que almejava aplicar procedimentos científicos no design em áreas como Arquitetura, projeto urbano e design industrial (LANGRISH, 2016). Na conferência de Boston estavam presentes arquitetos proeminentes como Walter Gropius, Serge Chermayeff, Christopher Alexander e Marvin Minsky, cofundador do laboratório de Inteligência Artificial do MIT (ROCHA, 2004). Gropius reconhecia o potencial das "máquinas" como meio para diminuir o processo de trabalho e com isso dar maior liberdade ao processo criativo, apesar de ser cauteloso em como isso poderia ser aplicado no dia a dia profissional. Ele afirmava:

> Parece que sempre erramos quando fechamos a porta muito cedo para sugerir novas potencialidades, sendo muitas vezes enganados por nossa inércia natural e aversão à necessidade de transformar nossos pensamentos. Por não estar em casa no vasto novo campo dos sistemas de computação, quero ser cauteloso. Ainda assim, acredito, se olharmos para essas máquinas como ferramentas potenciais para encurtar nossos processos de trabalho, eles podem nos ajudar a liberar nosso poder criativo [..] O que eu não posso imaginar ainda é um método praticável para o arquiteto ou projetista comum usar essas ferramentas quando são necessárias. A ênfase certamente estará na formulação inteligente das perguntas a serem respondidas pelo computador. Será então necessário formar uma nova profissão de auxiliar de arquitetura com o propósito de articular os problemas a serem resolvidos na própria linguagem do computador? [..] Enquanto isso, gostaria que nós, arquitetos, mantivéssemos a mente aberta às novas possibilidades que a ciência nos oferece. A crescente abrangência de nossas novas tarefas em Arquitetura e em desenvolvimentos urbanos precisa de novas ferramentas elaboradas para sua realização. Certamente caberá a nós, arquitetos, fazer uso deles de forma inteligente como meio de controle mecânico superior que pode nos fornecer uma liberdade cada vez maior para o processo criativo do projeto.
>
> (GROPIUS *apud* ROCHA, 2004. Pp. 90-91, tradução nossa)[2]

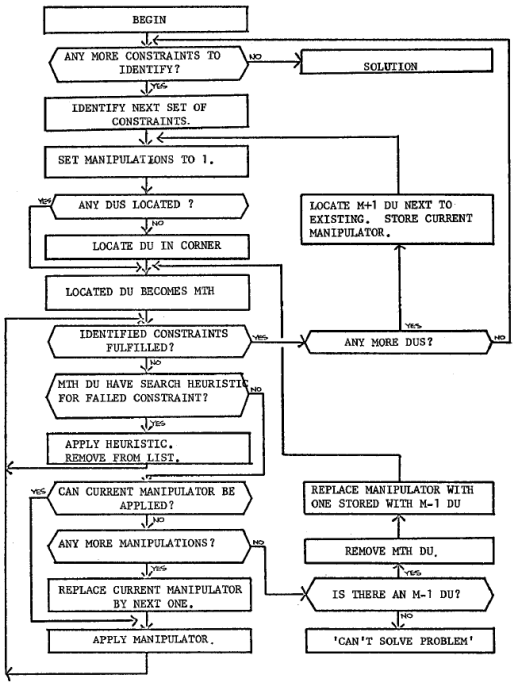
Essa discussão sobre como as ferramentas digitais poderiam ser empregadas adequadamente no processo de projeto, levando em consideração tarefas que permitiam ser automatizadas e a criatividade dos projetistas, foi tema de diversas pesquisas nesta época. Eastman (1968a, 1968b), por exemplo, investigou como determinados processos do projeto

[2] No original: *We seem to be always wrong when we close the door too early to suggested new potentialities, being often misled by our natural inertia and aversion to the necessity of transforming our thoughts. Being not at home in the vast new field of computer systems, I want to be cautious. Still I believe, if we look at those machines as potential tools to shorten our working processes, they might help us to free our creative power [...] What I cannot envisage yet is a practicable method for the average architect or designer to use these tools at the moment when they are needed. The emphasis will certainly be on the intelligent formulation of the questions to be answered by the computer. Will it then be necessary to educate a new profession of architectural assistants for the purpose of articulating the problems to be solved in the proper language of the computer? […] Meanwhile I wish we architects would keep an open mind towards the new possibilities offered us by science. The increasing comprehensiveness of our new tasks in architecture and in urban developments needs new elaborate tools for their realization. It will certainly be up to us, architects to make use of them intelligently as means of superior mechanical control which might provide us with ever-greater freedom for the creative process of design.*

cognitivo poderiam ser traduzidos em algoritmos (Figura 1) e, posteriormente, tê-los implementados em um computador. Para isso, realizou experimentos por meio de protocolos nos quais os projetistas deveriam verbalizar as decisões tomadas. O autor demonstra que certas tarefas feitas intuitivamente poderiam ser automatizadas, porque se estruturam de modo procedural, mesmo que inconscientemente. A capacidade iterativa superior dos computadores viria a elevar a criatividade humana:

> Apenas uma máquina seria paciente o suficiente para realizar tais simulações recursivas, mas é uma área de pesquisa que oferece grandes recompensas em potencial. Tais recursões podem exigir apenas a estruturação apropriada de DUs [Unidades de Projeto] e áreas representacionais separadas na memória. As simulações de máquina organizadas hierarquicamente podem aumentar significativamente sua "criatividade".
>
> (EASTMAN, 1968a, p. 78-79, tradução nossa)[3]

Figura 1 - Diagrama de blocos para um processo administrativo desenvolvido por Eastman (1968a)



Fonte: Eastman (1968a, p. 73)

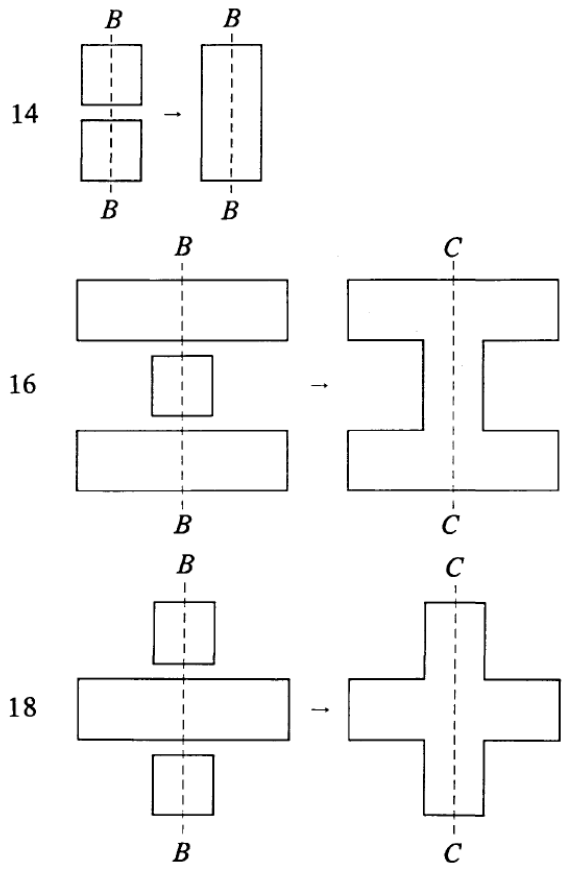
Stiny e Mitchell (1978), por sua vez, analisam as convenções de Paládio para o projeto de vilas e sua possível explicitação em regras, por meio de uma gramática da forma. Os autores partem do princípio de que existem regras intuitivas de projeto associadas a estilos arquitetônicos. Sua correta estruturação poderia vir a compor um sistema de análise dos

[3] No original: *Only a machine would be patient enough to carry out such recursive simulations, yet it is a research area offering rich potential rewards. Such recursions may only require the appropriate structuring of DUs [Design Units] and separate representational areas in memory. Hierarchically organizing machine simulations may significantly enhance their 'creativity'.*

edifícios existentes e geração de novos projetos (STINY; MITCHELL, 1978; SÖNMEZ, 2018). Exemplos de regras dessa gramática podem ser observados na Figura 2.

Os trabalhos de Eastman (1968a, 1968b) e Stiny e Mitchell (1978) são exemplos de pesquisas que lidam com problemas espaciais, isto é, de *Space Planning*. Essa disciplina estuda problemas relacionados à adjacência, à distância e a outras funções de arranjo de elementos espaciais (EASTMAN, 1973). Assim, estão inclusas nesta área desde layouts de mobiliários, fachadas e espaços desde a escala da Arquitetura até a das cidades (SÖNMEZ, 2018).

Figura 2 - Exemplos de regras da gramática desenvolvida por Stiny e Mitchell (1978) para geração de ambientes



Fonte: Stiny e Mitchell (1978, p. 9)

Negroponte (1970) também reconhece o potencial dos computadores como uma ferramenta para aumentar a velocidade e reduzir o custo de tarefas ao mesmo tempo que métodos existentes poderiam ser adaptados e especificados em sistemas de computador. Esses usos são focados na resolução de problemas (*problem-solving)* para casos bem definidos. Entretanto, o autor afirma que essas são aplicações limitadas, pois muitos problemas na Arquitetura são mal definidos e complexos. Assim, um computador executará ações dentro do recorte pré-estabelecido pelos projetistas, ou seja, sujeito a informações limitadas ou

incompletas. Ele escreve: "essas máquinas fariam apenas as tarefas monótonas, e fariam essas tarefas empregando apenas os procedimentos e as informações que os projetistas explicitamente lhes deram" (NEGROPONTE, 1970, p. 119, tradução nossa)[4]. Por outro lado, o autor apresenta um terceiro caminho pelo qual a relação humano-máquina seria de associação e de evolução mútua. Neste caso, o computador seria utilizado para problematizar situações (*problem-worrying),* sendo capaz de potencializar a cognição humana. Dentro deste contexto, ele escreve:

> Para fazer isso, uma máquina de arquitetura deve entender nossas metáforas, deve solicitar informações por conta própria, deve adquirir experiências, deve falar com uma ampla variedade de pessoas, deve melhorar com o tempo e deve ser inteligente. Deve reconhecer o contexto, particularmente as mudanças nos objetivos e as mudanças no significado ocasionadas pelas mudanças no contexto (NEGROPONTE, 1970, p. 119, tradução nossa)[5].

A máquina arquitetônica (*architecture machine)* seria como uma "espécie", com grande capacidade de adaptação a novos problemas e a novos contextos, porém com uma habilidade maior de executar tarefas que os seres humanos. O autor defende uma simbiose entre as duas espécies (ser humano e máquina) que estão em constante evolução (NEGROPONTE, 1970). Mitchell (1975) afirma que essa abordagem descrita por Negroponte (1970) seria o uso mais ambicioso dos computadores, justamente pela sua habilidade em lidar com problemas mal definidos.

A aplicação de algoritmos no processo de projeto e as suas implicações na liberdade criativa dos projetistas foi problematizada por Gropius (*apud* Rocha, 2004), Eastman (1968a, 1968b), Negroponte (1970) e Mitchell (1975). Percebe-se que um ponto fundamental na discussão é a distinção entre problemas bem definidos e mal definidos, que podem conduzir a abordagens distintas de "algoritmização" sobre os projetos de Arquitetura. No capítulo seguinte, as classificações de problemas são discutidas. Na sequência, aborda-se como algoritmos podem ser aplicados à automatização de layouts.

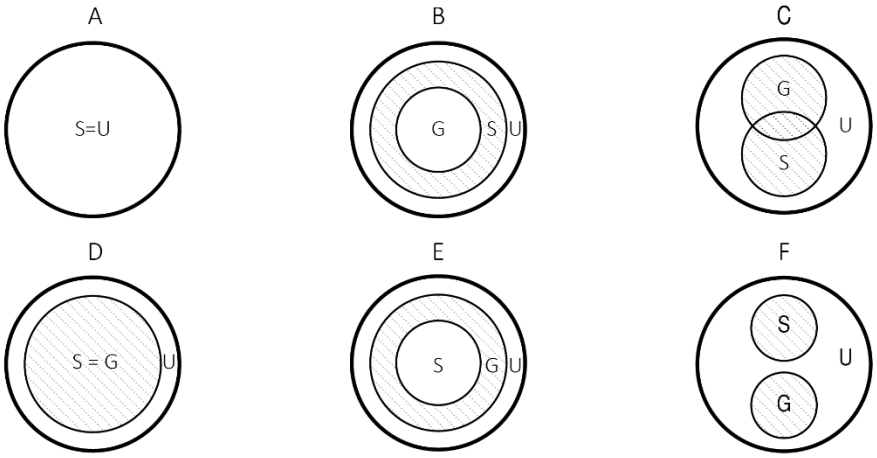
2.1 Classificação de problemas

A partir da separação entre "problema" e "solução" é possível elencar três classes de problemas: os bem definidos, os mal definidos e os malcomportados (*wicked),* uma

[4] No original: *These machines would do only the dull ignoble tasks, and they would do these tasks employing only the procedures and the information designers explicitly give them.*
[5] No original: *To do this, an architecture machine must understand our metaphors, must solicit information on its own, must acquire experiences, must talk to a wide variety of people, must improve over time, and must be intelligent. It must recognize context, particularly changes in goals and changes in meaning brought about by changes in context.*

subcategoria dos problemas mal definidos (ROWE, 1991). Como pano de fundo para a discussão, é importante explicitar relações possíveis entre o universo de todas as soluções possíveis (U), o conjunto de soluções viáveis (S) e o conjunto de soluções desejáveis (G), de acordo com o modelo de Mitchell (1975). Conforme pode ser visto na Figura 3, a relação "A" mostra uma situação em que U é igual a S, quando todas as soluções possíveis são viáveis. No caso "B", S é um subconjunto de U diante da exclusão implícita de possíveis soluções potenciais por não ser possível produzi-las. Em "C", S e G possuem uma intersecção, apontando que a fase de geração produz soluções viáveis e inviáveis, desejáveis ou não. Na sequência, em "D", S é igual a G, assim não há casos insatisfatórios dentre os viáveis. Em "E", há a geração de apenas algumas soluções viáveis dentre as desejáveis. Por último, em "F", nenhuma solução desejável é ao mesmo tempo viável, revelando que o procedimento de geração falhou.

Figura 3 – Relações entre soluções possíveis, geração de soluções e objetivos

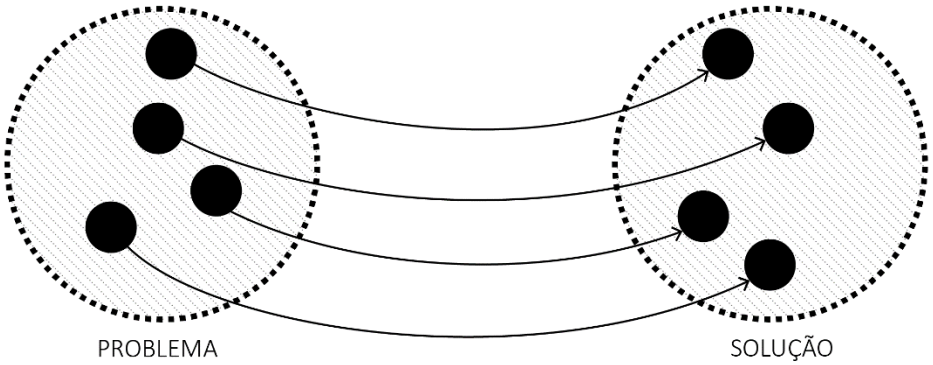


Fonte: adaptado de Mitchell (1975)

Ao longo do desenvolvimento de um projeto de Arquitetura, os seis casos descritos por Mitchell (1975) podem ocorrer inconscientemente ao se deparar com problemas mais ou menos definidos, dependendo do nível de entendimento do problema. A estruturação de problemas bem definidos, mal definidos e malcomportados (*wicked)* estão descritos nos parágrafos seguintes.

2.1.1 Problemas bem definidos

Figura 4 - Modelo para problemas bem definidos



Fonte: elaborado pelo autor

Problemas bem definidos (Figura 4), também chamados de bem estruturados ou domados, podem ser descritos como "[..] qualquer problema no qual o estado inicial ou posição inicial, as operações permitidas e o estado de objetivo são claramente especificados e uma solução única pode ser mostrada para existir" (COLMAN, 2015, s.n., tradução nossa)[6]. De modo semelhante, outro autor considera um problema bem definido quando "os fins, ou meta, já estão prescritos e aparentes; sua solução requer o fornecimento de meios apropriados" (ROWE, 1991, p. 40, tradução nossa)[7]. O enquadramento nesta classificação depende da boa clareza das entradas, dos procedimentos e dos objetivos. Jogos de tabuleiro como palavras-cruzadas são bons exemplos deste tipo de problema, haja vista que as entradas, operações e resultados para cada uma delas já é pré-determinada e conhecida (ROWE, 1991). Determinados casos também podem ser encontrados na Arquitetura. Rowe (1991) exemplifica que determinados problemas de *Space Planning* podem ser classificados como bem definidos na medida que possuem a prescrição de espaços e requisitos de adjacência de modo que se obtenha uma combinação aceitável.

Um possível exemplo de *Space Planning* que se aproxima de um problema bem definido, pode ser visto no trabalho Eastman (1968a; 1968b; 1969). Como descrito anteriormente, o autor demonstra que determinadas tarefas executadas manualmente por arquitetos poderiam ser transcritas na forma de algoritmos e, posteriormente, automatizadas por um sistema de computador. Isso ocorre porque aspectos do processo cognitivo seguem uma lógica sequencial algorítmica. Eastman (1968a) descreve o projeto arquitetônico como

[6] No original: *any problem in which the initial state or starting position, the allowable operations, and the goal state are clearly specified, and a unique solution can be shown to exist.*
[7] No original: *the ends, or goal, are already prescribed and apparent; their solution requires the provision of appropriate means.*

etapas de identificação de problemas, coleta de dados, análise, síntese e avaliação, ou simplesmente como identificação, geração e integração. Ao longo dessas etapas, os projetistas trabalham com "unidades de projeto" que constituem os elementos, dados para determinada classe de problema. As unidades de projeto são selecionadas e desenvolvidas no projeto a partir de uma memória semântica, de modo que os projetistas as buscam em classes de problemas similares na literatura ou em seu próprio repertório. Ao longo do projeto, elas são refinadas com informação, como a relação espacial com outras unidades de projeto, originando as restrições por objetivo ou internas a elas mesmas.

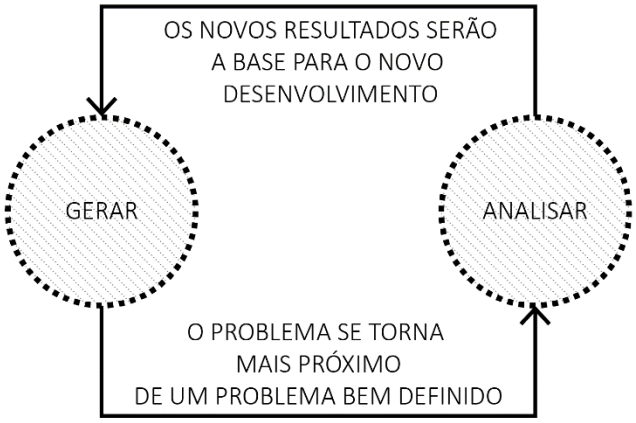
Nesse mesmo trabalho, Eastman (1968a) segue uma metodologia experimental por meio de quatro protocolos nos quais os projetistas tinham que verbalizar suas intenções para o projeto de um banheiro. Ao final, foram identificadas trinta e três restrições, como: "(C1) a área principal da sala tem uma área livre mínima, [..] (C3) orçamento de cinquenta dólares, [..] (C12) inclui materiais texturizados, [..] (C18) distância entre o piso e as torneiras, [..] (C20) sem janela frontal do banheiro, [..] (C33) toalheiros perto da pia e da banheira" (EASTMAN, 1968, p.48). De modo semelhante foram observadas vinte e quatro Unidades de Projeto, por exemplo: "(DU1) banheira, [..] (DU4) pias, [..] (DU5) espelho, [..] (DU10) janela, (DU15) materiais de parede, [..] (DU20) luminárias, [..] (DU24) torneiras para banheira" (EASTMAN, 1968a, p.49). Por fim, foram obtidas onze manipulações, entre elas: "(M1) localizar a unidade em um canto (de uma luminária ou sala), [..] (M2) girar a unidade na localização atual, [..] (M6) envolva a unidade, [..] (M10) alinhe, [..] (M11) localize sobre a pia" (EASTMAN, 1968a, p.48, tradução nossa)[8]. As restrições, unidades de projeto e manipulações foram articuladas em sequências de ações e traduzidas como algoritmos expressos em diagramas de blocos. O autor conclui que foi possível simular o processo intuitivo de projeto de banheiros em seus aspectos funcionais em uma estruturação não hierárquica e heurística.

Mitchell (1975) olha para os estudos de Eastman (1969) sobre layout de banheiros a partir do trabalho de Reitman (1965 *apud* MITCHELL, 1975). Reitman (1965 *apud* MITCHELL, 1975), ao analisar processos de solução de problemas, afirma que muitos critérios são deixados em aberto na declaração inicial do problema, mas que devem ser fechados pelos projetistas ao longo do projeto. Mitchell (1975) descreve que esse processo

[8] *No original:(C1) main area of room has a minimum clear area, [...] (C3) fifty dollar budget, [...] (C12) include textured materials, […] (C18) distance between floor and faucets, […] (C20) no bath front window, […] (C33) towel racks near sink and bath" (EASTMAN, 1968, p.48). "(DU1) Bathtub, [...] (DU4) Sinks, [...] (DU5) Mirror, [...] (DU10) Window, (DU15) Wall materials, [...] (DU20) Light fixtures, […] (DU24) Bathtub faucets" (EASTMAN, 1968a, p.49). "(M1) Locate Unit at a corner (of a fixture or room), [...] (M2) Rotate Unit at present location, [...] (M6) Enclose Unit, […] (M10) Align, […] (M11) Locate over sink.*

de fechamento pode ocorrer antes da busca por uma solução ou durante o processo. Isso pode ser feito por meio de entrevistas com os clientes, aplicação de questionários e por pesquisa na literatura, de modo que se consiga fechar o máximo de questões possíveis. Entretanto, Mitchell (1975) afirma que, muitas vezes, a única maneira de se fechar algumas das condicionantes é por ciclos de execução do projeto e análise crítica dos resultados (Figura 5). Assim, após inúmeros exercícios, o problema se aproxima de bem definido quando a solução final é encontrada, como ocorre no trabalho de Eastman (1969).

Figura 5 – Ciclo para maior definição de um problema

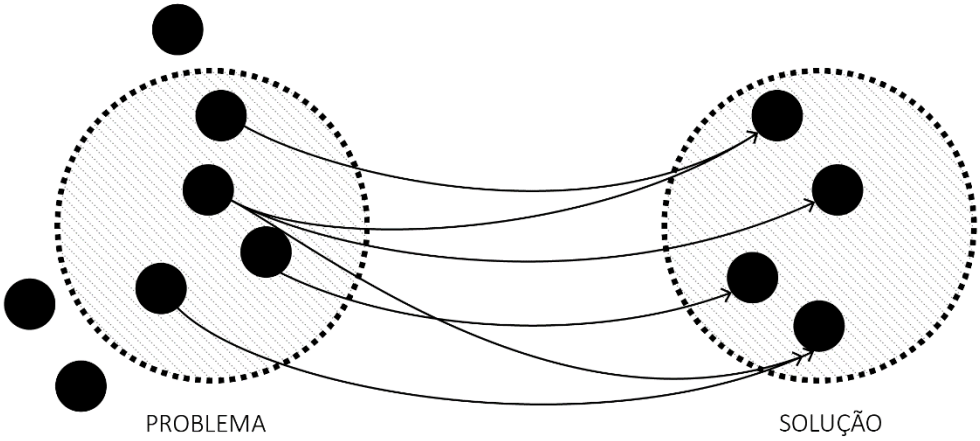


Fonte: elaborado pelo autor com base em Mitchell (1975)

Essa característica de apresentar condicionantes em aberto no projeto corrobora para boa parte dos problemas em Arquitetura serem classificados como mal definidos, como será explicado a seguir, diferentemente do que ocorre no caso A e D descritos por Mitchell (1975), onde há um total controle e conhecimento da geração de soluções em relação ao objetivo e é possível implementar algoritmos procedurais para a sua resolução.

2.1.2 Problemas mal definidos

Figura 6 – Modelo para problemas mal definidos



Fonte: elaborado pelo autor

Grande parte dos problemas existentes no projeto de Arquitetura podem ser classificados como mal definidos, como se vê na Figura 6 (MITCHELL, 1975; ROWE, 1991). Eles são descritos como "[..] qualquer problema em que a posição inicial, as operações permitidas ou o estado da meta não estejam claramente especificados, ou não seja possível demonstrar que existe uma solução única" (COLMAN, 2015, s.n, tradução nossa)[9]. Mitchell (1975) apresenta cinco características dos problemas mal definidos:

- Não há uma formulação completa ou definitiva no início. O problema é abstraído de um contexto emaranhado e com muitas informações. As condicionantes de projeto não são declaradas explicitamente, dependendo muitas vezes da inferência dos projetistas;
- A formulação obtida não é rigorosa. Tanto as soluções possíveis a serem investigadas quanto os critérios para alcançá-las não são holísticos, nem totalmente claros;
- A formulação não parece estável. As condicionantes e os critérios são alterados ao longo da exploração das soluções, inclusive por meio de novas ideias e percepções sobre o problema adquiridas ao longo do desenvolvimento;

[9] No original: *[…] any problem in which either the starting position, the allowable operations, or the goal state is not clearly specified, or a unique solution cannot be shown to exist.*

- A formulação pode incorporar conflitos e inconsistências que devem ser resolvidas ao longo do desenvolvimento. Além disso, eles só aparecem quando se inicia o processo de busca por uma solução;
- Os projetistas podem não possuir todas as informações necessárias em todos os momentos. Informações podem ser perdidas e recuperadas ao longo do desenvolvimento, conduzindo a novos conjuntos de soluções possíveis.

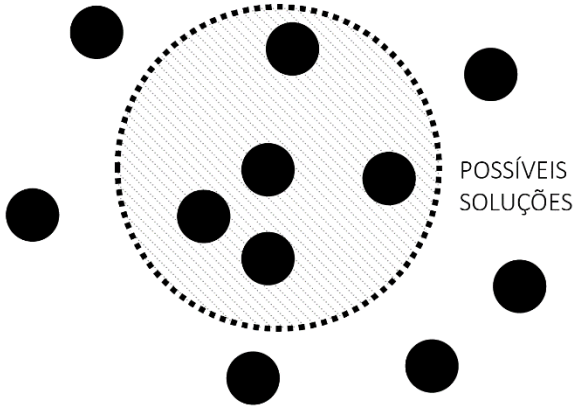
Deste modo, a maior parte dos problemas arquitetônicos se enquadra nesta categoria, devido à necessidade de equilibrar aspectos técnicos, subjetivos e de contexto ao longo de um processo de criação que não é linear (SÖNMEZ, 2018). Um bom exemplo de problema mal definido na Arquitetura é quando um cliente contrata um arquiteto ou arquiteta para um projeto de uma residência. Mesmo que o objetivo do projeto seja definido, serão necessárias inúmeras reuniões para esclarecer pontos suficientes do projeto para a sua execução (ROWE, 1991).

Desde os anos 1960, diversas pesquisas já vinham sendo realizadas a fim de se obter maior controle sobre problemas em Arquitetura por meio de automatização. Mitchell (1975, p. 15, tradução nossa)[10] conclui que: "parece provável que os avanços na pesquisa de inteligência artificial nos próximos anos resultarão na implementação de sistemas capazes de responder com eficácia a situações problemáticas cada vez mais mal definidas. Esses desenvolvimentos devem ter implicações importantes para o projeto arquitetônico automatizado".

[10] No original: *It seems likely that advances in artificial-intelligence research over the next few years will result in implementation of systems capable of responding effectively to increasingly ill-defined problem situations. These developments should have important implications for automated architectural design.*

2.1.3. Problemas *wicked*

Figura 7 – Modelo para problemas *wicked*



Fonte: elaborado pelo autor

A classificação de determinados problemas como malcomportados ou *wicked* (Figura 7) foi defendida por Rittel como o principal conjunto de problemas para designers (BUCHANAN, 1992). Apesar de ser colocado também como problema mal definido, problemas *wicked* apresentam características próprias. Rittel os descreve como uma "classe de problemas do sistema social que estão mal formulados, onde as informações são confusas, onde há muitos clientes e tomadores de decisão com valores conflitantes e onde as ramificações em todo o sistema são totalmente confusas" (RITTEL, 1967, B-141-42 *apud* BUCHANAN, 1992, p.15, tradução nossa)[11]. Esta abordagem rompe com o modelo de projeto linear estruturado em "definição" e "solução de problema". Isto é, por um lado, um rompimento entre o processo analítico de especificar todos os elementos e condicionantes envolvidos em um problema e, por outro, do processo de síntese que busca ponderar todos esses fatores em uma solução (BUCHANAN, 1992). Buchanan (1992) sugere que problemas mal definidos e *wicked* existem no processo de projeto pela sua própria natureza que, apesar de possuir aspectos universais, sempre passa por um processo subjetivo.

Rittel (1972) e Rittel e Webber (1973) descrevem dez características dos problemas *wicked*:

- Um problema *wicked* não tem uma formulação definitiva. Diferente de um problema bem definido que pode ser completamente descrito e, eventualmente, ser resolvido

[11] No original: *class of social system problems which are ill-formulated, where the information is confusing, where are many clients and decision makers with conflicting values, and where the ramifications in the whole system are thoroughly confusing.*

por diferentes pessoas, a definição de um problema *wicked* se aproxima mais da sua solução. Isso ocorre porque a própria informação necessária para entender o problema se refere ao mesmo tempo à solução, consequentemente a compreensão da questão só pode ser vista por meio dela. Rittel e Webber (1973) afirmam, por exemplo, que um problema de engenharia do programa espacial segue uma sequência clara de etapas, como: entender o problema ou a missão, colher informação, analisar os dados, sintetizar essas informações, esperar por um salto criativo e trabalhar na solução. Entretanto, esse procedimento não funciona para problemas *wicked* pela sua não linearidade e complexidade;

- Não possui uma regra de parada devido à inseparabilidade entre problema e solução, pois encontrá-la é entender a própria natureza do problema. Em um problema bem definido, como em um jogo de xadrez ou em uma equação, existem regras bem estabelecidas que determinam quando a solução foi encontrada. Entretanto, em problemas *wicked* não existem razões lógicas inerentes ao sistema que determinam a solução. O término da sua busca vem por meios externos, como esgotamento de recursos financeiros ou de tempo, pela limitação ou escolha do próprio projetista, por exemplo;

- As soluções para problemas *wicked* não são verdadeiras nem falsas, mas boas ou ruins. Para esta classe de problemas é impossível definir uma única solução; ela é relativa dependendo do contexto. Como Rittel (1972) exemplifica, isso ocorre, por exemplo, em projetos urbanísticos que não podem ser classificados como verdadeiros ou falsos, mas como melhores ou piores, dependendo do contexto;

- Não há um teste imediato nem definitivo para uma solução de problema *wicked*. Diferentemente de um problema bem definido, no qual há um controle sobre as soluções, neste caso a solução implementada gerará consequências não previstas. Pode-se, inclusive, retroceder em relação aos benefícios obtidos pela solução diante da quantidade de novos problemas a serem resolvidos;

- Toda solução para um problema *wicked* é única, porque não é possível aprender por tentativa e erro, sendo que cada uma delas conta significativamente. Em áreas como engenharias, matemática ou jogos de tabuleiro, a resolução de problemas pode passar por várias tentativas sem nenhuma penalidade fora do próprio sistema ou do indivíduo. Entretanto, no caso de problemas *wicked* as ações possuem uma reação em cadeia incontrolável e irreversível. Não é possível fazer experimentações;

- Não existe uma lista exaustiva de soluções possíveis para problemas *wicked*. Não existe um critério que determine que todas as variáveis para determinado problema foram consideradas. Assim, ao se encontrar uma solução, ela será baseada em um julgamento parcial de possibilidades;
- Todo problema *wicked* é único. Não é possível definir uma classe de problemas na qual uma mesma solução possa ser aplicada em todos os casos. Apesar de possíveis similaridades, sempre existirão diferenças entre dois problemas;
- Todo problema *wicked* pode ser considerado sintoma de outro problema de nível maior. Essa propriedade dificulta sua resolução, pois não há certeza que se esteja atuando sobre o nível adequado para a sua resolução, haja vista que é sintoma de outro problema;
- A existência de uma discrepância representando um problema *wicked* pode ser explicada por diferentes caminhos. Um problema pode ser definido como uma discrepância entre o que algo é e como deveria ser. Entretanto, neste caso, ele se origina de diversos outros problemas. Quando se tenta explicar o problema principal por uma das suas origens, a solução será enviesada pela escolha;
- Aquele que resolve um problema *wicked* não tem o direito de estar errado. Devido às características descritas anteriormente, há um nível maior de responsabilidade associada à resolução de problemas *wicked*.

O processo de projeto na Arquitetura oscila entre problemas bem definidos, mal definidos e *wicked*, variando de acordo com o escopo e variáveis envolvidas. Dentro deste contexto, determinadas ações podem ser "algoritmizadas" com o objetivo de reduzir o esforço com tarefas repetitivas e mesmo de explorar possíveis soluções não encontradas somente pela cognição. No capítulo seguinte, faz-se uma revisão da literatura sobre como algoritmos estão sendo aplicados na geração de layouts de Arquitetura.

2.2 Algoritmos para automatização de layouts na Arquitetura

Os algoritmos estão presentes em uma variedade de situações no cotidiano e seu impacto é cada vez maior na sociedade. Em síntese, eles podem ser definidos como "um conjunto de instruções que realizam uma tarefa" (BHARGAVA, s.n.), "uma sequência de instruções que especifica como executar uma operação de computador" (DOWNEY, 2018,

s.n.) bem como "qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores de saída. Portanto, é uma sequência de passos computacionais que transformam a entrada na saída" (CORMEN *et al.*, 2002, p. 3). Na mesma orientação, Kronsjö (1987) afirma que os algoritmos possuem quatro características: (1) cada uma de suas etapas não tem ambiguidades; (2) chegam a uma solução para determinado problema após a execução de um número finito e razoável de etapas; (3) processam dados de entrada e retornam valores; e, por último, (4) têm (preferencialmente) a capacidade de responder a mais de um conjunto de problemas.

Em essência, o pensamento algorítmico independe do uso de computadores. Euclides, por exemplo, por volta de 300 a.C., desenvolveu um algoritmo para encontrar o máximo divisor comum entre dois números inteiros e o próprio termo foi cunhado pelo matemático persa Al-Khowârizmî no século 9 d.C., ou seja, muito antes da Era Digital. Hoje, porém, os algoritmos podem ser executados por computadores. Alpaydin (2016, p. 1, tradução nossa)[12] afirma que "algumas das maiores transformações em nossas vidas na última metade do século se devem à computação e à tecnologia digital". Eles são a inteligência dos sites de busca e viabilizam a existência das redes sociais, compõem os cálculos bancários, as bolsas de valores, os sistemas de navegação de aeronaves, os sistemas operacionais de computadores e celulares, bem como são fundamentais na indústria para a racionalização da produção (CORMEN *et al.*, 2002).

Em relação à Arquitetura, muitas vezes a terminologia para os algoritmos aplicados se confunde com outros termos, como "projeto paramétrico", "projeto associativo" e "projeto algorítmico". Apesar da parametrização e criação de algoritmos estarem relacionadas, referem-se a conceitos diferentes. O primeiro deles remete à associação de variáveis ou constantes. Por exemplo, se "A" e "B" possuem uma relação de causa e efeito, se "A" for modificado, isso afetará diretamente "B". Os programas dentro do paradigma BIM, por exemplo, são essencialmente paramétricos, pois os dados contidos em elementos construtivos podem ser associados a tabelas ou a qualquer vista para visualização. Quando essas informações são modificadas diretamente em um elemento, em uma tabela ou em algum desenho, elas são automaticamente atualizadas nas outras. Isso difere do processo algorítmico, cujas definições apresentadas anteriormente remetem a uma sequência de passos

[12] No original: *some of the biggest transformations in our lives in the last half century are due to computing and digital technology.*

para a resolução de um problema. Em geral, relações paramétricas costumam estar incluídas nos sistemas algorítmicos (LEACH, 2012).

Diversos tipos de algoritmos podem ser aplicados na Arquitetura. Uma classificação possível é separá-los em determinísticos e não determinísticos, dentre os quais se pode destacar (RACEC; BUDULAN; VELLIDO, 2016):

- Algoritmos que não aprendem. Algoritmos que possuem entradas e restrições bem definidas nos quais os problemas a serem resolvidos podem ser declarados exaustivamente. Geralmente, são aplicados para problemas bem definidos, como ambientes retangulares, ou com especificações rigorosas;
- Algoritmos procedurais. Neste caso, uma sequência de passos e regras são determinadas para a execução de tarefas. Um paradigma que se utiliza de processos procedurais é a gramática da forma, por exemplo;
- Algoritmos probabilísticos. Por meio desta abordagem, visa-se um conjunto de soluções aceitáveis ao invés de buscar soluções específicas. Racec, Budulan e Vellido (2016) destacam que o uso de algoritmos probabilísticos pode ser vantajoso para a resolução de problemas de layout, entre outros motivos, pela possibilidade de se chegar a uma solução sem a necessidade de especificar todos os requisitos;
- Otimizações estocásticas. Estes algoritmos se utilizam de números aleatórios no processo de otimização, os quais atribuem peso para as alternativas. Consequentemente, há diferentes respostas a cada execução do algoritmo (CALIXTO, 2016). Assim, métodos estocásticos podem fazer uso de funções objetivo para alcançar resultados melhores (RACEC; BUDULAN; VELLIDO, 2016);
- Algoritmos evolutivos. Neste caso, os algoritmos procuram simular métodos baseados na seleção natural, isto é, são sistemas que simulam a combinação de genes (quaisquer características), hereditariedade, reprodução de indivíduos, variações que afetam a probabilidade de sobrevivência dos indivíduos e competição entre eles (CALIXTO, 2016; DAS et. al, 2016). Racec, Budulan e Vellido (2016) descrevem que esta abordagem pode resultar em grande diversidade de resultados;
- Algoritmos generativos. Algoritmos capazes de gerar formas, servindo como intermediadores no processo de tomada de decisões do projeto (CALIXTO, 2016).

Algoritmos vêm sendo aplicados para a automatização de tarefas no projeto de Arquitetura nos âmbitos de interpretação, geração e avaliação (SÖNMEZ, 2017). Dentro deste contexto, o *Space Planning* é um dos caminhos possíveis para a sistematização referente ao arranjo automatizado de layouts de mobiliários. Leite e Celani (2019) realizaram um Mapeamento Sistemático da Literatura (MSL) sobre layouts espaciais e de mobiliário automatizados por meio das bases Web of Science, Scopus, Avery e Cumulative Index of Computer-Aided Architectural Design (CumInCAD). As autoras chegaram a cinquenta e oito artigos publicados entre 2008 e 2018, sendo 78% das publicações relativas aos últimos cinco anos. Os resultados apontam para um significativo aumento das publicações nesta área nos últimos anos. Verifica-se uma diversidade de metodologias: gramática da forma, projeto paramétrico ou associativo, algoritmos evolutivos, algoritmos genéticos, grafos, simulação baseada em desempenho ambiental ou ocupação humana, sintaxe espacial e *machine learning* . Além disso, destaca-se a integração de estratégias algorítmicas e generativas ao BIM, visando melhor coordenação da documentação. Outro aspecto analisado é para qual momento os layouts automatizados são desenvolvidos. Os resultados evidenciam que a maior parte dos trabalhos se concentra na fase de pré-configuração, sendo o desenvolvimento feito, em sua maioria, somente por projetistas. A questão da participação dos usuários no projeto por meio de personalização em massa e da criação de algoritmos foi amplamente discutida em outro trabalho (LEITE, 2020). Em resumo, o trabalho de Leite e Celani (2019) destaca o crescimento da relevância da área e a possibilidade da exploração da integração de novas ferramentas e metodologias.

Diante da possibilidade de "algoritmização" de etapas do processo de projeto, esta dissertação aborda o tema de layout de mobiliário dentro da fase de pré-configuração, um tema discutido por Bazalo e Moleta (2015); Biagini, Donato e Pellis (2015) e Langenhan *et al.* (2013). Desenvolve-se aqui o layout criado diariamente nos escritórios de Arquitetura nas etapas iniciais do processo de projeto, como subsídio para análises projetuais posteriores. Assim, este layout que, nas etapas preliminares carrega somente informações de tamanho, posicionamento e orientação, pode ser entendido como uma ferramenta projetual (DAS *et. al*, 2016; KJØLAAS, 2000). A automatização deste layout inicial permite uma rápida simulação da espacialidade almejada pela Arquitetura, e ao mesmo tempo contribui para a tomada de decisões sobre áreas e dimensões mínimas e desejáveis, além de possibilitar a automatização da definição de outros elementos construtivos, como o layout de elétrica, ajudando, por exemplo, na estimativa de custos.

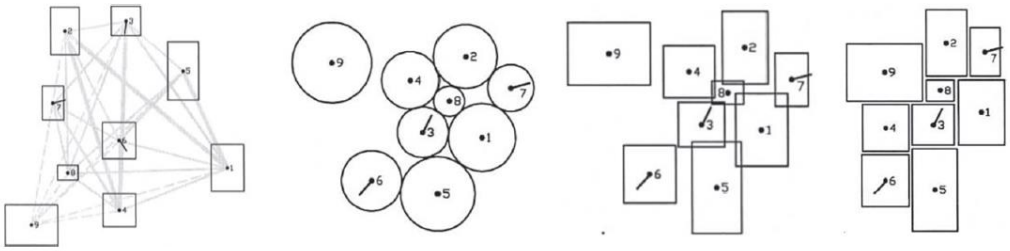
Diversas estratégias têm sido adotadas para a automatização de layouts. Algumas delas são apresentadas no capítulo seguinte. Em seguida, estudos de casos chaves são abordados, sendo divididos em algoritmos determinísticos e não determinísticos e, por fim, uma discussão é realizada levando em consideração os trabalhos levantados.

2.2.1 Categorização de metodologias e procedimentos para automatização de layouts

Uma categorização de procedimentos para a automatização de layouts é realizada por Du *et at.* (2020) por meio de uma revisão da literatura. Os autores englobam geração de espaços e de layouts de mobiliário, apresentando uma rica discussão para classificação de métodos. Du *et at.* (2020) trabalham com as seguintes classificações para geração de layouts automatizados: baseada em forças físicas; programação matemática; teoria dos grafos; atribuição de célula; divisão espacial; baseada em agentes e *machine learning*.

A abordagem de geração de layouts baseada em forças físicas (Figura 8) se refere a um sistema de atração e de repulsão entre elementos (DU *et al.,* 2020). As instâncias espaciais trabalhadas são chamadas de "nós". Elas são pontos no espaço que recebem os vetores das forças. Apesar da sua natureza adimensional, os nós geralmente são representados como círculos ou retângulos sobre um plano bidimensional. As relações espaciais são desenhadas como linhas que conectam estas geometrias. Para cada uma destas, por sua vez, um peso numérico é atribuído de modo que represente o grau de afinidade entre eles. Em uma análise com elementos espaciais, após o sistema encontrar um equilíbrio, os projetistas podem fazer um refinamento dos resultados, retirando espaços residuais ou sobrepostos (DU *et al.,* 2020). Arvin e House (2002), por exemplo, estudam como modelar objetivos topológicos (como adjacência e orientação) e geométricos (como área, proporção e alinhamento) por meio desta abordagem.

Figura 8 – Método de forças físicas. Da esquerda para direita: espaços e vetores, resolução topológica, resolução geométrica e layout final



Fonte: Arvin e House (2002) *apud* DU *et al.,* (2020, pág. 8)

No caso da programação matemática (Figura 9), os parâmetros relativos ao layout são explicitados em fórmulas matemáticas. A localização dos elementos é representada por coordenadas e as restrições espaciais, a fim de evitar sobreposição ou obstrução de passagem, são traduzidas em variáveis dentro de equações. Essa possibilidade de construção de funções matemáticas abre caminhos para a aplicação de algoritmos evolutivos na resolução de layouts (CALIXTO, 2016; DU *et al.,* 2020; MICHALEK; CHOUDHARY; PAPALAMBROS, 2002).

Figura 9 – Método de programação matemática. Da esquerda para direita: locação do espaço e dimensão e distância entre os espaços, fórmula para não sobreposição e layout gerado

$$R_1(x_1, y_1, w_1, h_1,);$$

$$d_x(R_1, R_2) = max\{V_x(R_1), V_x(R_2)\} - min\{V_x(R_1), V_x(R_2)\} - w(R_1) - w(R_2)$$

$$d_y(R_1, R_2) = max\{V_y(R_1), V_y(R_2)\} - min\{V_y(R_1), V_y(R_2)\} - h(R_1) - h(R_2)$$

$$f_2(\mathrm{I}) = \sum_{i=1}^{N_s-1} \sum_{j=i+1}^{N_s} f_{ov}(F_i, F_j) + \sum_{i=1}^{N_s} \sum_{j=1}^{N_a} f_{ov}(F_i, A_j)$$

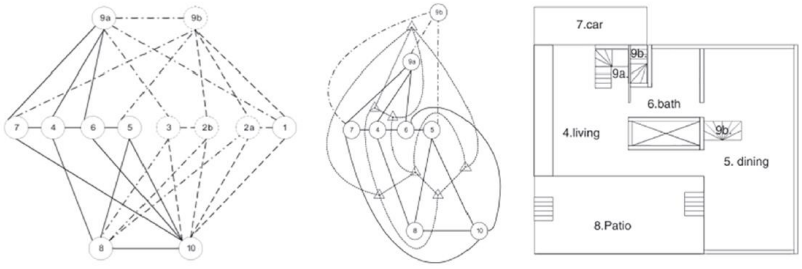
$$f_{ov}(\mathrm{R_1}, \mathrm{R_2}) = \omega(\mathrm{R_1} \cap \mathrm{R_2})\mathrm{h}(\mathrm{R_1} \cap \mathrm{R_2})$$

Fonte: Rodrigues, Gaspar e Gomes (2013a, 2013b) *apud* DU *et al.,* (2020, pág. 8)

Outra metodologia possível para a geração de layouts é a teoria dos grafos (Figura 10). As relações de afinidade entre os elementos são valoradas, interpretadas por algoritmos e convertidas em gráficos (BIAGINI; DONATO; PELLIS, 2015; FU *et al.*, 2017). Du *et al.* (2020) destacam o claro faseamento entre uma análise topológica e outra de atribuição de geometrias. O autor exemplifica esse conceito por meio do trabalho de Wong e Chan (2009) que, para um sistema generativo de layouts, criam inicialmente uma matriz de afinidade entre os espaços. Em seguida, essas relações são transformadas em um gráfico de nós e fios. Isso é interpretado por algoritmos capazes de espacializar estas relações e, por fim, são inseridas as geometrias.

Figura 10 - Diagramas relacionados à teoria dos grafos. Da esquerda para direita: matriz de proximidade, grafo plano, grafo para uma porta e layout gerado

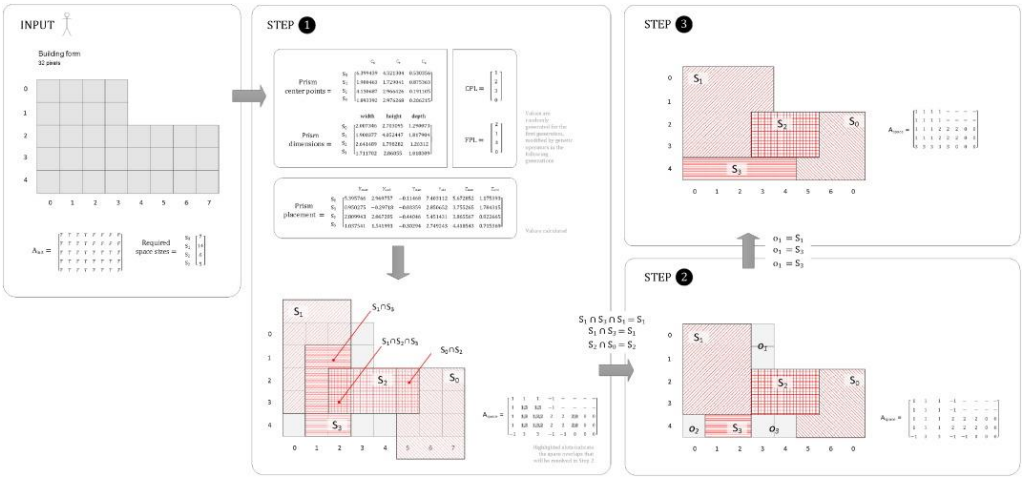
	8	2a	2b	7	5	4	6	3	1	9a	10	9b
8	-	3	3	0	1	3	0	1	0	0	3	0
2a		-	0	0	0	0	0	0	2	0	3	3
2b			-	0	0	0	0	2	0	0	3	3
7				-	0	1	0	0	0	3	3	3
5					-	0	2	0	0	0	2	3
4						-	2	0	0	3	0	0
6							-	0	0	3	1	0
3								-	0	3	3	0
1									-	3	3	3
9a										-	0	3
10											-	0
9b												-



Fonte: Wong e Chan (2009) *apud* DU *et al.,* (2020, pág. 8)

A atribuição de célula consiste na conversão de um espaço em uma malha bidimensional ou tridimensional com células do mesmo tamanho seguida da atribuição de diferentes funções ou elementos para estes espaços (DU *et al.,* 2020). Uma matriz é definida para representar as células do edifício e uma segunda para os espaços que devem ser atribuídos. Na sequência é feita a atribuição dos valores referentes a segunda matriz na primeira. Ao final, são realizadas operações de intersecção e união para subdivisão do espaço inicial (Figura 11). Essa metodologia pode ser vista nos trabalhos de Dino (2016) e Lopes *et al.* (2020).

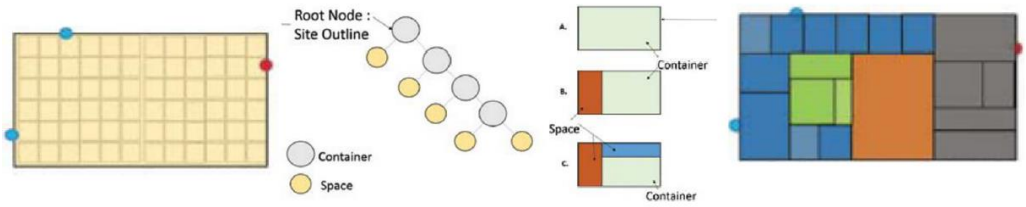
Figura 11 – Etapas de uma metodologia de atribuição de célula desenvolvido por Dino (2016)



Fonte: Dino (2016, p. 134)

O quinto método abordado por Du *et at.* (2020) se refere à divisão espacial (Figura 12). Neste caso, um espaço pré-determinado é dividido recursivamente segundo uma árvore de dados cujos nós se referem aos espaços divididos. Em primeiro lugar, um espaço é definido pelos usuários. Em seguida, as dimensões espaciais e suas relações de proximidades são acrescentadas na árvore. Depois o espaço é recortado e por fim o layout final é gerado. Este método é trabalhado, por exemplo, por DAS *et al.* (2020), Knecht e Koenig (2010), Correia, Duarte e Leitão (2012), Koenig e Knecht (2014) e Abdelmohsen, Assem e Tarabishy (2016).

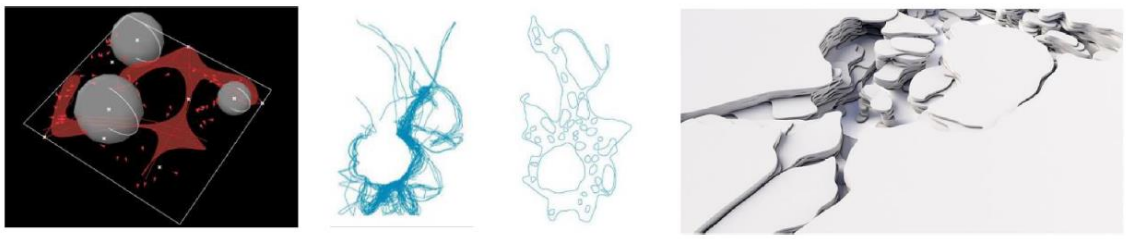
Figura 12 – Método de divisão espacial. Da esquerda para direita: layout pré-definido, árvore de dados, processo de divisão espacial e layout gerado



Fonte: DAS *et al.* (2020) *apud* DU *et al.,* (2020, pág. 9)

O layout espacial também pode ser gerado por agentes (Figura 13), como, por exemplo, com base em simulações dos caminhos realizados pelos usuários no espaço. Inicialmente, seus movimentos são simulados a partir de pontos de atração e de repulsão, obstáculos e acessos. Na sequência, os caminhos dos usuários são desenhados, formando uma malha. Por fim, os espaços restantes podem acomodar espaços funcionais (DU *et al.,* 2020). Ghaffarian, Fallah e Jacob (2018) trabalham este conceito para a geração de novos espaços, enquanto Chang (2018) utiliza este conceito para a modificação em tempo real do mobiliário segundo a ação dos usuários.

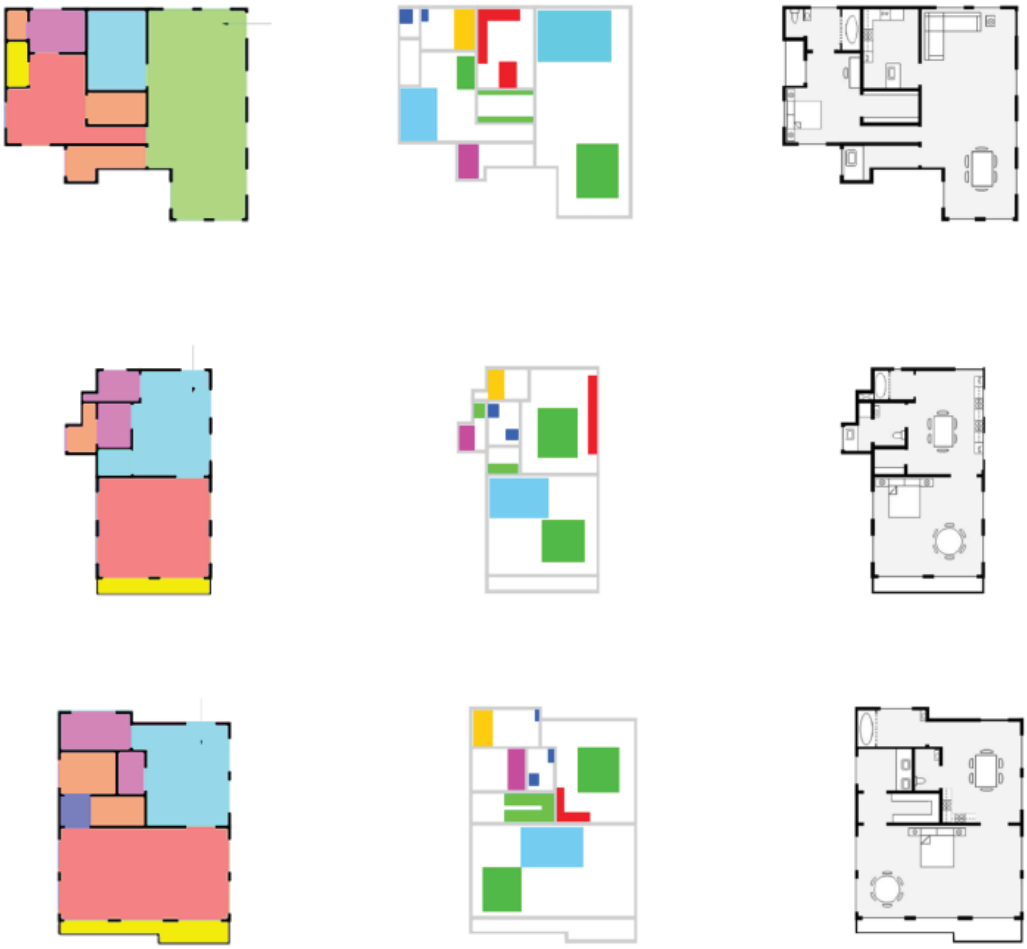
Figura 13 – Método baseado em agentes. Da esquerda para direita: simulação do movimento dos ocupantes, caminhos de circulação, espaços de circulação e espaço de layout gerado



Fonte: Ghaffarian, Fallah e Jacob (2018) *apud* DU *et al.,* (2020, pág. 9)

Por fim, o layout pode ser gerado por *machine learning* (Figura 14). Neste caso, algoritmos conseguem reconhecer padrões de um grande banco de dados de layouts já realizados anteriormente e assim sugerir uma solução para um novo caso sem a necessidade de uma descrição formal. Deste modo, esta abordagem procura imitar o processo de criação dos arquitetos (DU et al., 2020). Diversas pesquisas têm sido realizadas utilizando esta metodologia. Chaillou (2019), por exemplo, trabalha com *Generative Adversarial Networks* (GAN) para a partir de um banco de dados de imagens e, após analisá-las, gerar novos layouts. As, Pal e Basu (2020) analisam plantas de residências por meio de *deep learning* para a geração de novos edifícios.

Figura 14 – Método de *machine learning*. Geração por meio de Generative Adversarial Networks (GAN)



Fonte: Chaillou (2019, pág. 44)

Du *et al.* (2020) analisa sessenta e seis artigos e os classifica dentro dessas sete categorias. Em seguida, faz uma análise do processo de automatização de cada metodologia. O autor analisa (1) a viabilidade, considerando requisitos para uma aplicação prática; (2) a usabilidade, analisando se o processo é facilmente controlado pelo usuário; (3) a velocidade de geração, quão rápido se pode chegar a uma solução por este método; (4) a variação, quão fácil é gerar diferentes soluções; (5) a capacidade de trabalho com maior número de edifícios com mais de um pavimento; (6) a capacidade de trabalhar com espaços não retangulares e, por fim, (7) a necessidade da existência de limites pré-estabelecidos. Os parâmetros de variação, trabalho com diferentes níveis, capacidade de trabalhar com formas não retangulares e existência ou não de limites pré-definidos foram classificados como quantitativos. Assim, Du *et al.* (2020) classifica as metodologias segundo a força de cada uma dentro desses critérios com base nos sessenta e seis artigos coletados (Tabela 1).

Tabela 1 – Comparação entre os métodos de automatização de layouts. O símbolo "/" significa que a propriedade não pode ser comparada e o símbolo "+" refere-se à força de cada propriedade

Comparação entre os métodos de geração de layout

	Variação			Outros parâmetros		
	Mudança dos limites espaciais	Mudança de topologia	Mudança de geometria	Mais de um pavimento	Espaços não regulares	Limites pré-estabelecidos
Baseada em forças físicas	Sim	+++	+	+	+	Não
Programação matemática	Sim	+	+++	++	++	Não
Teoria dos grafos	Sim	+++	+	++	++	Não
Atribuição de célula	Não	++	++	+++	+++	Sim
Divisão espacial	Não	+++	+	++	++	Sim
Agentes	Não	/	/	++	+++	Sim
Machine learning	Sim	/	/	+++	+++	Não

Fonte: Adaptado de Du *et al.* (2020, p. 11)

Du *et al.* (2020) destacam que mudanças geométricas são mais fáceis de serem realizadas através de um modelo de programação matemática. Entretanto, os autores apontam que mais operações são necessárias quando se pretende realizar mudanças topológicas, como rotação, espelhamento e alterações de dimensões. O *machine learning*, por sua vez, se mostra como um método com alto potencial para gerar espaços com formas irregulares a partir da leitura de um grande conjunto de dados. Entretanto, afirmam também que pode existir a combinação entre métodos, como de programação matemática e de divisão, para solucionar ambientes com formas irregulares.

É possível inferir que a mudança da posição e de dimensões de mobiliários durante o processo de geração de layouts de mobiliários poderiam ser facilitadas por meio da metodologia de programação matemática. E, se associadas com outras metodologias, estas poderiam alcançar melhores resultados. Du *et al.* (2020) não pontuam a associação destas metodologias com o BIM. Entretanto, devido a sua natureza paramétrica e a facilidade de se acessar e modificar parâmetros, sua utilização poderia potencializar os processos dentro de cada uma delas. Deste modo, conforme será descrito na seção de desenvolvimento, esta dissertação combinou as estratégias de programação matemática e de divisão espacial para o desenvolvimento das implementações de *Space Planning* em BIM.

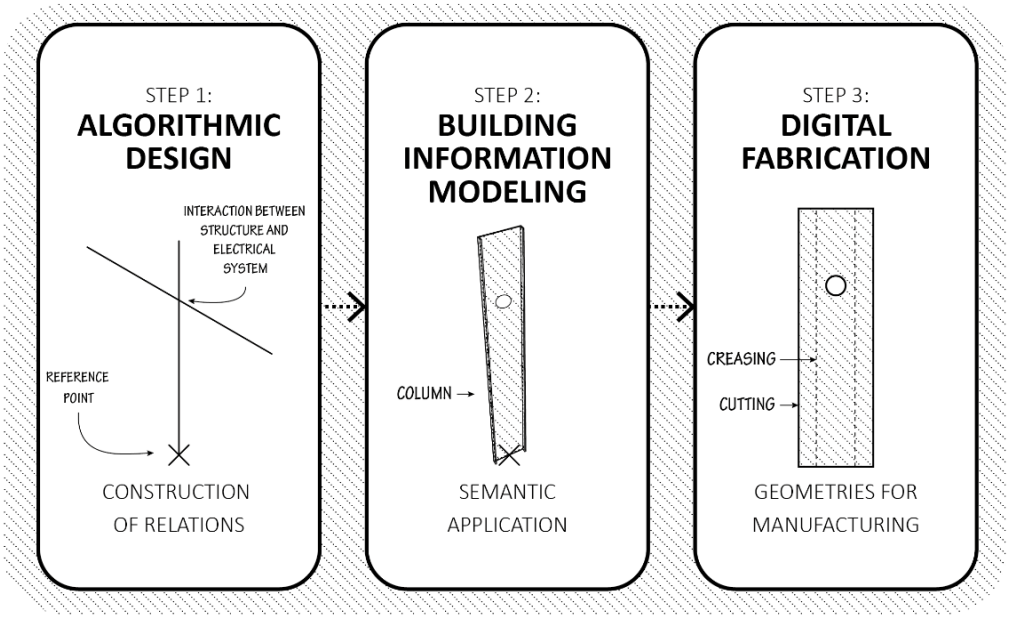
No capítulo seguinte, estudos de casos chaves de automatização de layouts de mobiliários são apresentados. Eles foram divididos em dois grupos, com o objetivo de facilitar a discussão sobre a classificação de problemas abordada no capítulo anterior. O primeiro deles se refere à utilização de algoritmos determinísticos e o segundo aos não determinísticos.

2.2.2 Algoritmos determinísticos

Um algoritmo pode ser definido como determinístico quando seu comportamento é determinado exclusivamente pelas entradas e pelo estado inicial (BUTTERFIELD; NGONDI; KERR, 2016). Consequentemente, os resultados podem ser obtidos através de métodos procedurais controlados. Muitos problemas podem ser resolvidos por algoritmos determinísticos e os benefícios da sua implementação se aplicam à maioria das questões cotidianas enfrentadas pelos arquitetos.

Em um trabalho anterior, desenvolvemos algoritmos em BIM para a automatização de layouts de banheiros pré-fabricados com o objetivo de contribuir para a personalização em massa do setor (CÔCO JÚNIOR; CELANI, 2018), aumentando sua produtividade. Um fluxo de trabalho foi desenvolvido heuristicamente, integrando processo algorítmico, BIM e fabricação digital, por meio de programação matemática. Inicialmente, três tipologias foram estabelecidas simulando o portfólio de uma indústria do ramo de pré-fabricação. Isso delimitou o problema como bem definido. Para cada uma das tipologias, regras de posicionamento relativas a objeto-objeto e a objeto-espaço foram estabelecidas para resolver conflitos entre sistemas e garantir a melhor configuração em diferentes casos. Em seguida, o código inseriu semântica construtiva nestes elementos, ou seja, geometrias básicas originadas de equações matemáticas receberam instanciações em BIM. Linhas foram transformadas em perfis de *light steel frame*, retângulos em placas de *drywall,* pisos cerâmicos e placas de forro. Por fim, esses mesmos elementos foram abstraídos em geometrias que poderiam ser cortadas em máquinas CNC (Figura 15). O trabalho demonstrou que algoritmos determinísticos podem trazer um grande benefício para automatizar trabalhos repetitivos dentro de um fluxo de trabalho entre Arquitetura e indústria.

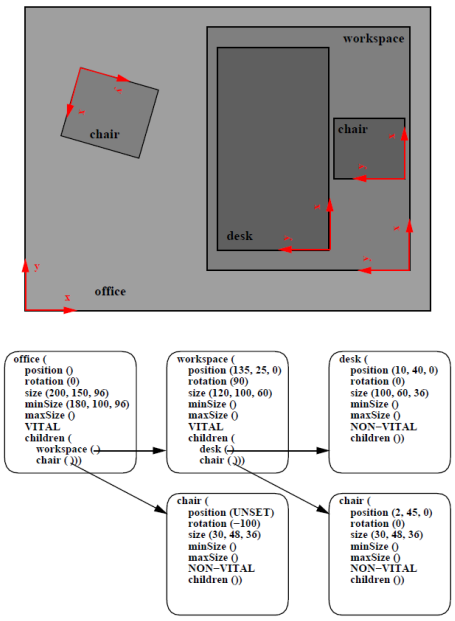
Figura 15 - *Workflow* desenvolvido que integra projeto algorítmico, BIM e fabricação digital



Fonte: adaptado de Côco Júnior e Celani (2018)

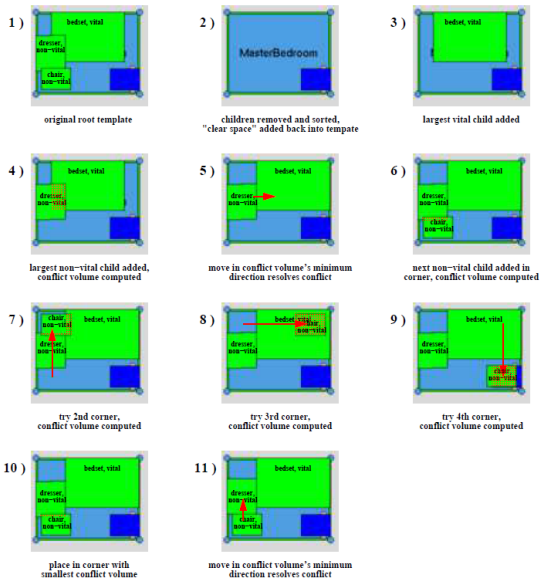
De modo semelhante, diversas outras pesquisas têm demonstrado aplicabilidade de algoritmos determinísticos na geração de layouts. Kjølaas (2000) implementa um sistema de geração de mobiliário para preencher simultaneamente diversos espaços com mobília de sala de estar e de jantar, duas tipologias de quartos e duas tipologias de escritórios. O sistema generativo tem como objetivo usar o layout como ferramenta para o estudo da funcionalidade espacial. Para isso, o autor simplifica o mobiliário e usa retângulos para demarcar o espaço em frente às portas, atribuindo-lhes uma hierarquia. Inicialmente, um recorte de pesquisa foi realizado para ambientes retangulares e, separadamente, seis *templates* com arranjos de mobiliário foram criados para diferentes funções, como "sentar" e "trabalhar" (Figura 16). Cada um deles poderia ser inserido no espaço em relação ao sistema de coordenadas (XYZ) do ambiente, segundo parâmetros de largura, profundidade, altura e rotação. Assim, dependendo da conformação do espaço, os *templates* poderiam ser escalados, rotacionados ou espelhados de modo que um conjunto pudesse gerar oito variações para melhor se ajustarem à Arquitetura. Assim, por meio de uma metodologia de programação matemática, Kjølaas (2000) controla as transformações geométricas e topológicas dos ambientes.

Figura 16 - Hierarquia entre *templates* e mobiliário e suas variáveis



Fonte: Kjølaas (2000 p.19)

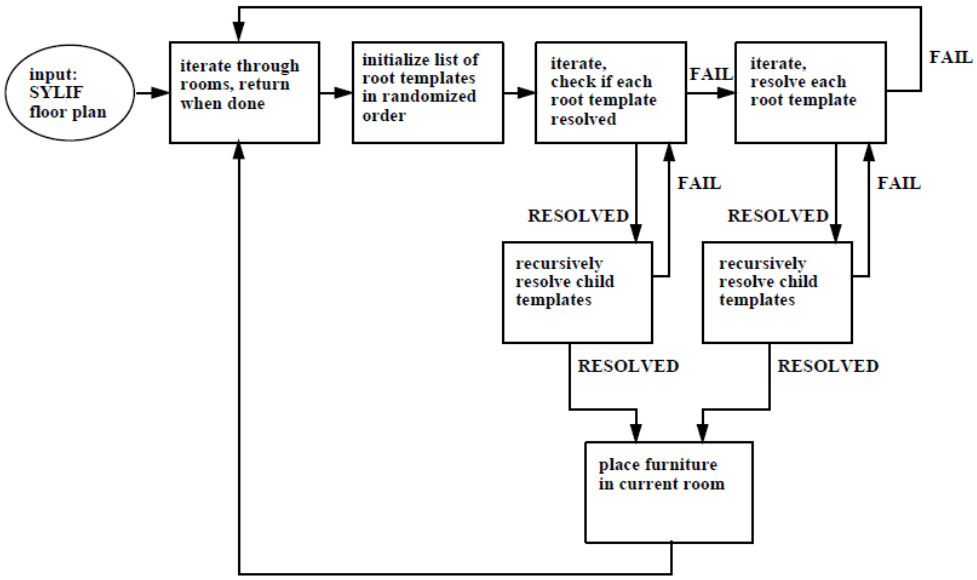
Figura 17 - Sistema recursivo para resolução de conflitos e colocação de mobiliário. Os retângulos em azul marcam as aberturas



Fonte: Kjølaas (2000 p.26)

Esses *templates* possuíam "*templates* filhos" com o mobiliário, de modo que, dependendo das dimensões dos conjuntos principais, esses seriam rearranjados. Kjølaas (2000), assim, adotou um sistema recursivo para a inserção do layout, permitindo uma liberdade para deslocamento dos elementos após sua inserção, seguindo uma hierarquia de prioridade (Figura 17). Por exemplo, no caso da geração do layout de um dormitório, a cama seria o primeiro elemento inserido, porque é essencial dentro deste ambiente. Em seguida, um guarda-roupa seria posicionado. Caso houvesse um conflito entre os dois, a cama seria deslocada para uma outra posição até que se encontrasse uma configuração espacial aceitável. Caso contrário, outra peça do mobiliário seria movida e assim por diante (Figura 18). Um dos resultados possíveis pode ser visto na Figura 19.

Figura 18 – *Workflow* para geração de layout



Fonte: Kjølaas (2000 p.32)

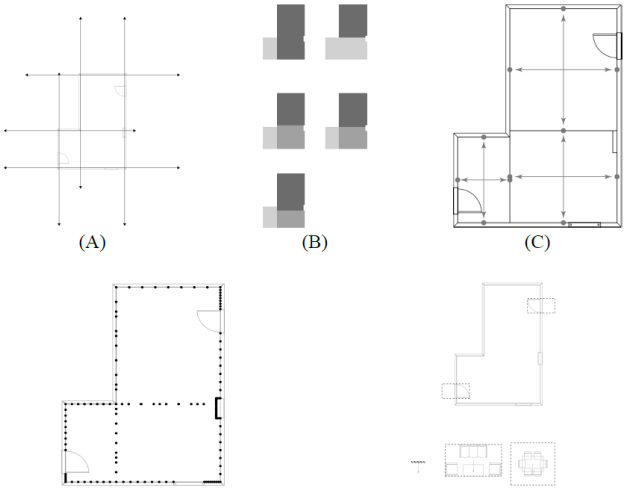
Kjølaas (2000) aponta que novos trabalhos poderiam ser feitos a partir do que foi desenvolvido, como a ampliação da metodologia para espaços de escritórios ou prédios públicos. O autor conclui que a automatização da geração de layouts é uma poderosa ferramenta, quando necessário fazer projetos de interiores para muitos ambientes. Ao mesmo tempo, destaca que essa metodologia pode favorecer a diversidade de soluções, ao contrário de layouts sempre iguais e monótonos. Mesmo assim, o trabalho de Kjølaas (2000) se limita a ambientes retangulares e produz um layout bem esquemático. Nos resultados apresentados, inclusive, nota-se algumas soluções que não são adequadas. Na Figura 19, por exemplo, nota-se em um dos quartos a cabeceira da cama na mesma orientação que a porta ou, no escritório dos estudantes, a criação de espaços residuais. Além disso, todos os *templates* desenvolvidos devem ser pré-determinados, consequentemente há uma baixa capacidade interativa entre o usuário e o sistema.

Figura 19 – Um dos resultados gerados por Kjølaas (2000)

Fonte: Kjølaas (2000, p. 51)

De maneira semelhante, Abdelmohsen, Assem e Tarabishy (2016) desenvolvem um algoritmo heurístico com base em regras para a automatização de layouts para espaços residenciais por meio de *templates*. Além disso, os autores associam uma metodologia de programação matemática a um método de divisão de espaços, e assim conseguem trabalhar com espaços não retangulares (Figura 20).

Figura 20 – Metodologia para subdivisão de espaços não retangulares: (A) desenho das paredes virtuais, (B) identificação dos espaços filhos, (C) identificação dos pontos centrais e dos pontos nas bordas, (D) identificação das novas bordas dos segmentos de parede e (E) identificação da abertura e criação de um *bounding box,* aplicação de regras de circulação para *templates* para uma sala de jantar, estar e TV

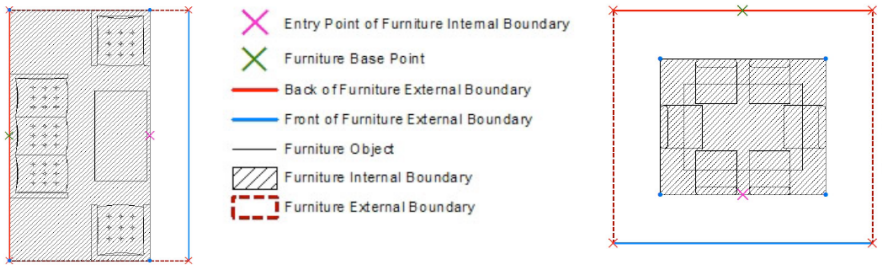


Fonte: Abdelmohsen *et al.* (2016, p. 497)

O algoritmo foi iniciado pela identificação das bordas de um cômodo em "L", das suas aberturas e dos pilares existentes. Em seguida, paredes virtuais foram feitas para a

criação de subzonas retangulares com o objetivo de reduzir espaços com formas complexas em casos mais simples. Todas as divisões possíveis foram exploradas e aquelas com melhores proporções entre largura e comprimento foram selecionadas. Na sequência, grupos de mobiliário foram estabelecidos (Figura 21), selecionados e inseridos nos novos espaços, também relativos às suas áreas e proporções. É importante destacar que os autores procuraram emular o processo de projeto cognitivo, assim não geraram nem testaram todas as possibilidades de posicionamento dos layouts, apenas aquelas que heuristicamente seriam viáveis. Os pontos ótimos de inserção dentro dos limites dos espaços subdivididos foram determinados visando uma boa circulação e melhor acomodação do *template*.

Figura 21 – Grupos de layouts e variáveis trabalhadas com o objetivo de inseri-los na melhor subzona



Fonte: Abdelmohsen *et al.* (2016, p. 495)

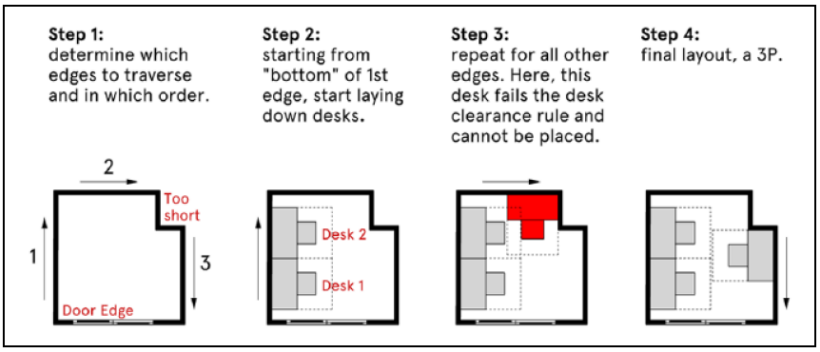
Deste modo, Abdelmohsen, Assem e Tarabishy (2016) desenvolvem um algoritmo que insere um layout para uma sala de estar e sala de jantar em ambientes não retangulares. Os resultados podem ser vistos na Figura 22. Os autores apontam que novos trabalhos podem ser realizados no sentido de deduzir regras implícitas a partir dos layouts gerados. Outra possibilidade de atuação seria aumentar a complexidade do contexto, seja pela forma do cômodo, das funções inseridas, dos *inputs* (como visibilidade, acesso, aspectos ambientais) ou de geometria com modelos tridimensionais.

Figura 22 – Resultados obtidos por Abdelmohsen *et al.* (2016)

Fonte: Abdelmohsen *et al.* (2016, p. 498)

Já Anderson *et al.* (2018) desenvolvem um sistema de geração automatizada de layouts para salas privadas de trabalho da WeWork. Os autores desenvolveram três algoritmos: (1) Algoritmo de Rotação de Layout, (2) Algoritmo de Esquerda Direita e (3) Algoritmo de Força Bruta. O primeiro deles (Figura 23) passa por todas as bordas do ambiente; caso uma das arestas seja menor que a largura de uma mesa, ela é ignorada. Caso contrário, devem ser inseridas quantas mesas couberem na aresta a partir de uma das extremidades. O algoritmo é executado em sentido horário, em sentido anti-horário e por fim da linha mais longa para a mais curta.

Figura 23 – Algoritmo de Rotação de Layout

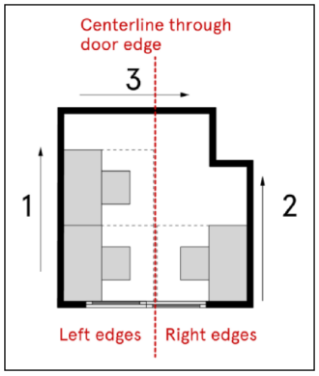


Fonte: Anderson *et al.* (2018, p. 167)

O algoritmo Esquerda Direita (Figura 24) é bem parecido com o anterior. Entretanto, possui a peculiaridade de criar um eixo de simetria perpendicularmente em relação à porta de entrada. Além disso, o layout sempre é posto de baixo para cima, proporcionando maior simetria nas soluções. Neste caso, também são executados alguns "relaxamentos" na

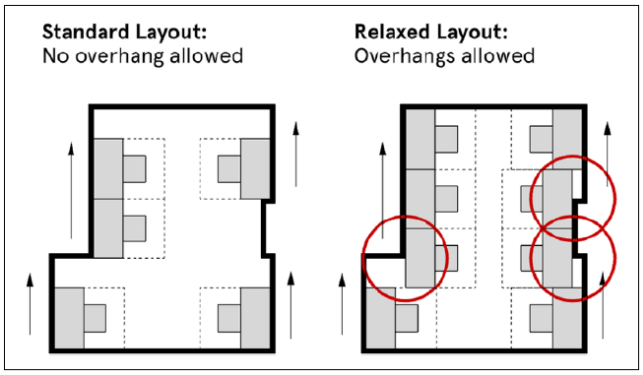
colocação (Figura 25), isto é, se existir alguma saliência na borda, uma mesa pode ser posta mesmo que fique um pouco fora da parede.

Figura 24 – Algoritmo Esquerda Direita



Fonte: Anderson *et al.* (2018, p. 169)

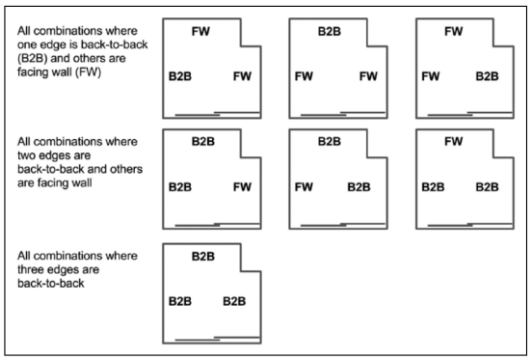
Figura 25 – Algoritmo padrão e com relaxamento



Fonte: Anderson *et al.* (2018, p. 170)

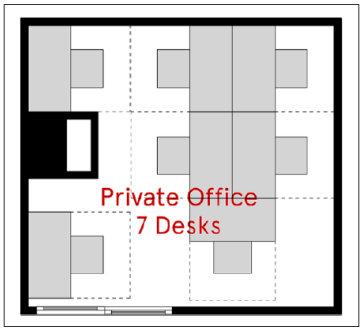
O layout de Força Bruta é executado somente se os dois primeiros não alcançarem uma solução, devido a uma demanda maior de uso computacional. Neste caso, o algoritmo arranja duas combinações de mesas, uma com uma mesa faceando a parede (FW) ou uma de frente para a outra (B2B). Em seguida, testa as combinações possíveis com B2B em uma parede, em duas ou em três (Figura 26). O layout é então gerado em sentido horário ou anti-horário, permitindo, inclusive, uma ocupação da área central da sala e uma mesa rotacionada em 90º no final do conjunto (Figura 27).

Figura 26 – Algoritmo de Força Bruta



Fonte: Anderson *et al.* (2018, p. 170)

Figura 27 – Maximização da colocação das mesas com o Algoritmo de Força Bruta



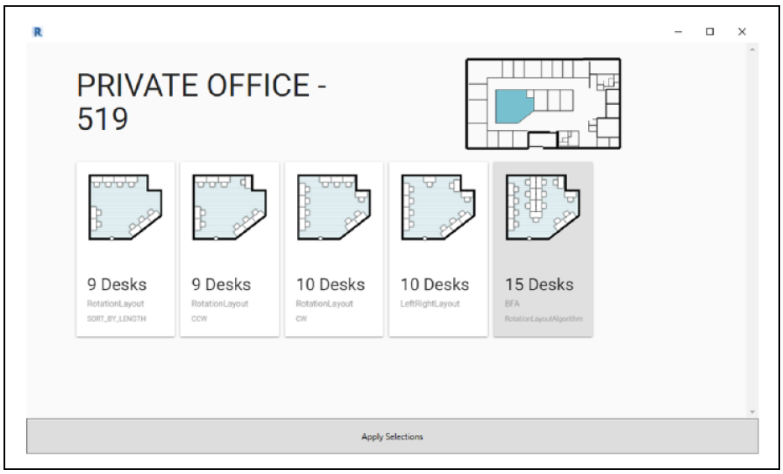
Fonte: Anderson *et al.* (2018, p. 171)

O sistema foi implementado em Python utilizando a biblioteca de geometrias Shapely e hospedado em um servidor com o REST API servida pela Flask web framework. Isso permitiu maior flexibilidade para futuras aplicações. O primeiro protótipo foi implementado como um *Add-in* no Autodesk Revit (Figura 28), por este ser o programa BIM comumente

utilizado na WeWork. Neste protótipo, o desenvolvimento ocorre pelo seguinte procedimento:

- O usuário seleciona um ou mais ambientes;
- O usuário escolhe entre um sistema automatizado padrão ou avançado, neste último é habilitado comandos como dimensões das mesas e tolerâncias;
- Na sequência, é gerado uma visualização prévia com alternativas de layouts para cada um dos ambientes selecionados;
- O usuário escolhe uma delas e confirma;
- Neste momento, o layout escolhido é traduzido para a linguagem do Revit, isto é, mesas e cadeiras são substituídas por famílias específicas;
- Por último, o layout pode ser ajustado manualmente dentro do Revit.

Figura 28 – Interface para geração de layout



Fonte: Anderson *et al.* (2018, p. 175)

Os algoritmos desenvolvidos foram comparados com 13000 layouts de salas comerciais desenvolvidos manualmente por arquitetos. Os autores descrevem que em 77% dos casos o algoritmo teve um desempenho igual àqueles feitos por arquitetos, sendo que se houvesse um relaxamento na interpretação este número poderia chegar a 97%, e em 6% o desempenho foi superior. Anderson *et al.* (2018) apresentam algumas diretrizes para a interface em oposição a um sistema totalmente automatizado, mas que favoreça a produtividade e um ambiente flexível. São elas:

- Apresentar mais de uma opção para o usuário;

- Ser transparente em relação ao algoritmo e, se possível, apresentar uma documentação acessível do código;
- Permitir substituições manuais de parâmetros de entrada e de saída;
- Deixar a tomada de decisões definitivas para os projetistas, enquanto aumentam seus recursos com uma iteração mais rápida.

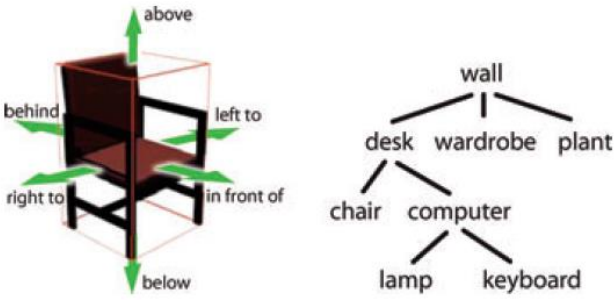
O trabalho de Anderson *et al.* (2018) traz uma importante contribuição tanto pela metodologia incremental desenvolvida como pelos resultados obtidos. Os autores demonstram que é possível sistematizar tarefas repetitivas como meio para potencializar o processo de projeto dos arquitetos. Ao mesmo tempo, demonstram que um algoritmo de geração de layouts pode alcançar resultados iguais ou melhores que os produzidos manualmente por arquitetos.

2.2.3 Algoritmos não determinísticos

Algoritmos não determinísticos ocorrem quando em determinados pontos do código há uma escolha por qual caminho seguir, podendo ser arbitrária ou não (BUTTERFIELD; NGONDI; KERR, 2016). Assim, para os mesmos valores de entrada pode-se obter mais de um resultado plausível. A seguir, são apresentados estudos de caso que utilizam os métodos probabilístico, baseado em agentes, otimização estocástica, algoritmos evolutivos e generativos e *machine learning* para a resolução de layouts.

Germer e Schwarz (2009) criam um sistema procedural de geração de layouts por meio de agentes com o objetivo de preencher uma grande quantidade de espaços com baixo custo de processamento. Os autores se utilizam dessa abordagem para criar ambiências em tempo real para visualizações urbanas integradas com interiores de edifícios para jogos. Como explicado anteriormente, agentes são entidades autônomas dentro de um ambiente artificial que atuam segundo regras pré-estabelecidas. Neste caso, os autores atribuíram essa função aos mobiliários e aos objetos de decoração (Figura 29; Figura 30).

Figura 29 – Mobiliários e objetos como agentes, suas orientações e relações de parentesco



Fonte: Germer e Schwarz (2009, p. 2070)

Inicialmente, foram criadas *bounding boxes* orientadas, representando os elementos a serem inseridos. Em seguida, uma relação de parentesco foi atribuída entre os elementos, por exemplo: uma cadeira deveria estar associada com uma mesa, um teclado a um computador, um computador a uma mesa e uma mesa a uma parede. Assim cada um dos agentes segue três ações: buscar, organizar e descansar. Na primeira delas, cada agente analisa possíveis "pais" com que pode se associar, examinando, inclusive, a posição dos outros agentes. Se ele for bem-sucedido no encontro, passará para a próxima fase, caso contrário, será deletado. Na etapa de organização, os agentes são posicionados e alinhados em relação aos pais e é verificado se houve algum conflito com outro agente. Se existir, ele tentará achar aleatoriamente outra posição viável. Se mesmo assim não encontrar, procurará outro "pai" ao qual possa se associar. Por fim, a fase de descanso ocorre quando o posicionamento do agente é concluído. Assim, o agente passará a se mover em relação à geometria pai e, caso essa geometria seja retirada do sistema, o agente voltará ao modo de busca.

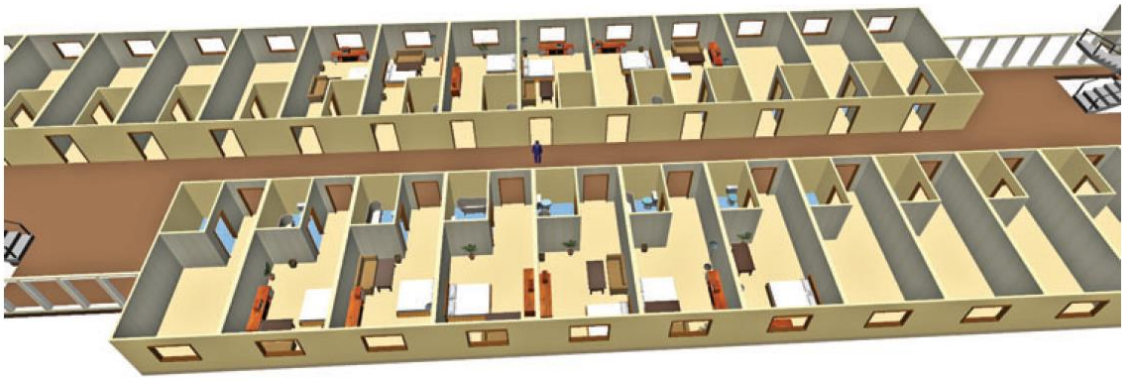
Figura 30 – Exemplos de layouts gerados para um escritório e unidade de um hotel

Fonte: Germer e Schwarz (2009, p. 2073)

Deste modo, antes de executar o programa, o usuário deve atribuir manualmente semântica aos elementos. Atribui-se *tags* para os cômodos segundo suas funções, bem como

para as suas faces, determinando onde estão as portas, vãos, janelas e paredes (porque os elementos construtivos podem receber agentes também). Insere-se informações de quantos elementos do mobiliário serão inseridos e qual o seu estilo. Por último, cada objeto recebe informações sobre a quais pais podem se associar e como isso pode acontecer, espaços livres e espaços laterais, uma opção para inserção em malhas e uma opção de como se orientar em relação ao pai. Germer e Schwarz (2009) destacam que as mesmas informações utilizadas para um único cômodo poderiam ser aplicadas a muitos outros automaticamente. Os autores descrevem que o algoritmo criado conseguiu inserir aproximadamente trezentos objetos em menos de quatro segundos. Isso permitiu que, conforme o usuário andasse por um edifício virtual, o layout fosse criado em tempo real por onde ele passava, evitando-se, portanto, gastos desnecessários com o processamento computacional (Figura 31).

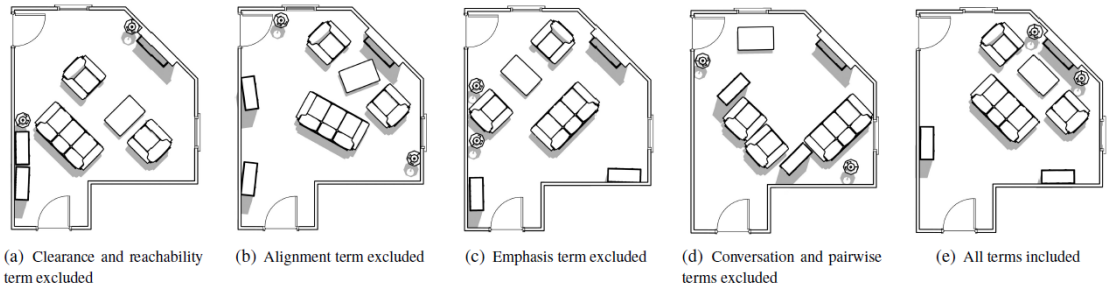
Figura 31 – Layout gerado para um hotel em relação ao movimento do usuário no espaço



Fonte: Germer e Schwarz (2009, p. 2074)

Merrell *et al.* (2011) desenvolvem um sistema que automatiza a geração de layouts de mobiliário para residência a partir de diretrizes funcionais e visuais. No primeiro grupo estão contidas funções para a liberação, isto é, os afastamentos em relação a outros elementos; circulação para permitir que todo mobiliário seja acessado; relação de pares, ou seja, a relação entre um elemento e outro e, por fim, conversação, parâmetro que diz respeito a distância entre duas pessoas para que se escutem bem durante um diálogo. No segundo grupo, por sua vez, estão os parâmetros de equilíbrio, onde os elementos se distribuem harmonicamente no ambiente; alinhamento relativo a outros elementos e às paredes do cômodo e, por último, há a ênfase, isto é, o estabelecimento de um ponto focal dominante no ambiente. Cada uma destas propriedades compõe uma função-custo que trabalha dentro da cadeia de Markov de Monte Carlo. O peso de cada um do uso pode ser visto na Figura 32 e juntas integram uma função de densidade explorada pelo algoritmo probabilístico Metropolis-Hastings.

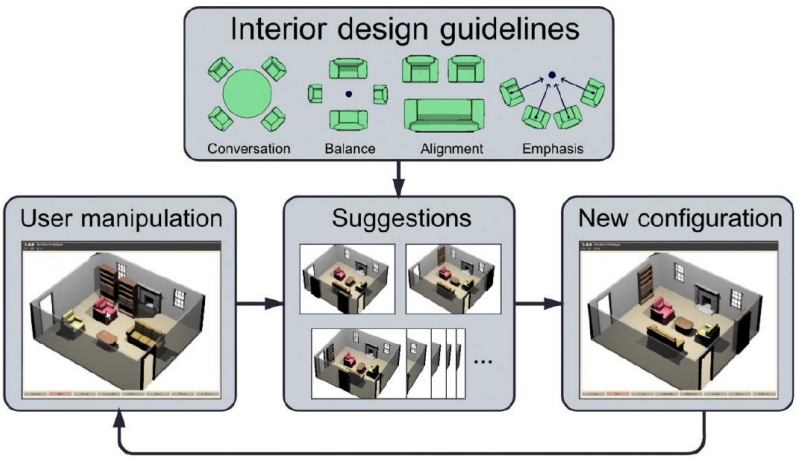
Figura 32 – Peso de cada uso na função de densidade



Fonte: Merrell *et al.* (2011, s.n.)

O sistema desenvolvido favorece a interação entre usuários e algoritmos por meio de uma interface interativa. Inicialmente, o usuário desenha um cômodo em três dimensões e seleciona os mobiliários que devem ser inseridos a partir de uma biblioteca pré-estabelecida. Em seguida, alternativas de arranjos de layouts são geradas a partir de diretrizes segundo uma função de densidade. Nesta etapa o usuário pode alterar a posição dos equipamentos e novamente solicitar uma nova sugestão de composição (Figura 33).

Figura 33 – *Workflow* do sistema desenvolvido por Merrell *et al.* (2011)

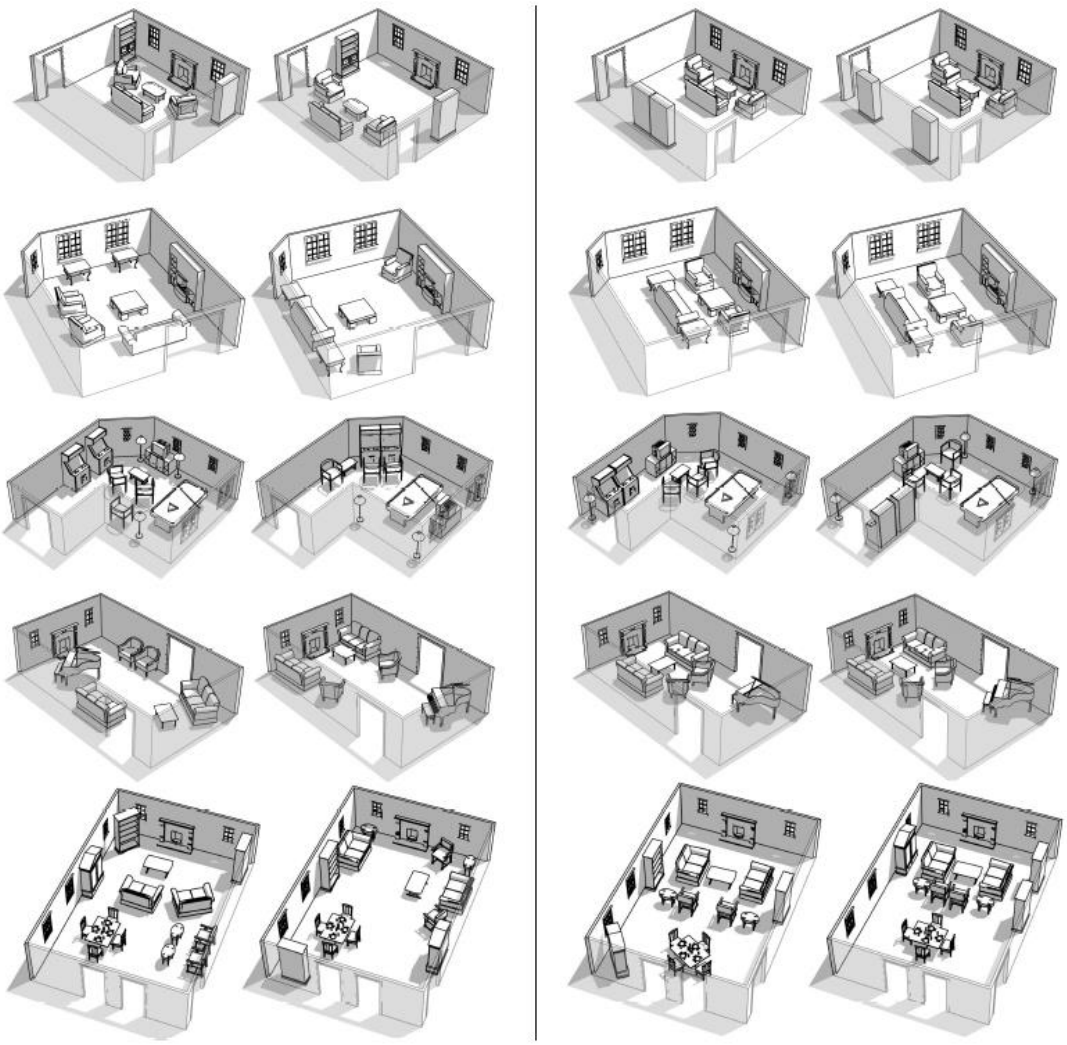


Fonte: Merrell *et al.* (2011, s.n.)

Um workshop foi realizado com o objetivo de testar a efetividade do algoritmo desenvolvido. Os dezoito participantes integravam o departamento de Ciência da Computação e não tinham experiência anterior com o desenvolvimento de layouts de Arquitetura. Inicialmente, os participantes receberam uma instrução sobre a interface e foi solicitado que gerassem o layout de duas salas de estar, uma sala de jogos, uma sala de piano, uma sala de estar aberta e uma sala de jantar. Apesar de usarem o ambiente virtual, não utilizaram de processos automatizados. Merrell *et al.* (2011) descrevem que, enquanto nem

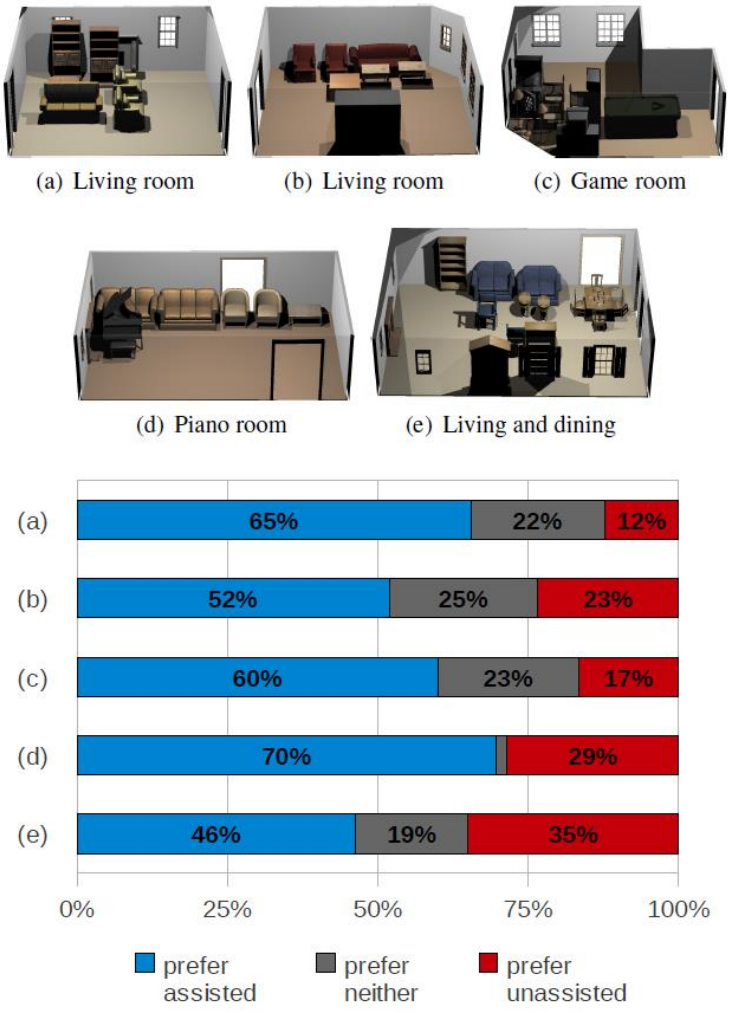
todos os participantes conseguiram concluir todos os layouts, o sistema desenvolvido gerou todas as soluções com um tempo médio de 0,734 segundos cada. Na sequência, os usuários repetiram o experimento utilizando o sistema criado (Figura 34). Os resultados foram avaliados por profissionais de projeto de interiores, demonstrando uma preferência pelos layouts produzidos que tiveram a assistência dos algoritmos (Figura 35). Apesar dos bons resultados, os autores ressalvam que sua intenção não era a de substituir o trabalho dos projetistas. Eles reconhecem a complexidade e a subjetividade de um projeto de layout de interior, mas acreditam que sistemas inteligentes podem auxiliá-los no processo de criação.

Figura 34 - Resultados obtidos no experimento com leigos em Arquitetura de interiores; à esquerda sem o auxílio do sistema de automatização e à direita com o uso do sistema criado



Fonte: Merrell *et al.* (2011, s.n.)

Figura 35 – Comparação entre os resultados gerados com e sem auxílio do sistema de automatização



Fonte: Fonte: Merrell *et al.* (2011, s.n.)

Uma abordagem similar foi realizada por Yu *et al.* (2011) para a criação de cenas para jogos. Relações espaciais, de hierarquia e de pares foram extraídas e utilizadas para a otimização do arranjo de mobiliários por uma função custo. Essa mesma função associada a métodos probabilísticos estocásticos também foi utilizada por Kán e Kaufmann (2018). Neste caso, os autores visam gerar o layout de mobiliário e objetos para um cômodo levando em conta parâmetros de estética, ergonomia e regras funcionais com o menor custo computacional possível. Mesmo que os autores não se aprofundem nestes conceitos, aplicam funções custo que utilizam esses parâmetros. Assim, levam em consideração os espaços de circulação de modo que os objetos não obstruam a passagem, as relações de "parentesco" entre os elementos, seus alinhamentos, distribuição e ritmo, visibilidade e proporção. Além disso, os autores aplicaram uma função que considera a proporção áurea para a distribuição dos objetos no espaço.

Para cada umas das categorias de objeto foram separados os parâmetros de tolerância de espaçamento para cada uma das faces, probabilidade de ficar encostando em uma parede, possibilidade de receber outros objetos filhos, probabilidade de ser um pai, a importância de o elemento estar naquele ambiente e, por fim, um número mínimo e máximo de inclusão de elementos da mesma categoria naquele espaço. A cada iteração de otimização é realizada uma sequência de dez ações: (1) mudar aleatoriamente a posição do objeto, (2) mudar aleatoriamente sua orientação, (3) alinhá-lo com o objeto mais próximo, (4) alinhá-lo com a parede mais próxima, (5) fixá-lo em uma parede mais próxima, (6) fixá-lo no objeto mais próximo, (7) conectá-lo a um dos possíveis pais, (8) adicionar novos objetos filhos aos pais, (9) adicionar novos objetos no sistema e (10) remover objetos aleatórios do layout. O algoritmo foi aplicado para layouts de dormitório, cozinha, sala de estar e escritório. Os resultados podem ser vistos na Figura 36.

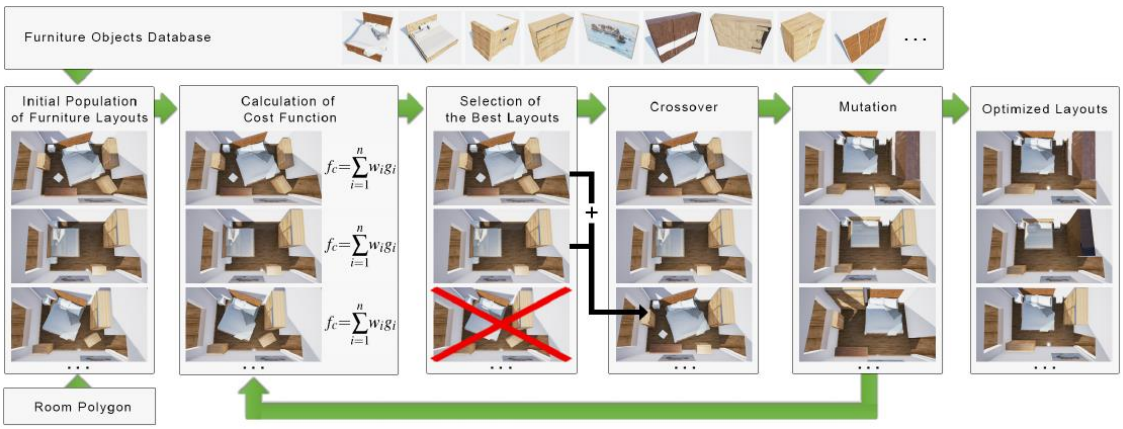
Figura 36 – Layouts gerados por Kán e Kaufmann (2018). Na esquerda estão os ambientes produzidos sem decoração e na direita com objetos

Fonte: Kán e Kaufmann (2017, p. 498)

Os mesmos autores utilizaram em outro trabalho um sistema com algoritmos genéticos para a elaboração de layouts (KÁN; KAUFMANN, 2017) em relação aos parâmetros de estética, ergonomia e funcionalidade, inclusive com a aplicação de materiais nos elementos em busca de uma uniformidade na linguagem. De modo semelhante aos casos anteriores, os autores trabalham com uma função custo, mas aplicam algoritmos genéticos para a otimização. Essa metodologia também foi adotada por Akase e Okada (2013), Sanchez, Roux e Gaildrat (2003), mas Kán e Kaufmann (2017) trabalham com maior grau de automatização. Os autores procuram simular o processo evolucionário natural (Figura 37). Assim, primeiramente, as instâncias são selecionadas aleatoriamente dentro de uma

biblioteca. A probabilidade de seleção é proporcional à importância do elemento naquele ambiente. Em seguida, a posição e a rotação de cada elemento são atribuídas aleatoriamente. A partir de então, o sistema passa por etapas iterativas de otimização por uma função custo que procura selecionar os melhores indivíduos, criar indivíduos por *crossover* e alterá-los por mutação (similarmente ao processo descrito anteriormente em Kán e Kaufmann (2018). É importante destacar que, sempre que houvesse a sobreposição de elementos, essa opção seria descartada. Kán e Kaufmann (2017) aplicam um modelo de ilhas de algoritmos genéticos, isto é, o sistema separa toda a população em subconjuntos (ilhas) nos quais aplica processos iterativos independentes e os melhores indivíduos de cada ilha podem migrar para outras. No caso descrito, os autores utilizaram quatro ilhas com cinquenta indivíduos em cada uma delas. A metodologia utilizada pode ser vista na Figura 37 e os resultados na Figura 38.

Figura 37 – Metodologia utilizada por Kán e Kaufmann (2017) para a geração de layouts por meio de algoritmos genéticos



Fonte: Kán e Kaufmann (2017, s. n.)

Figura 38 - Layouts gerados por algoritmos genéticos (à esquerda sem otimização de materiais e à direita com otimização de materiais)

Fonte: Kán e Kaufmann (2017, s. n.)

O algoritmo criado foi testado por meio de um experimento, a fim de verificar se existiria uma diferença significativa entre projetos criados manualmente por profissionais da área e pelo sistema. Primeiramente, dois profissionais de projeto de interiores foram

convidados a criar propostas para ambientes de cozinha, sala de estar e dormitório que também viriam a ser utilizados como base para o sistema automatizado. Para cada um dos ambientes houve duas possibilidades de aplicação de materiais, uma otimizada e outra não. Assim, ao final foram determinadas seis opções para avaliação. Os produtos foram avaliados por meio de questionários respondidos por trinta participantes, sendo 23 homens e 7 mulheres, na idade entre 23 e 49 anos. Os resultados demonstram que em alguns casos o processo algoritmo superou o manual (Tabela 2).

Tabela 2 – Preferência por layouts gerados por algoritmos ou manualmente

	Layout criado por algoritmos	Layout criado manualmente
Cozinha	21	9
Cozinha + Otimização de materiais	14	16
Sala de estar	8	22
Sala de estar + Otimização de materiais	16	14
Dormitório	3	27
Dormitório + Otimização de materiais	8	22

Fonte: adaptado de Kán e Kaufmann (2017)

Em uma publicação subsequente e por meio de um novo experimento, Kán e Kaufmann (2018) discutem a metodologia trabalhada anteriormente (algoritmos genéticos) com a de minimização de custo (*greedy cost minimization*). Para tanto, Kán e Kaufmann (2018) realizaram um experimento similar ao descrito anteriormente (KÁN; KAUFMANN, 2017). Os quarenta e quatro participantes deveriam escolher a preferência entre layouts gerados por uma ou outra metodologia para dormitórios, cozinhas, salas de estar e escritórios. Além disso, deveriam decidir entre as opções com ou sem decoração, sendo esta possibilidade disponível apenas com a minimização de custo. Os resultados demonstram um desempenho superior deste último algoritmo em seis dos oito casos (Tabela 3). Além disso, todos os resultados com decoração apresentaram resultados superiores para os layouts criados pelo algoritmo de minimização de custo.

Tabela 3 - Preferência por layouts gerados por algoritmos genéticos ou por algoritmo de minimização de custo (*greedy cost minimization*)

	Layout criado por algoritmos genéticos	Layout criado por algoritmos de minimização de custo
Dormitório	19	25
Dormitório + Decoração	13	31
Cozinha	31	13
Cozinha + Decoração	20	24
Sala de estar	25	19
Sala de estar + Decoração	10	34

Escritório	12	32
Escritório + Decoração	6	38

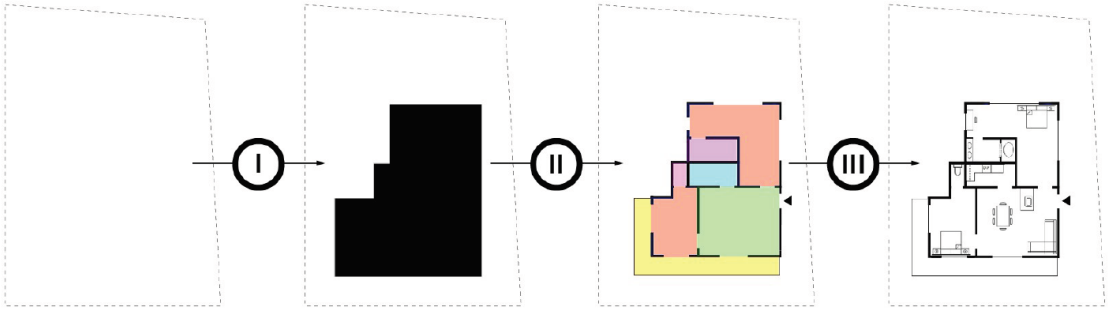
Fonte: adaptado de Kán e Kaufmann (2018)

Os trabalhos de Kán e Kaufmann (2017) e Merrell *et al.* (2011) demonstram como determinados processos de automatização de layouts de mobiliário podem ocorrer de modo satisfatório por meio da criação de funções e utilização em modelos probabilísticos.

Mais recentemente, entretanto, com o aprofundamento das capacidades computacionais e do aprimoramento das técnicas de Inteligência Artificial, novas abordagens têm sido aplicadas por meio de *machine learning*. Belém, Santos e Leitão (2019) apresentam uma lista exaustiva das estratégias de *machine learning* e como elas podem ser aplicadas na Arquitetura em diferentes circunstâncias. Os autores descrevem sua utilização em fases de definição conceitual e exploração do projetista, na fase de instanciação destes conceitos, na modelagem tridimensional e na otimização para melhor desempenho. Entre as estratégias utilizadas está a Rede Neural Generativa Adversária (GAN) aplicada por exemplo por Chaillou (2019).

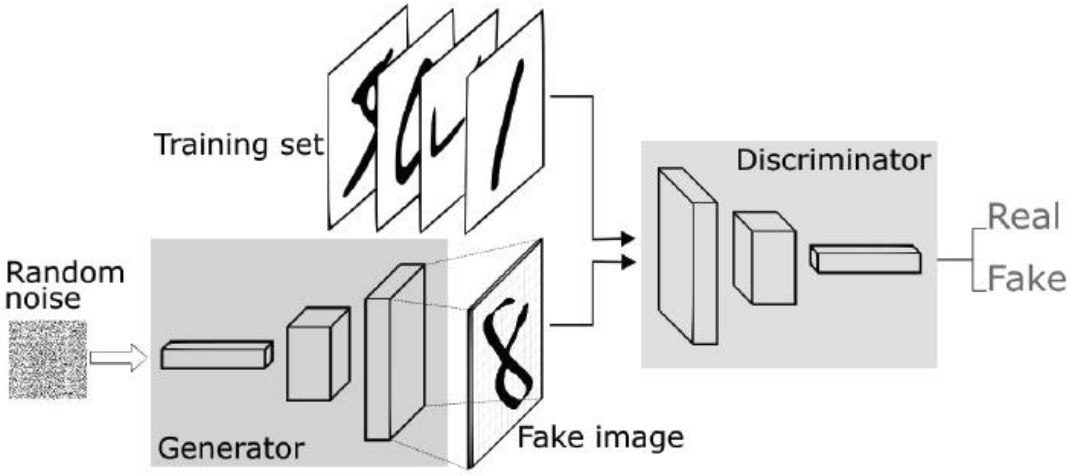
Chaillou (2019) (1) analisa e cria o contorno de plantas de apartamentos, (2) propõe uma divisão de cômodos e (3) prescreve os layouts (Figura 39) por meio de *machine learning*. Uma das estratégias centrais do desenvolvimento foi a utilização de Redes Neurais Generativas Adversárias, metodologia trabalhada por HUANG e ZHENG (2018) na leitura e produção de plantas arquitetônicas. Assim como qualquer sistema de *machine learning*, o GAN aprende a partir de um banco de dados fornecido ao sistema. Neste caso ele está divido em "gerador" e "discriminador". Enquanto este último é treinado para reconhecer imagens de um conjunto de dados e distinguir imagens que não fazem parte deste conjunto, o gerador cria imagens. A partir do feedback do sistema discriminador, o sistema gerador se adapta para produzir novas imagens mais realistas. Este looping entre as duas estruturas têm um grande potencial de criar imagens (Figura 40).

Figura 39 – Etapas do sistema desenvolvido por Chaillou (2019)



Fonte: Chaillou (2019, p. 32)

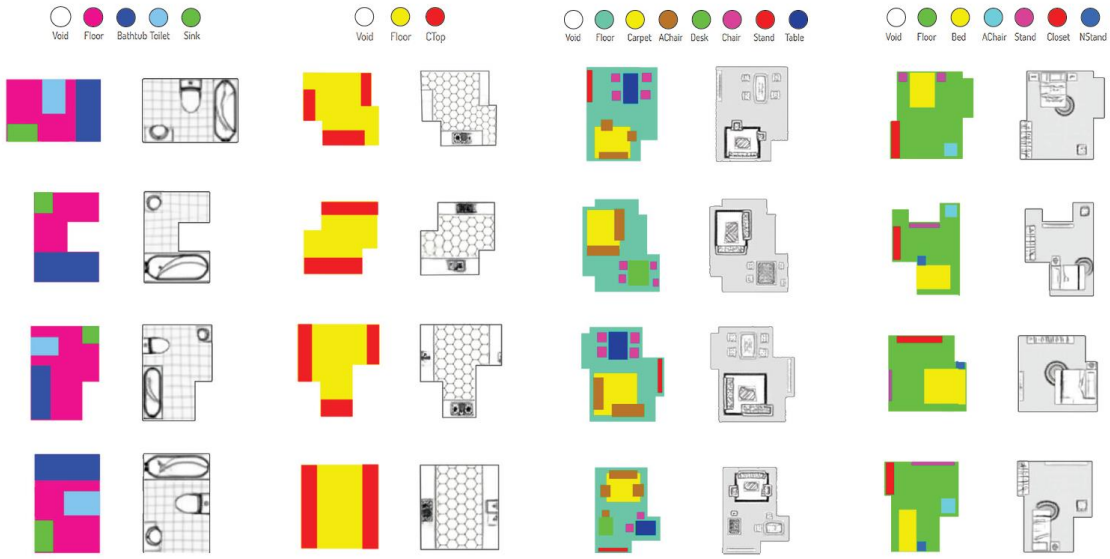
Figura 40 – Típica estrutura de Redes Neurais Generativas Adversárias (GAN)



Fonte: Chaillou (2019, p. 26)

Inicialmente, para a geração dos contornos das plantas, o modelo foi treinado com uma base extensiva de plantas de edifícios de Boston. Na sequência, o sistema foi treinado com mais de setecentas plantas de apartamentos, a fim de ser capaz de propor a subdivisão em cômodos. Por fim, a rede neural é capaz de analisar cada uma das plantas e propor sua ocupação com um layout de mobiliário. Em um primeiro momento essa representação não é acurada, então a imagem passa por uma etapa final de renderização, isto é, o mobiliário é representado com maior grau de detalhe. Isso ocorre pela tradução de retângulos coloridos em tipos específicos de mobiliário (Figura 41).

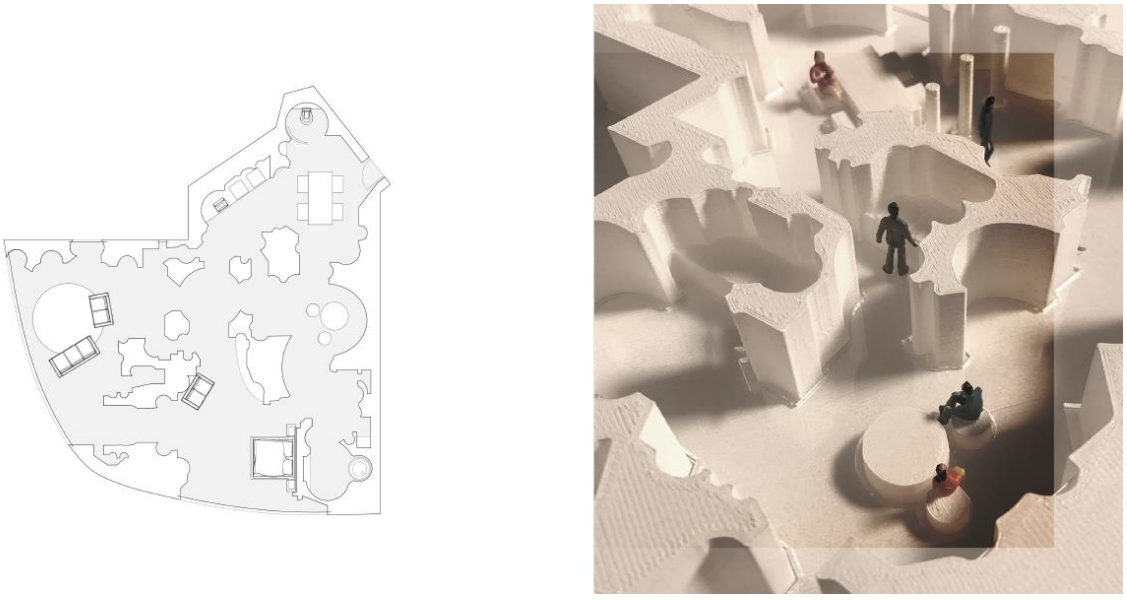
Figura 41 – Processo de renderização de plantas



Fonte: Chaillou (2019, p.28)

Após a consolidação desta metodologia, o modelo é treinado para interpretar estilos arquitetônicos, como Barroco ou uma unidade de Manhattan. Diversas aplicações foram realizadas para diferentes formas e plantas (Figura 42). Chaillou (2019) demonstra, portanto, que algoritmos de *machine learning* podem ser aplicados na análise e geração de novas plantas em contextos bem diversos, inclusive com aplicação de diferentes estilos arquitetônicos.

Figura 42 – Planta gerada em estilo barroco



Fonte: Chaillou (2019, p. 102)

Os trabalhos relatados neste capítulo formam uma amostra de possíveis abordagens e resultados da automatização de layouts de mobiliário no projeto de Arquitetura. Fica

evidente a existência de possibilidades de atuação desde problemas bem definidos até problemas mal definidos. A geração de layouts com menor escopo e quantidade de elementos é possível por meio de algoritmos determinísticos (ABDELMOHSEN; ASSEM; TARABISHY, 2016, ANDERSON *et al.*, 2018; CÔCO JÚNIOR; CELANI, 2018) e possibilita trabalhar com diversos ambientes simultaneamente (KJØLAAS, 2000). Outros autores buscam atuar sobre cenários mais flexíveis e não tão restritos (GERMER; SCHWARZ, 2009; MERRELL *et al.*, 2011; KÁN E KAUFMAN, 2017, 2018; CHAILLOU, 2019), alcançando bons resultados.

Em todos os casos, os autores identificam determinados aspectos que podem ser incluídos entre os parâmetros do algoritmo, como estética, ergonomia e funcionalidade (KÁN E KAUFMAN, 2018) ou somente o uso do espaço (KJØLAAS, 2000). Pode-se inferir que outras características mais ligadas à subjetividade dificilmente passariam por um processo de algoritmização com as tecnologias atuais, devido a sua característica *wicked*. Mesmo as atuais abordagens de *machine learning* só conseguem criar soluções a partir de um grande banco de dados, isto é, de soluções já desenvolvidas anteriormente (CHAILLOU, 2019). Há, portanto, uma criação a partir de padrões já estabelecidos e não uma real inovação.

Todos os trabalhos descritos acima atuam durante a fase de concepção com o objetivo de reduzir o esforço nas tarefas repetitivas que são automatizáveis, aumentando a produtividade dos projetistas. Entretanto, não há um foco na interface entre Arquitetura e projeto de interiores com o objetivo de encurtar as distâncias entre a fase de concepção e aquelas com maior detalhamento. É comum que a geração automatizada de layouts seja utilizada apenas como uma forma de representação, buscando uma semelhança com o mundo real, mas sem o foco na melhor solução arquitetônica, como pode ser visto na indústria de jogos. Por outro lado, o processo algoritmo pode ser utilizado como uma ferramenta de interação entre o potencial cognitivo dos arquitetos e a racionalização algorítmica para a elaboração de projetos. Possivelmente, os trabalhos que mais se aproximam disso são o de Anderson *et al.* (2018), o qual possui um *workflow* integrado com BIM e o de Côco Júnior e Celani (2018), que sistematizam uma integração entre o projeto de banheiros pré-fabricados e os desenhos para a fabricação. Outra discussão pouco explorada é em relação às novas articulações entre arquitetos e as ferramentas de Inteligência Artificial. Esse aspecto está mais presente em Côco Júnior e Celani (2018) e em Belém, Santos e Leitão (2019).

Os resultados dos trabalhos apresentados apontam que determinados processos de automatização de layout de interiores podem ter um resultado equivalente ou superior aos processos manuais (MERRELL *et al.*, 2011, KÁN E KAUFMAN, 2018). Os experimentos de Merrell *et al.* (2011) e Kán e Kaufman (2018) de certa forma, poderiam ser considerados aprovados no teste de Turing (1950), o qual propunha que alguém não conseguiria distinguir se estaria conversando com uma máquina ou com outro ser humano. Da mesma forma, Kán e Kaufmann (2017) e Merrell *et al.* (2011) buscam medir se robôs-arquitetos pode se passar por um arquiteto-humano, mas não chegam a propor uma possível simbiose entre arquitetos e máquinas.

De qualquer forma, os experimentos apontam para a possibilidade de aplicação de algoritmos para minimizar o esforço dos projetistas e potencializar o processo criativo. Essa relação é corroborada pelo aumento das capacidades computacionais e do desenvolvimento de novas tecnologias. Darko *et. al*, (2020) fazem uma pesquisa estequiométrica sobre a aplicação de tecnologias de Inteligência Artificial em Arquitetura, Engenharia e Construção (AEC). Os autores chegam a vinte tecnologias com maior importância de aplicação no setor. Dentre elas, o BIM se encontra entre as dez primeiras, juntamente com *machine learning* e algoritmos genéticos. Esta dissertação faz aplicação de algoritmos em BIM como meio para encurtar a distância entre as fases de concepção e de desenvolvimento.

2.3 A construção de algoritmos em BIM

Neste capítulo, um panorama é realizado sobre os paradigmas de BIM e de Linguagem de Programação Visual (LPV). Ambos foram utilizados como estratégias para o desenvolvido desta dissertação.

2.3.1 BIM

Os benefícios do BIM já são amplamente reconhecidos na literatura como um meio para elevar a qualidade e eficiência projetual (YIN *et al.*, 2019). Visto a recorrente extrapolação de orçamento e cronograma tanto em países desenvolvidos quanto subdesenvolvidos (ABANDA; TAH; CHEUNG, 2017), há um esforço de governos ao redor do mundo para melhorar o desempenho desse setor. Um exemplo disso é o *UK Construction Strategy 2025*, um programa do governo britânico que pretende melhorar o desempenho da indústria. Para tanto, a pré-fabricação e o BIM foram identificados como campos estratégicos de atuação. Segundo os autores, os benefícios da pré-fabricação são alavancados com a

utilização de BIM na medida em que esse favorece a colaboração nas etapas iniciais de projeto, o armazenamento de informação em bibliotecas padronizadas e o acesso a esses dados ao longo da vida do edifício. Outro exemplo é o governo brasileiro que vem desenvolvendo estratégias para o setor. Em 2015, por exemplo, o governo financiou uma pesquisa estruturada do BIM na União Europeia e no Brasil, desenvolvendo um conjunto de recomendações e conclusões para a difusão do BIM no Brasil. Os objetivos eram "incorporar o fortalecimento do BIM no país como uma 'agenda estratégica nacional da construção civil; desenvolver ações coordenadas [do governo] [..]'; de forma gradual (por estágios), tornar o BIM obrigatório em projetos e obras do governo federal [..]" (MOHAMAD; AMORIM, 2015, p. 6). Três anos depois, por meio do decreto nº 9.377 ficou instituída a Estratégia Nacional de Disseminação do BIM no Brasil (Estratégia BIM BR). O plano, entre outros objetivos, visa fazer o PIB da Construção Civil crescer de 2,0% para 2,6% ao ano entre 2018 e 2028. Já em 2020, o decreto Nº 10.306 de 2 de abril de 2020 instituiu a aplicação gradual do BIM para obras federais até 2028. O BIM é um investimento estratégico para a indústria no presente e para cenários futuros (ESTRATÉGIA BIM BR, 2018).

Oesterreich e Teuteberg (2016) discutem como a digitalização e a automatização observada em setores industriais estão presentes na construção civil. Os resultados qualitativos de uma revisão sistemática da literatura apontam que o BIM é considerado uma tecnologia central para a construção civil dentro da quarta revolução industrial. Enquanto a pré-fabricação permite uma racionalidade produtiva, modularização e flexibilidade de cronograma, mediante a independência de serviços, sua integração com BIM contribui para uma melhor gestão do fluxo de informação. Hamid, Tolba e Antably (2018) defendem que pode existir um melhor fluxo de trabalho entre projetistas e fabricantes por meio da obtenção de arquivos para corte em máquinas de controle numérico a partir da modelagem de informação. Eles concluem que o seu uso e a automatização do fluxo de informação trazem benefícios tanto para projetistas quanto para fabricantes.

Paralelamente, algoritmos podem estar associados ao BIM para geração, otimização, análise e fabricação. Eastman *et al.* (2014) afirmam que o BIM permite que arquitetos e engenheiros executem simulações e análises nas etapas iniciais de projeto. A construção de algoritmos e a sua integração com um sistema paramétrico viria a favorecer esses mecanismos de avaliação. Deste modo, há tanto uma exploração de possibilidades quanto convergência para as melhores soluções. Bianconi, Filippucci e Buffi (2019), por exemplo, desenvolveram um sistema generativo para obtenção de habitações com melhor desempenho térmico. Os dados são carregados em uma plataforma online de visualização onde o usuário pode baixar

o envelope da edificação, um render, imagens, gráficos de desempenho térmico, um arquivo DWG contendo um desenho técnico e um arquivo IFC. Similarmente, WANG (2017) explora a conexão entre o projeto paramétrico e BIM para o desenvolvimento de fachadas. Veloso, Celani e Scheeren (2018), por sua vez, trabalham com a geração de layouts de apartamentos em BIM por meio de uma gramática da forma.

Diante deste contexto, o BIM é entendido como paradigma fundamental na instanciação da metodologia proposta nesta dissertação. A capacidade de armazenar, acessar, modificar e atribuir informações aos objetos com semântica construtiva potencializam o trabalho com informações não materiais (custo e fabricante, por exemplo) bem como de variáveis materiais (material, posição, dimensões, entre outros). Deste modo, a sua inerente característica paramétrica facilita a manipulação das informações por meio de algoritmos. Outra característica importante para o escopo deste trabalho é a capacidade de explorar relações topológicas entre os objetos, por exemplo: se um elemento A sempre está abaixo do elemento B, eles podem compor um elemento C com as relativas relações paramétricas atribuídas.

2.3.2 Programação Visual

A Linguagem de Programação Visual (LPV) é uma *"linguagem formal com notação gráfica"* (PREIDEL; BORRMANN, 2016, p. 5, tradução nossa)[13]. Esse paradigma consiste em uma linguagem orientada à objetos na qual os métodos podem ser representados como caixas e as suas relações como fios (HILS, 1992). Myers (1990) afirma que uma Linguagem Visual é composta por Programação Visual e Visualização do Programa. Enquanto o primeiro caso pode ser definido como "qualquer sistema que permite ao usuário especificar um programa em duas (ou mais) dimensões*"* (MYERS, 1990, p. 98, tradução nossa)[14], o segundo "é especificado de maneira convencional e textual, e os gráficos são usados para ilustrar algum aspecto do programa ou sua execução em tempo de execução" (MYERS, 1990, p. 98-99, tradução nossa)[15]. Assim a sintaxe da LPV é desenvolvida e representada por meios multidimensionais.

As Linguagens de Programação Visual possuem uma maior aceitação entre arquitetos em relação às escritas. Isto decorre principalmente da relativa facilidade de se construir

[13] No original: *formal language with a graphical notation.*
[14] No original: *any system that allows the user to specify a program in a two-(or-more)-dimensional fashion.*
[15] No original: *is specified in a conventional, textual manner, and the graphics is used to illustrate some aspect of the program or its run-time execution.*

relações entre métodos pelo fluxo de informação e devido a visualização em tempo real das geometrias criadas pelos algoritmos. Além disso, para Myers (1990), a utilização de dados multidimensionais favorece a leitura visual humana. O crescente aumento das comunidades de usuário e a possibilidade de compartilhar códigos e plugins também incentivam a criação de algoritmos a partir da somatória de códigos pré-existentes. Atualmente, há diversas plataformas que permitem o trabalho com este tipo de linguagem. O Grasshopper para Rhinoceros3D e Rhino.Inside.Revit da Mcneel, o Dynamo para Autodesk Revit e o Marionette pertencente ao Vectorworks são alguns exemplos de softwares disponíveis no mercado. A implementação de algoritmos por meio de Programação Visual favorece um maior acesso para os profissionais que não estão familiarizados com a criação de algoritmos e pode ser a entrada para uma programação mais complexa (MYERS, 1990). Por consequência, ao invés dos arquitetos se limitarem às soluções pré-definidas nos softwares de modelagem da construção e executarem tarefas uma a uma, eles mesmos podem desenvolver seus algoritmos conforme a necessidade de responder a determinado problema. Este trabalho adota a LPV como uma ferramenta com o objetivo de aproximar procedimentos algoritmos de profissionais que ainda não estão familiarizados com área, bem como visa facilitar implementações futuras do código pela comunidade. Mesmo assim, trechos específicos do código foram desenvolvidos em Python dentro da LPV.

Preidel e Borrmann (2016) fazem uso desse paradigma para o desenvolvimento de sistemas que verificam se determinados aspectos de um edifício se encontram dentro das normas legais e de segurança do local em que estão sendo construídos. Os autores também fazem uma classificação dos possíveis métodos dentro de uma LPV a serem utilizados. Essa categorização pode ser observada na Tabela 4. Essa classificação de métodos foi utilizada na documentação do artefato desenvolvido nesta dissertação.

Tabela 4 - Classificação dos métodos e operadores em Linguagem de Programação Visual em BIM

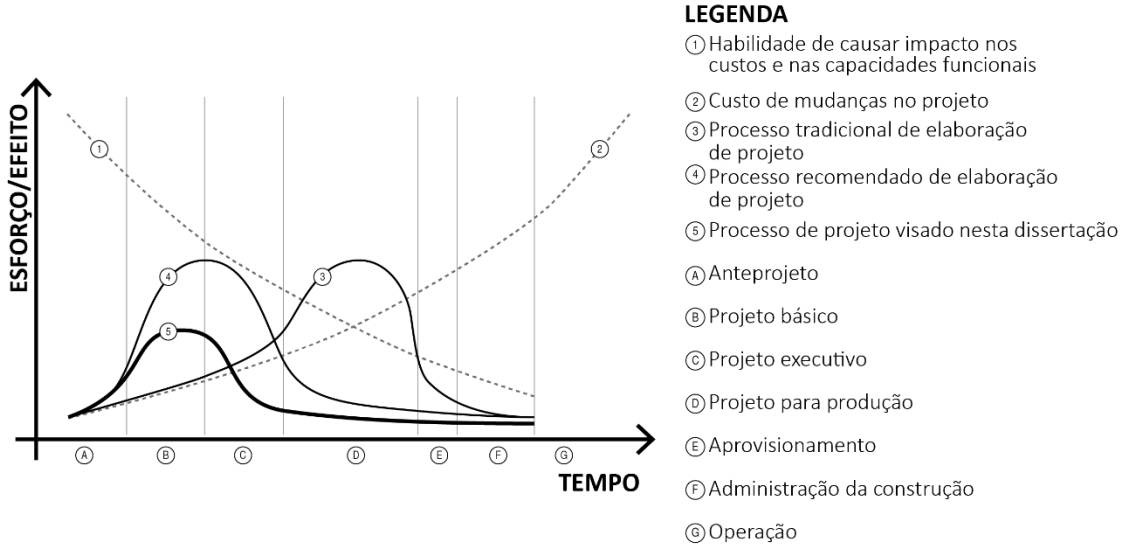
Métodos	Operadores operações	Exemplos
Lógicos	Comparam dois valores e retornam um valor booleano	*Equal, Unequal, LargerThan, SmallerThan*
	combinam valores booleanos	*And, Or, Not, Xor, IsNotDefined*
Matemáticos	Básicos	*Plus, Minus, Product, Division*
	Conjuntos	*Union, Intersection, Complement, Difference*
Geométrico-Topológicos	(1) Retornam atributos booleanos e topológicos (2) Atributos geométricos (3) Atributos direcionais	(1) *Equal, Disjoint, Touch, Overlap, Contains, Within* (2) *CloserThan, FartherThan* (3) *Above, Below*
	Operações geométricas gerando objetos geométricos a partir dos existentes	*ConvexHull, Sceleton*

	Operadores de avaliação geométrica	*ShortestDistance*, *MaximalDistance*
Relacionais	Álgebra relacional	*Projection*, *Selection*, *Join*
	Agregação	*Sum*, *Average, Min, Max, Count*
Relacionados a Modelagem do Edifício	Seleção de elementos da construção	*TypeFilter*, *Filter*, *Selection*
	Operadores para acessar e recuperar dados de atributo	*GetProperty*, *GetRelated*
Métodos utilitários	Visualização de conteúdo	*Panel*, *Watch*

Fonte: Adaptado de Preidel e Borrmann (2016, p. 410)

A associação do BIM com algoritmos abre a oportunidade para rever abordagens estabelecidas na Arquitetura. Ambos os paradigmas, entretanto, devem ser vistos como conjuntos de ferramentas que podem permitir maior produtividade para arquitetos nas fases iniciais de projeto, reduzindo o esforço inicial a cada novo projeto (Figura 43).

Figura 43 – Curvas de esforço/efeito projetual e construtivo ao longo do tempo. A associação do BIM com algoritmos pode reduzir o esforço inicial de projeto



Fonte: adaptado de Eastman *et al.* 2014

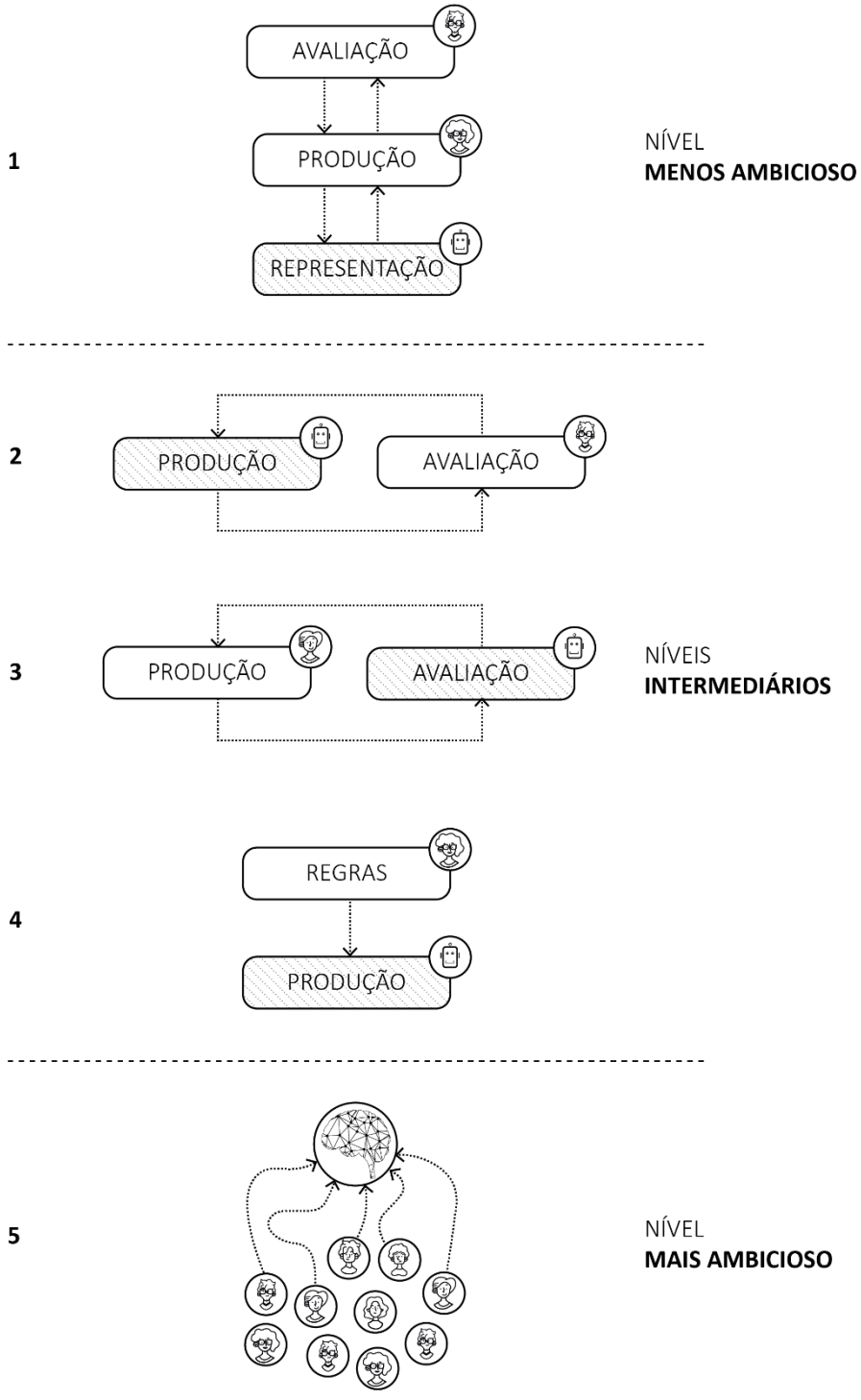
2.3.3 Articulações entre arquitetos e máquinas no processo de concepção projetual

Diante deste contexto de diferentes classes de problemas na Arquitetura, de abordagens algorítmicas e paradigmas para fluxos de trabalhos mais eficientes, uma discussão pode ser realizada sobre quais são as articulações possíveis entre arquitetos e máquinas no processo de concepção projetual.

As diferentes instâncias do projeto arquitetônico se distribuem dentro de um intervalo no qual há em um dos seus extremos problemas bem definidos e no outro problemas *wicked*. No geral, aqueles que estão enquadrados no primeiro grupo são mais facilmente automatizados, inclusive há uma maior receptividade dos arquitetos para este tipo

de automatização. Quando se trata das outras classes de problemas, ainda há uma centralização nos esforços humanos. Nos anos 1970, Mitchell (1975) já apresentava cinco níveis de colaboração entre arquitetos e máquinas, segundo a classificação de problemas descrita anteriormente no capítulo 2.1 (Figura 44).

Figura 44 - Cinco níveis de divisão do trabalho entre arquitetos e máquinas podem dividir o trabalho



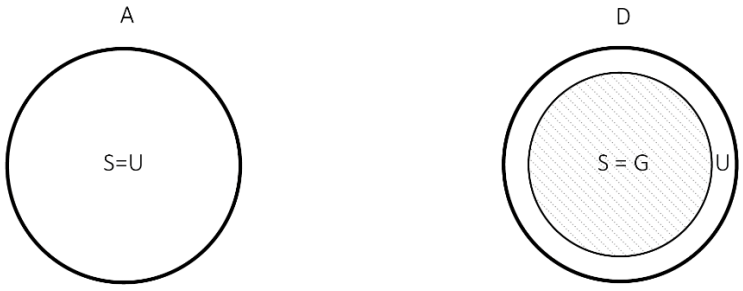
Fonte: elaborado pelo autor baseado em Mitchell (1975)

Mitchell (1975) já afirmava que (1) a representação é o nível menos ambicioso para o uso de computadores. Isto é, quando os computadores são utilizados como banco de dados, produção de relatórios, desenhos ou perspectivas. Os problemas neste caso são quais tipos de elementos devem ser inseridos, seus atributos e relações.

Em um nível intermediário, (2) a máquina pode receber a função de avaliar as soluções geradas por arquitetos. Nesta utilização, há o pré-requisito que os critérios da solução sejam explícitos e bem definidos. Esse uso ocorre, por exemplo, em sistemas de simulação estrutural, térmico, acústico e de orçamento. (3) Mitchell (1975), descreve, ainda, que esta lógica poderia ser invertida caso o conjunto de soluções fosse bem definido, mas os critérios para a escolha não. Assim, o computador poderia gerar alternativas de soluções dentro de um espaço pré-determinado como aceitáveis enquanto os arquitetos poderiam selecionar uma delas por questões mais sutis e subjetivas. Exemplos disso são os trabalhos apresentados anteriormente que possuem uma maior capacidade interativa entre arquitetos e o sistema, como ocorre no trabalho de Merrel *et al.* (2011) e Anderson *et al.* (2018) que sugerem diferentes layouts e cabe o usuário escolher a melhor solução.

Quando os conjuntos de soluções em potencial e os critérios de solução são completamente definidos (Figura 45), (4) o processo de projeto pode ser totalmente automatizado. Mitchell (1975) aponta que este tipo de abordagem tende a ser bastante restrita em seu escopo, mas já observava a aplicação na produção de plantas arquitetônicas e estruturais ideais. O trabalho de Eastman (1968a), de desenvolvimento de layout de banheiros por meio de protocolos e algoritmos procedurais, é um bom exemplo para este caso. A mesma orientação pode ser encontrada em Côco Júnior e Celani (2018), Anderson *et al.* (2018) e Kjølaas (2000).

Figura 45 - Casos A e D segundo os modelos de relações entre geração de soluções e objetivo de Mitchell (1975). Modelos aplicáveis para problemas bem definidos



Fonte: elaborado pelo autor

Por fim, (5) o nível mais ambicioso descrito por Mitchell (1975) e Negroponte (1970) é aquele em que as máquinas seriam capazes de lidar com problemas mal definidos de modo inteligente e flexível, assim como um bom projetista humano desenvolveria ou até melhor. Os trabalhos de Kán e Kaufmann (2018) e Merrel *et al.* (2011) demonstram por meio de experimentos que dentro de determinados recortes voltados para parâmetros funcionais, as máquinas podem ter um desempenho igual ou superior que os dos projetistas. Entretanto,

as abordagens estão longe de responder a toda a complexidade projetual, como os próprios autores ressalvam.

A subutilização do potencial computacional e a sua redução à representação para problemas mal definidos faz com que possíveis soluções não sejam consideradas devido à limitação da análise humana, da sua cognição e da sua memória projetual, bem como de limitação de tempo e de orçamento (caso B, Figura 45). Semelhantemente, outro sintoma é ter a percepção de um conjunto maior de soluções possíveis, mas não ter os meios para alcançá-los, restringindo as soluções viáveis a um conjunto com um número menor do que poderia ser (caso E, Figura 45). Enquanto esses dois casos são marcados por limitações dos projetistas, há a possibilidade de se ter uma assertividade menor também nas soluções, pela geração de resultados que não respondem da melhor maneira possível aos problemas levantados (casos C e F, Figura 45).

Nos casos descritos por Mitchell (1975) nos quais há associação entre arquitetos e máquinas, por sua vez, amplia-se a exploração de soluções, ao mesmo tempo que se fornece subsídios consistentes para a tomada de decisões projetuais de modo mais criterioso. Essa abordagem se adequa bem aos problemas mal definidos por expandir o campo de soluções possíveis, enriquecendo o ciclo de identificar novas questões, possíveis respostas e abordagens para alcançá-las. Evidentemente, mesmo que haja maior automatização, os processos são calibrados pelos projetistas que estabelecem parâmetros, regras e procedimentos para a exploração e convergência em uma solução. É possível que o refinamento desses processos permita um maior controle de problemas mal definidos, conduzindo-os para problemas mais controlados após inúmeros processos de racionalização. Entretanto, isso reduz sua flexibilidade e generalização. Em todos os casos, mesmo nos processos mais automatizados, não há a exclusão dos arquitetos do projeto arquitetônico; o que ocorre é que sua atuação pode acontecer de outras maneiras. Os arquitetos podem determinar os critérios e parâmetros pertinentes ao início, desenvolvimento e avaliação de resultados.

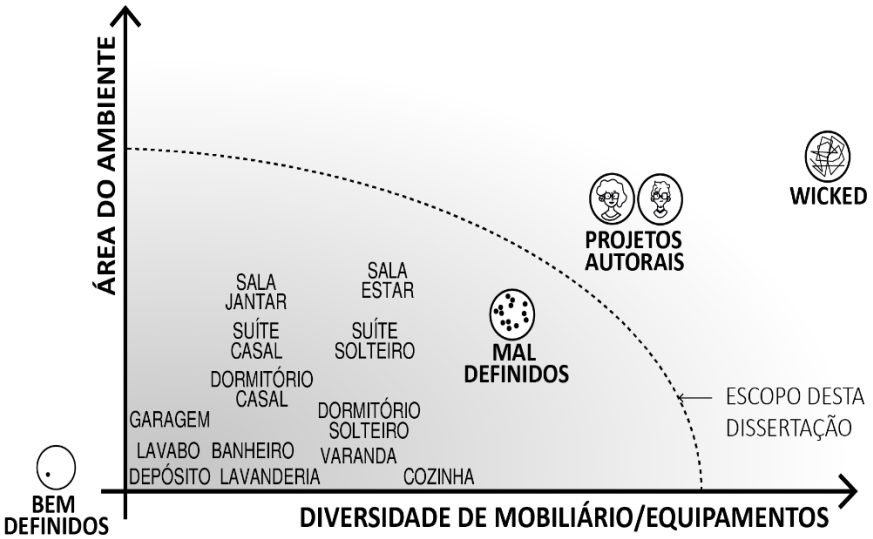
Ainda em relação aos problemas mal definidos na Arquitetura, muitos deles podem ser do tipo *wicked* quando o projeto apresenta um grande escopo, muitas condicionantes, grande número de unidades de projeto e maior liberdade criativa. Esses fatores contribuem para uma maior subjetividade projetual. Existem campos de pesquisa na Arquitetura que procuram explorar especificamente essas questões, como a Fenomenologia (KOLAREVIC; KLINGER, 2008; NORBERG-SCHULZ, 2008; PALLASMAA, 2017), a Complexidade

(MORIN, 2015) e a Neurociência (DAMÁSIO, 2004). Observa-se que os projetos autorais de Arquitetura apresentam muitas destas características citadas, o que, consequentemente, inviabilizaria sua completa automatização com as ferramentas atuais de Inteligência Artificial (BELÉM; SANTOS; LEITÃO, 2019).

Dito isto, três pontos devem ser destacados:

Em primeiro lugar, em uma análise qualitativa sobre a geração de layouts de espaços de habitações verticais, observa-se que existe uma gradação entre espaços que se aproximam mais de problemas bem definidos e de outros mal definidos, quando observados pelos parâmetros de área e quantidade de mobiliário/equipamentos. Se dentro da análise sistêmica for levado em conta somente parâmetros como as dimensões das unidades de projeto, seu posicionamento, orientação e relações topológicas, um lavabo será muito mais enquadrado dentro de um problema bem definido que uma suíte de casal ou uma sala de estar, por exemplo. Nestes últimos casos, a formulação do problema é menos rigorosa e menos estável, abrindo mais possibilidades de arranjo espacial que a primeira situação. Em segundo lugar, mesmo que seja possível identificar padrões para resolver estes layouts, quando se insere a variável da subjetividade dos projetistas, isto é, de projetos autorais, os problemas se aproximam mais de problemas *wicked*. Isso se dá pela habilidade abdutiva humana de criação (Figura 46). Mesmo que exista uma lógica intuitiva de arranjo de layouts, ele sempre será um problema único, sem uma formulação definitiva e sem um critério claro de finalização.

Figura 46 - Diagrama qualitativo de classes de problemas para diferentes tipos de layout de ambientes habitacionais



Fonte: Elaborado pelo autor

Com base nestes dois primeiros pontos é possível realizar uma distinção entre tipos de layout e quais informações estão nele associadas. Sabe-se que a criação de um projeto de Arquitetura é incremental por apresentar diversos problemas mal definidos que precisam de resolução. Assim, as soluções geradas em determinada etapa de desenvolvimento produzem informações que serão utilizadas para a próxima fase e assim sucessivamente. O layout nas fases iniciais apresenta informações genéricas, como dimensões, posição e orientação. Com estes parâmetros associados a boa parte dos elementos no início do projeto, é possível gerar diferentes arranjos espaciais e com isso analisar setorização, circulação, acessibilidade, iluminação, acústica, disposição de pontos de elétrica e hidráulica, criação de visuais e dinâmicas de usos. Assim, percebe-se que, mesmo que não se saiba qual o modelo exato de mobiliário ou equipamento empregado, já é possível explorar diferentes qualidades espaciais desde as primeiras etapas do processo criativo. Esse layout pode ser chamado de "instrumental" e nas fases seguintes de anteprojeto e de projeto executivo, deve receber informações mais detalhadas, como a do fabricante, dimensões precisas e esquemas de montagem. Desta maneira, problemas essencialmente mal definidos podem ser fracionados em partes menores de modo a se aproximarem de problemas bem definidos e com isso serem traduzidos em algoritmos e executados por um computador.

No início dos anos 1970, Negroponte (1970) já propunha que arquitetos e máquinas deveriam ser vistos como duas espécies distintas, cada um seguindo sua própria linha evolucionária. Assim, ao invés dos computadores executarem tarefas como "escravos" dos projetistas, eles poderiam interpretar as intenções subjetivas humanas e propor abordagens para a resolução de problemas. Em outras palavras, o autor propunha uma simbiose entre as duas espécies, potencializando o que cada uma desenvolve de melhor. Uma análise que, ainda hoje, permanece relevante.

Nesta dissertação, a discussão apresentada por Negroponte (1970) motivou uma analogia das possíveis relações entre arquitetos e máquinas em relações ecológicas do campo da Biologia. Essas associações podem ser harmônicas ou desarmônica. Algumas delas se encontram definidas abaixo:

- Comensalismo: "1. associação simbiótica entre duas espécies na qual uma (o comensal) é beneficiado enquanto o outro (o hospedeiro) não é afetado. 2. Estado de viver ou comer benignamente juntos" (MAI; OWL; KERSTING, 2005, p. 112, tradução nossa);

- Protocooperação: "forma de simbiose em que ambos os simbiontes são beneficiados, mas as relações não são obrigatórias" (MAI; OWL; KERSTING, 2005, p. 440, tradução nossa);

- Mutualismo: "tipo de simbiose envolvendo interação entre dois ou mais indivíduos de diferentes espécies no qual todos os indivíduos são beneficiados. Muitas vezes mutualismo é uma relação de completa dependência entre os envolvidos." (MAI; OWL; KERSTING, 2005, p. 353, tradução do autor). Isso normalmente envolve bens ou serviços, como alimento, defesa ou transporte e tipicamente resulta na obtenção de novas habilidades por pelo menos um parceiro (HERRE; WEST, 1997 *apud* BEGON; TOWNSEND; HARPER, 2006). Um exemplo desta relação na natureza acontece entre o peixe *Labroides dimidiatus* que come ectoparasitas, bactérias e tecido necrótico do peixe *Hemigymnus melapterus* que fornece proteção (BEGON; TOWNSEND; HARPER, 2006);

- Competição: "[..] Associação simbiótica em que o crescimento ou a sobrevivência de uma das espécies de populações é afetada negativamente nos casos em que um recurso comunitário é escasso" (MAI; OWL; KERSTING, 2005, p. 113, tradução do autor);

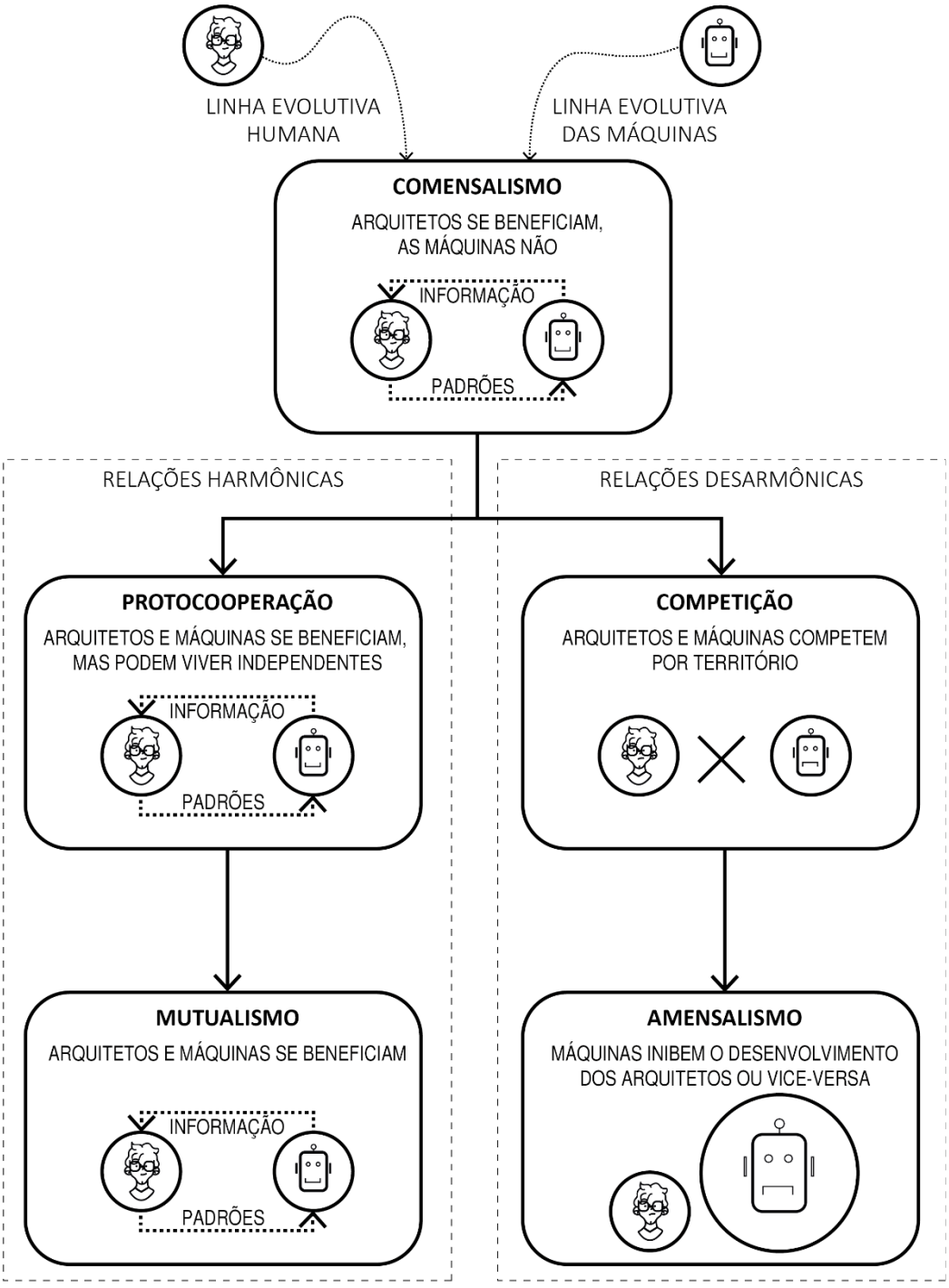
- Amensalismo: "Uma associação entre duas espécies que é prejudicial para uma das espécies, mas não tem efeito na outra" (MARTIN; RUSE; HOLMES, 2005, p. 21, tradução do autor). Um exemplo comum disso é a liberação de toxinas químicas por algumas plantas para inibir o crescimento de outras (MARTIN; RUSE; HOLMES, 2005).

Šijaković e Perić (2017) se utilizam do conceito de simbiose e as relações ecológicas de comensalismo, mutualismo e parasitismo para expressar relações entre edifícios e intervenções por meio de um processo de *retrofit*. Wang, Gao e Váncza *et al.* (2019) estudam o trabalho colaborativo em tempo real entre humanos e robôs no qual é necessária uma ação coordenada de ações e de comunicação, uma simbiose na fabricação. Essa mesma interação entre humanos e robôs também é discutida por Gill (2012).

Em uma leitura no campo do projeto e fabricação da Arquitetura (Figura 47), quando os computadores são utilizados apenas como representação, isto é, no nível menos ambicioso (MITCHELL, 1975), pode-se dizer que há uma relação comensalista. Isto ocorre porque os arquitetos são beneficiados, mas as máquinas não. Dados são analisados e geometrias são

desenhadas nos computadores, mas eles são usados como uma calculadora ou uma prancheta virtual. Todo o desenvolvimento está concentrado na inteligência humana.

Figura 47 - Relações ecológicas entre arquitetos e máquinas inspirado em Negroponte (1970)



Fonte: elaborado pelo autor

Essa relação pode evoluir para relações harmônicas ou desarmônicas. No primeiro caso, pode existir inicialmente uma protocooperação entre as duas espécies, isto é, ambos se

beneficiam, mas podem viver independentes. Os algoritmos podem gerar ou avaliar soluções por humanos. Por um lado, os arquitetos podem ter soluções mais adequadas e um espaço de soluções possíveis maior que processos tradicionais, por outro lado, a inteligência implementada no computador aumenta, ou seja, ele também é beneficiado.

Em um grau de profundidade maior, arquitetos e máquinas podem desenvolver uma relação mutualística na qual há uma completa independência entre ambos. Esta seria uma simbiose completa e o nível mais ambicioso de uso dos computadores. Por meio de técnicas de *machine learning,* as máquinas possuiriam uma capacidade de analisar grande quantidade de informações fornecidas pelos arquitetos e organizá-las em padrões, resultando em sugestões para os projetistas, antevendo problemas e resolvendo-os como ou de modo mais eficiente que os seres humanos. Conforme mais iterações fosse realizada, maior seria a inteligência e independência das máquinas para executar ações sem a necessidade de intervenção humana. Esse sistema de retroalimentação e aprendizado já apresenta resultados avançados na indústria automobilística. A Tesla, por exemplo, já aplica algoritmos de *machine learning* para carros autônomos. Enquanto eles se deslocam pelas ruas, sensores alimentam um banco de dados da empresa os quais são utilizados para treinar a inteligência artificial e permitirem resultados cada vez melhores a cada trajeto. Neste sistema de completa autonomia, será possível simplesmente entrar no carro e dizer para onde se quer ir ou simplesmente nem falar nada. O carro terá a capacidade de olhar o calendário do passageiro e levá-lo para o seu destino (TESLA, s.d.; TESLA, s.d.).

Similarmente, o mesmo princípio poderia ser aplicado na Arquitetura. No caso de arranjo de mobiliário, um sistema algoritmo pode popular determinado cômodo e receber pequenos ajustes do projetista. A Inteligência Artificial, por sua vez, analisaria estas modificações e, após inúmeras iterações, não erraria mais nenhum layout. Deste modo, os projetistas dividiram o trabalho com um parceiro virtual de trabalho. Embora nesta dissertação não se tenha aplicado sistemas de *machine learning*, sua implementação pode ser explorada em trabalhos futuros, inclusive associando com BIM.

Essa não é uma realidade distante. O mundo está passando por uma nova revolução industrial na qual as fronteiras entre o mundo digital e o biológico estão cada vez mais difusas (MENGES, 2015). Menges (2015) descreve, por exemplo, que as máquinas de produção (robôs) podem passar de um paradigma baseado em instrução para outro baseado em comportamento. Dentro do primeiro caso, os projetistas são treinados para produzirem um conjunto de desenhos e modelos para traduzir uma construção virtual para o mundo físico.

Assim, qualquer desvio da máquina das instruções predefinidas estaria relacionado a uma imprecisão que deveria ser retirada. Por outro lado, quando há uso de uma abordagem baseada em comportamento, o que seria uma imprecisão se torna um potencial gerador. Neste caso, os dados são constantemente coletados e devolvidos para o sistema, de maneira que as informações são obtidas rapidamente e podem originar novas ideias.

Alternativamente há um caminho harmônico, o comensalismo pode evoluir para relações desarmônicas. Em um primeiro momento poderia existir uma competição entre arquitetos e máquinas por espaço de mercado. Algoritmos poderiam executar diversas tarefas melhor que arquitetos, levando a uma massiva substituição da mão de obra humana. De uma outra perspectiva, os escritórios de Arquitetura que se encontram mais bem adaptados harmonicamente com as máquinas viriam a dominar o mercado e suplantar aqueles que não possuem uma simbiose com algoritmos.

Neste cenário pessimista, os arquitetos poderiam ser substituídos por máquinas. Assim, ao invés do arquiteto ser contratado para desenvolver um projeto, uma máquina o faria diretamente para o cliente muito melhor que um profissional. Uma Arquitetura sem arquitetos. Essa temática é discutida por Belém, Santos e Leitão (2019) e por Celani *et al.* (2015), os quais defendem que uma completa ausência dos arquitetos do projeto não é algo que se possa esperar em um cenário próximo. Para eles existem aspectos subjetivos diretamente influenciados pela cultura que são transitórios e efêmeros, isto é, dificilmente poderiam ser implementados em um sistema de Inteligência Artificial com as tecnologias atuais. Belém, Santos e Leitão (2019), entretanto, apresentam um cenário de competição no qual aqueles que conseguem se adaptar e dominar as tecnologias disruptivas da sua época conseguem tirar maior proveito delas. Estes cenários de interação entre arquitetos e máquinas não são estáticos, mas são transformados em sintonia com os saltos tecnológicos. Assim, uma relação harmônica pode evoluir para uma desarmônica e vise e versa.

Negroponte (1970) apontava para uma simbiose entre duas espécies, entre projeto e computação, Arquitetura e máquina. Hoje, o mundo passa por uma rápida e profunda mudança na qual as barreiras entre os meios digital e físico estão sendo rompidas e caminham de um cenário de completa independência para, possivelmente, uma relação de colaboração desde um comensalismo ou até um mutualismo entre Arquitetura e máquinas mediante o desenvolvimento da Inteligência Artificial. Os arquitetos devem se adaptar a este novo ecossistema que ganha forma a cada dia.

Dentro deste contexto, esta dissertação trabalha com a relação comensalista com o objetivo de reforçar os caminhos para uma futura simbiose mutualística entre arquitetos e máquinas, contribuindo para a consolidação de processos sistematizáveis e para o aumento da produtividade no desenvolvimento de projetos com o uso de BIM por meio da automação de layouts de mobiliário. Assim, desenvolve-se um algoritmo em BIM com as seguintes diretrizes com base na revisão da literatura:

- Criar um modelo que vá além da representação espacial;
- Desenvolver um algoritmo flexível que consiga gerar soluções para diferentes formas do ambiente e posições das aberturas;
- Favorecer a interação entre processos manuais e automatizados;
- Deixar a tomada de decisões definitivas para os projetistas;
- Automatizar tipologias como meio de facilitar projetos autorais.

Nos capítulos seguintes, apresenta-se o desenvolvimento incremental deste algoritmo.

3 MATERIAIS E MÉTODOS

3.1 Metodologia

3.1.1 Caracterização

Esta dissertação pode ser classificada segundo seus objetivos como prescritiva e, quanto aos seus procedimentos, como construtiva, seguindo o delineamento do *Design Science Research* (LACERDA *et al.*, 2013*;* DRESCH; LACERDA; ANTUNES JR., 2015*).*

3.1.2 Procedimentos

O delineamento do *Design Science Research* (DSR) consiste em pelo menos cinco etapas sequenciais. Inicialmente, a fase de Conscientização do problema diz respeito à compreensão da problemática envolvida e tem como produto a formalização do problema. A sugestão, na sequência, é uma etapa abdutiva para elaboração de uma ou mais alternativas para a solução dos problemas, resultando em um conjunto viável de artefatos e a escolha de no mínimo um deles para ser desenvolvido e testado. Esses, por sua vez, são categorizados em construtos para descrever os problemas e para especificar soluções; modelos que expressam relações entre construtos; métodos que apontam um conjunto de passos para executar uma tarefa e instanciações que dizem respeito à concretização de um artefato em seu ambiente (LACERDA *et al.*, 2013). Há, então, o desenvolvimento do artefato escolhido por meio de diferentes abordagens dedutivas, como a criação de algoritmos até o desenvolvimento de maquetes. Mesmo que o artefato venha a responder a um problema específico, ele deve ser generalizável para uma classe de problemas. Em seguida, o artefato gerado passa por uma etapa de avaliação na qual seu comportamento é testado e validado segundo relações causais, regras pré-definidas e explicitação dos procedimentos pelos quais os resultados podem ser alcançados. E, por fim, a conclusão formaliza todo o processo, discute os resultados e comunica às comunidades interessadas, sejam acadêmicas ou profissionais (LACERDA *et al.*, 2013; DRESCH; LACERDA; ANTUNES JR., 2015).

3.1.3 Conscientização

A etapa de conscientização foi realizada por meio de uma revisão da literatura sobre classes de problemas, abordagens algorítmicas, uso de BIM e Linguagem de Programação

Visual implementadas no projeto arquitetônico, em especial na fase de concepção. Ao final, foi possível contextualizar a relação entre o processo criativo dos arquitetos e a "computarização" de tarefas. Assim, a problemática desta dissertação foi definida como a falta de sistematização de etapas estratégicas e consequentemente a reduzida produtividade no desenvolvimento de projetos.

3.1.4 Sugestão e Desenvolvimento

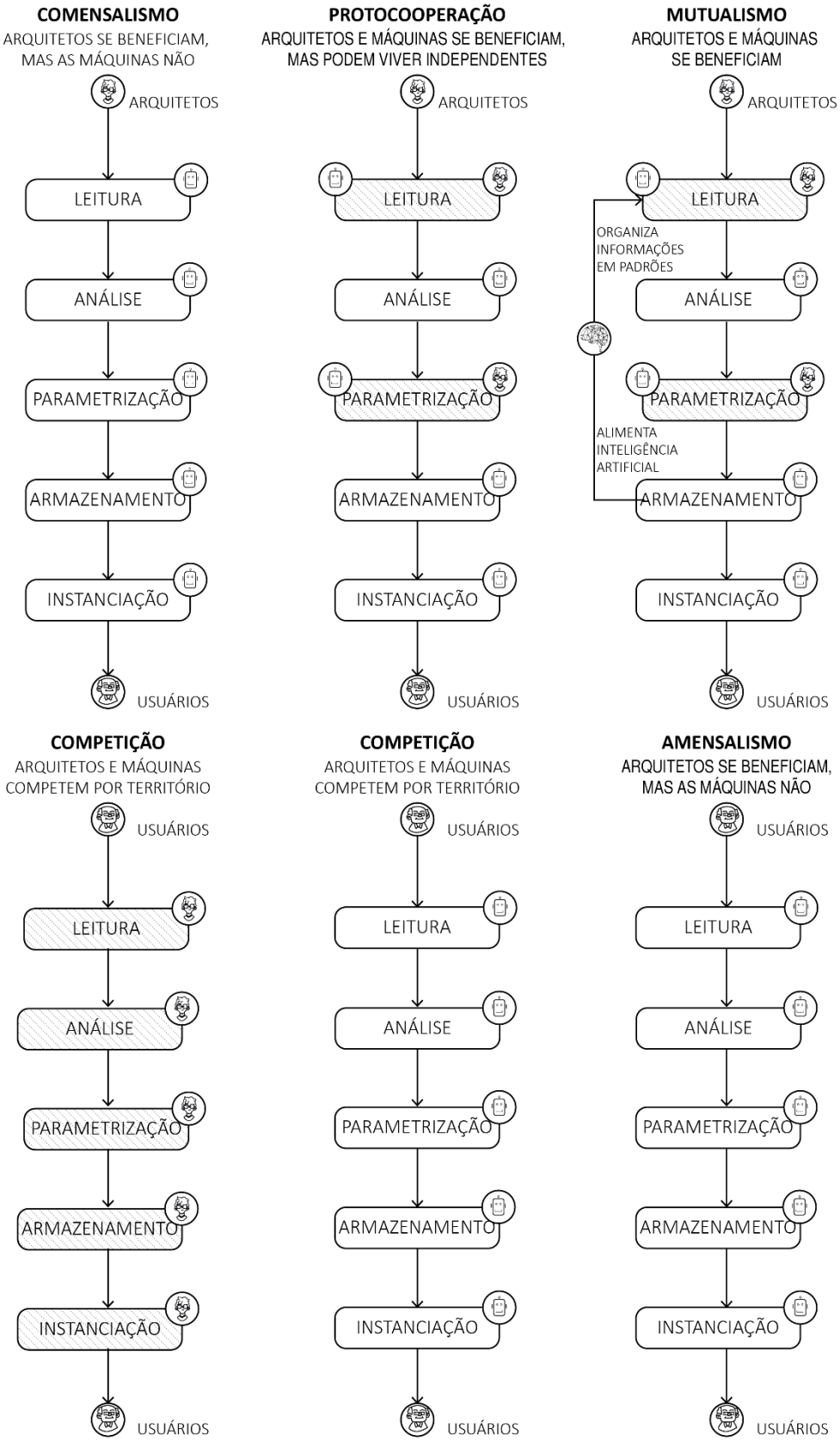
Com base na problemática definida na etapa de conscientização, possíveis Construtos, Modelos e Métodos foram sugeridos para resolver os problemas levantados. O desenvolvimento de layouts de mobiliário como um instrumento importante para tomada de decisões, desde as etapas preliminares do projeto, foi o primeiro construto definido. A criação de algoritmos foi o segundo, permitindo diminuir o esforço neste desenvolvimento e garantindo maior precisão no projeto.

Um recorte foi realizado para desenvolvimento de layouts automatizados dentro de tipologias habitacionais multifamiliares verticais. Não fazem parte do escopo desta pesquisa, portanto, projetos autorais, considerados dentro de uma outra classe de problemas.

Quando esses dois construtos foram relacionados, originaram cinco modelos que categorizam possíveis relações entre arquitetos e algoritmos. As terminologias das relações ecológicas entre seres vivos foram emprestadas da Biologia para expressar possíveis abordagens projetuais. Isso foi inspirado pela leitura de Negroponte (1970) que descreve os computadores como uma nova espécie. Cinco categorias foram definidas abdutivamente: comensalismo, competição, mutualismo, amensalismo e protocooperação. Elas foram descritas no final da seção de fundamentação teórica.

Dentro do escopo desta dissertação, uma metodologia foi descrita a partir do modelo comensalista (Figura 48). Ela se refere quando o arquiteto se encontra no início do processo de projeto, sendo responsável pela entrada dos inputs do sistema e pela definição das suas regras, enquanto toda a geração do layout acontece por meio de um sistema automatizado até chegar ao usuário final. Ela foi descrita no início da seção de desenvolvimento, na qual também se encontram suas instanciações. Os materiais utilizados estão descritos no capítulo seguinte.

Figura 48 – Modelos de relações entre arquitetos e máquinas. Os modelos comensalista e de protocooperação foram trabalhados nesta dissertação



Fonte: elaborado pelo autor

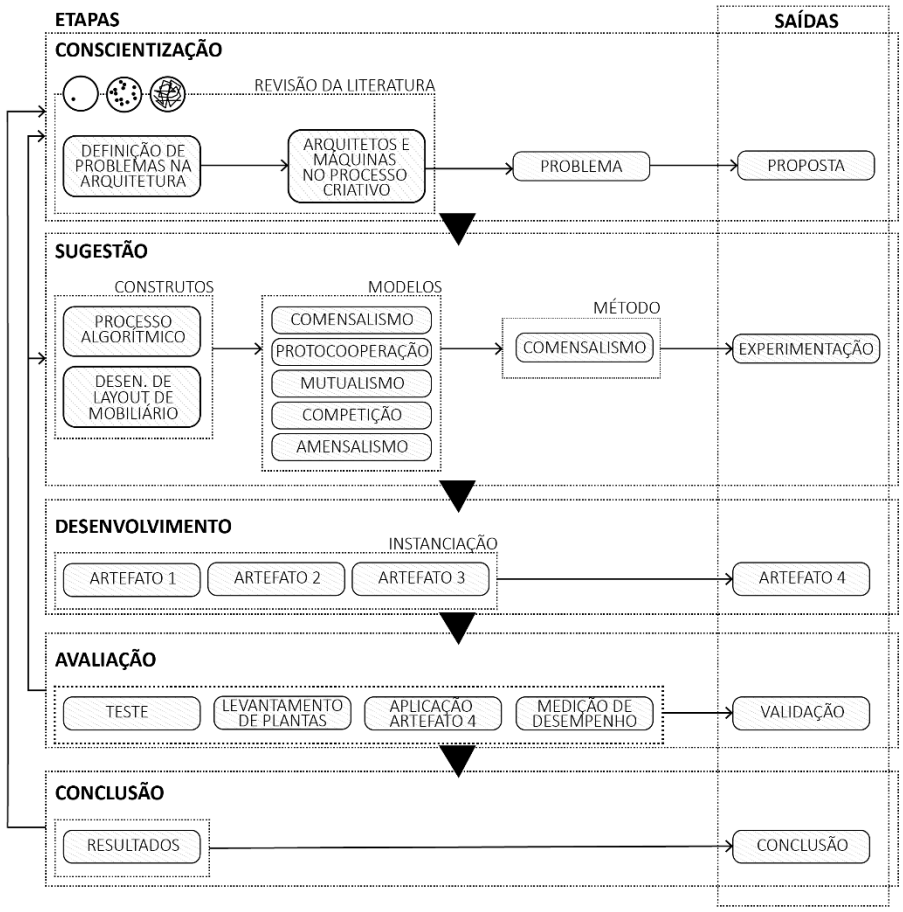
3.1.5 Avaliação

Um experimento foi proposto a fim de avaliar a eficácia dos artefatos desenvolvidos em relação ao objetivo principal desta dissertação, isto é, de sistematizar processos no projeto de Arquitetura, proporcionando um aumento da produtividade com uso de BIM por meio da automação de layouts de mobiliário. O Experimento consiste no desenvolvimento de um layout de dormitório utilizando o sistema desenvolvido. Sua descrição se encontra na seção de Avaliação.

3.1.6 Conclusão

Por fim, os resultados obtidos são descritos na conclusão bem como é realizada uma discussão dos modelos, métodos e artefato trabalhados pela perspectiva da revisão da literatura (Figura 49). Com base nos trabalhos desenvolvidos dentro do escopo desta dissertação, uma discussão é feita para trabalhos futuros.

Figura 49 – Metodologia desta dissertação com base no delineamento do Design Science Research



Fonte: elaborado pelo autor

3.2 Materiais

3.2.1 Software

Nesta dissertação, os softwares descritos abaixo foram utilizados:

- Rhinoceros - Versão: 7.0.20168.13075 (Work in progress)
- Grasshopper - Versão 1.0.0007
- Componente Python no Grasshopper - Versão 7.0.20168.13075
- IronPython 2.7.9.0
- Autodesk Revit 2021 em Português
- Rhino.Inside.Revit - Versão 0.0.7482.23623 (26/06/2020 13:07:26)
- Dynamo Core 2.5.0.7460
- Dynamo Revit - Versão 2.5.07586
- Rhythm - Versão 2020.6.22
- Spring nodes - Versão 203.2.0

Em todos os casos foram registradas as últimas versões utilizadas.

3.2.2 *Hardware*

O computador utilizado para desenvolvimento contém as seguintes configurações:

- Fabricante: Acer
- Modelo: Nitro AN515-51
- Processador: Intel(R) Core(™) i7-7700HQ CPU @2.80GHz 2.81 GHZ.
- Memória instalada (RAM): 16GB
- Tipo de sistema: Sistema Operacional de 64 bits, processador com base em x64

4 DESENVOLVIMENTO

Esta seção tem como objetivo descrever as instanciações realizadas a partir dos modelos e métodos apontados na seção de metodologia como possíveis relações entre arquitetos e máquinas. O desenvolvimento seguiu o delineamento classificado como comensalista, de acordo com o escopo desta dissertação. As instanciações foram feitas incrementalmente com o objetivo de prescrever possíveis meios para elevar a produtividade do projeto arquitetônico por meio da automatização de layouts em BIM.

4.1 Procedimentos

Os modelos de interação entre arquitetos e máquinas descritos na seção de metodologia foram subdivididos em cinco etapas sequenciais nas quais ora podem se concentrar nas habilidades humanas ora na inteligência computacional. As metodologias se iniciam com processos de leitura, seguidas de análise, parametrização, armazenamento e instanciação (Figura 50). Em essência, as diferenças entre as abordagens comensalista e de protocooperação ocorrem na fase de parametrização do sistema.

A primeira fase é a leitura. O usuário deve entrar com o projeto em BIM e com as suas informações separadas corretamente dentro de categorias. É necessário a criação de paredes, ambientes, mobiliários, portas e janelas cada um dentro da sua respectiva classificação. Assim, ao passo que estes elementos formam, no mínimo, um cômodo no modelo, as próximas etapas serão executadas automaticamente segundo um comando do usuário.

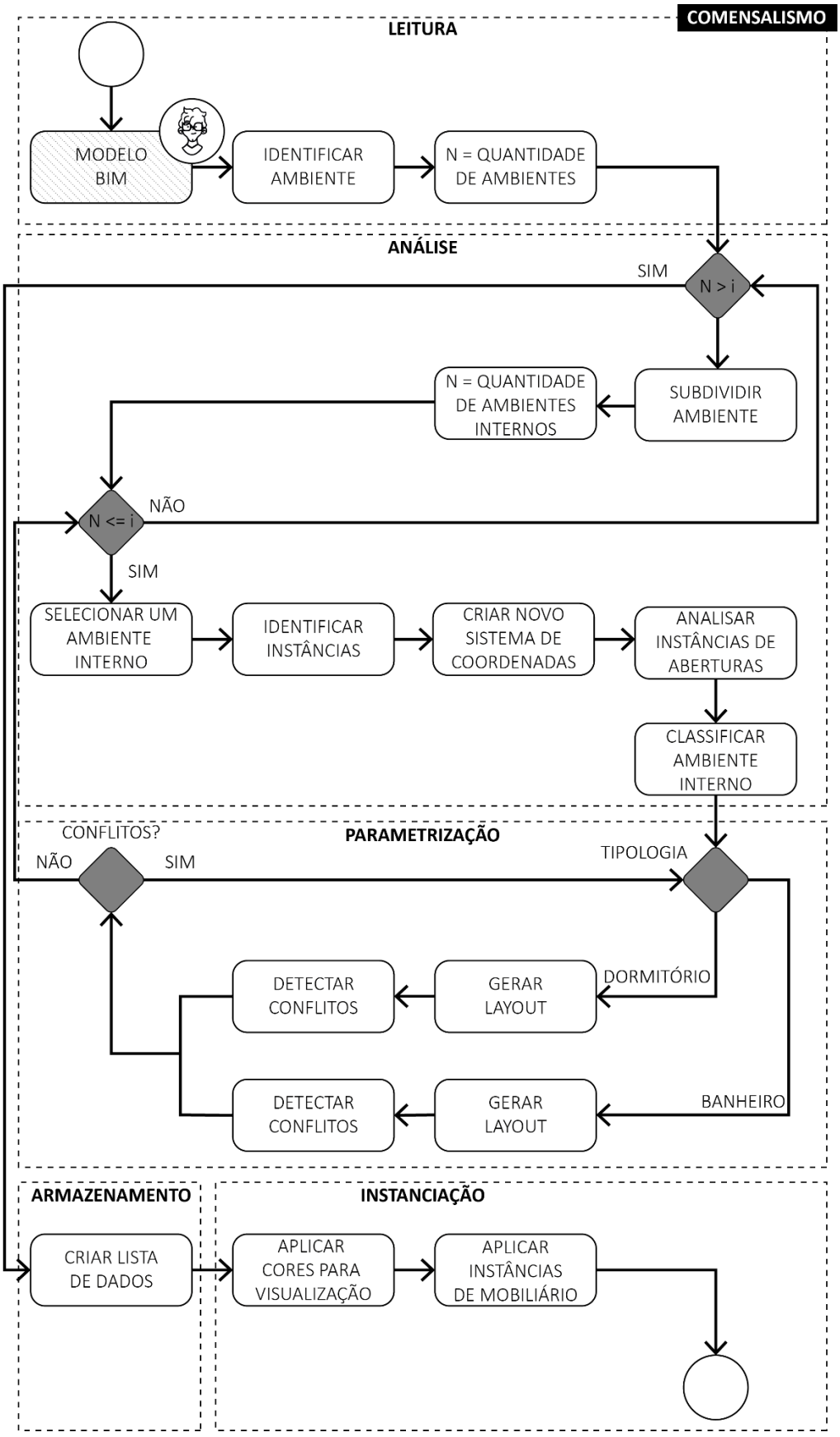
A segunda fase diz respeito à análise de cada um dos ambientes identificados na etapa de leitura. Levando em conta que pode existir uma grande quantidade de possibilidades de formas de cômodos de uma habitação, adota-se neste momento um processo de subdivisão para espaços não retangulares a fim de simplificar o sistema de colocação do layout. Na sequência, cada um dos subambientes gerados é trabalhado separadamente, ou seja, são identificados as paredes, janelas e portas que pertencem àquele conjunto. Até este momento não foi criado nenhuma distinção entre o modelo comensalista e de protocooperação, entretanto, no final da etapa de análise há uma importante diferença. Enquanto no primeiro caso a classificação do ambiente segundo sua função acontece pelo próprio sistema, no segundo há uma interação com o usuário já dentro da etapa de parametrização.

A fase de parametrização consiste no terceiro passo do desenvolvimento. Dentro de uma abordagem mais automatizada, comensalista, o código recebe a classificação obtida e aplica algoritmos procedurais diferentes segundo a função do ambiente analisado. Se o código identifica, por exemplo, que um cômodo se parece com um dormitório, ele aplicará as regras e mobiliário de um dormitório, de modo semelhante ocorreria para banheiro, sala ou cozinha. Além disso, é feita uma subclassificação destes ambientes, como dormitório de solteiro, de casal ou suíte, ou mesmo entre um banheiro ou um lavabo. Após inserir o layout, o código identifica se há conflitos. Caso haja alguma sobreposição entre os mobiliários ou conflito com as aberturas, ele deve rearranjar o que foi feito até encontrar uma solução aceitável. Um outro caminho seria os arquitetos intervirem nesta etapa, resultando em uma protocooperação. Nesta abordagem, os usuários determinam qual a função de cada ambiente e entram com as características no conjunto, por exemplo, uma ou duas camas de solteiro, cama de casal, dimensões de mobiliário etc. Em seguida, o código recebe estes inputs, gera o layout, detecta os conflitos e, caso chegue a uma solução aceitável caberá aos arquitetos aceitarem a solução gerada ou não. Caberá a eles, portanto, finalizar o processo ou repetir o ciclo.

Na sequência, as informações geradas passam por uma fase de armazenamento. Aqui estão inclusos todos os dados importantes para a aplicação das instâncias de mobiliário na etapa seguinte, como: posicionamento, rotação e dimensão. Elas são estruturadas em listas indexadas e podem alimentar um banco de dados.

Por fim, a fase de instanciação consiste em aplicar filtros para facilitar a visualização e instâncias de mobiliário com a linguagem BIM.

Figura 50 – Procedimento para o modelo comensalista desenvolvido para esta dissertação



Fonte: elaborado pelo autor

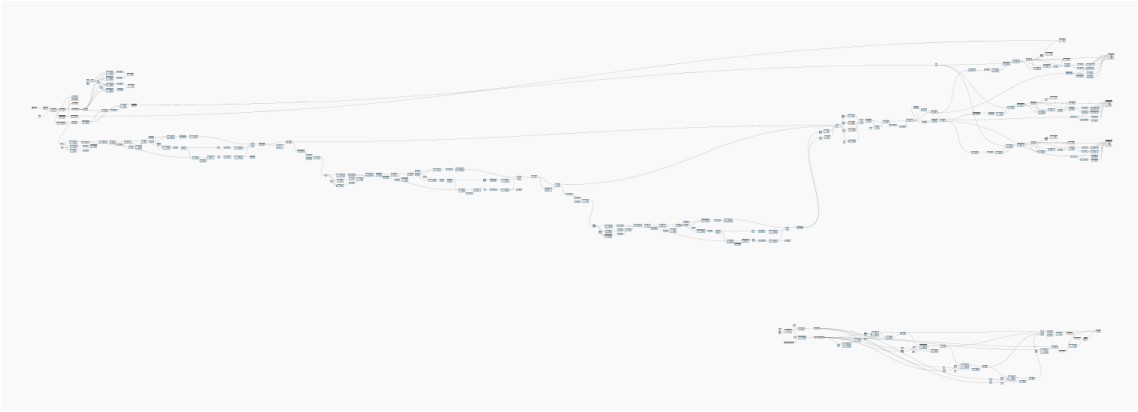
4.2 Instanciações

Instanciações foram realizadas como prova de conceito do modelo de protocooperação, prevendo que ele poderia evoluir em trabalhos futuros até um modelo mutualístico. Elas estão descritas segundo a sua ordem de criação. Destaca-se o desenvolvimento incremental e a característica exploratória dos artefatos.

4.2.1 Artefato 1[16]

A primeira instanciação foi feita com o Dynamo e componentes (nós) dos pacotes Clockwork e Rhythm. Ela consistiu na geração de layout de um dormitório utilizando uma cama de solteiro ou de casal e um armário. O usuário deve ter modelado um cômodo retangular com apenas uma porta e uma janela. Ao executar o programa pelo ambiente do Dynamo ou pelo Reprodutor do Dynamo (pelo qual o código é executado por um menu de interface no próprio Revit) é solicitado que se faça uma seleção ao redor dos elementos que serão analisados. Apesar de entrarem no programa diversos elementos selecionados pelo usuário, a categoria "ambientes" foi deixada pré-selecionada no código. Em seguida, uma operação de intersecção foi criada entre as listas, filtrando apenas os ambientes. Como último passo da fase de leitura, linhas equivalentes as bordas deles foram extraídas para análises geométricas nas próximas etapas. O desenvolvimento pode ser visto na Figura 51 e Figura 52 de modo esquemático.

Figura 51 – Visão geral do algoritmo em Dynamo



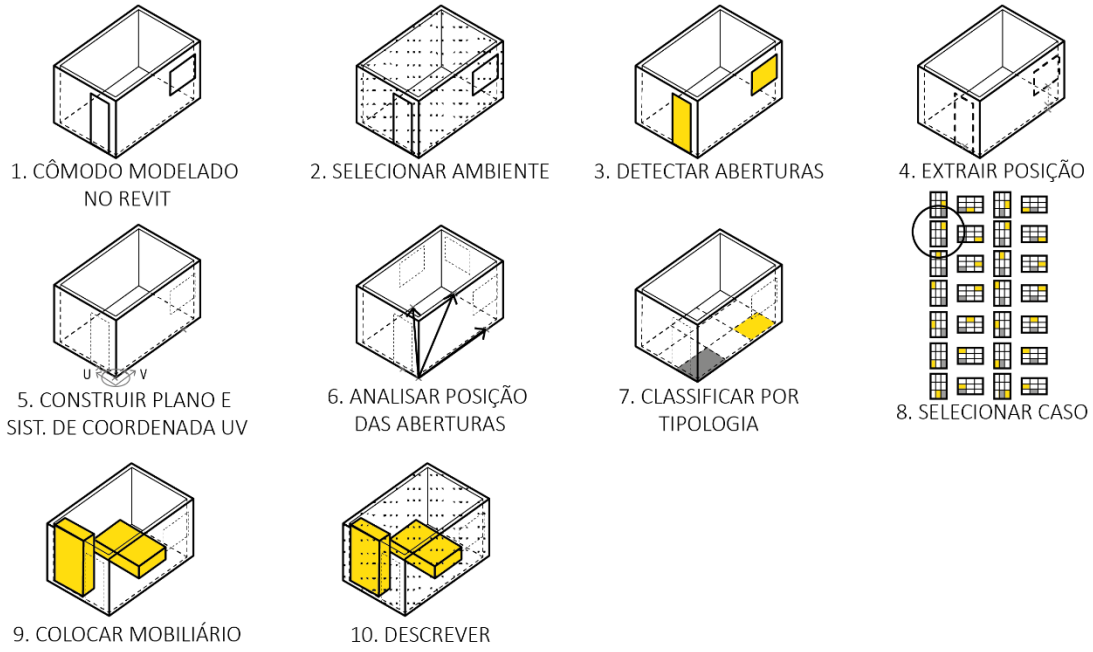
Fonte: elaborado pelo autor

Já na etapa de análise, as portas e janelas foram facilmente identificadas em cada ambiente por meio de nós que interpretam a quais ambientes pertencem estas aberturas. Na

[16] O artefato pode ser visto em funcionamento em: https://www.youtube.com/watch?v=GuhNEgkGxII

sequência, pontos relativos às suas localizações foram extraídos e, no caso das janelas, projetados no plano XY. Deste modo, todos os pontos que marcassem o posicionamento das aberturas estariam no mesmo plano, facilitando as análises futuras.

Figura 52 - Aplicação do layout desenvolvido na primeira instanciação



Fonte: elaborado pelo autor

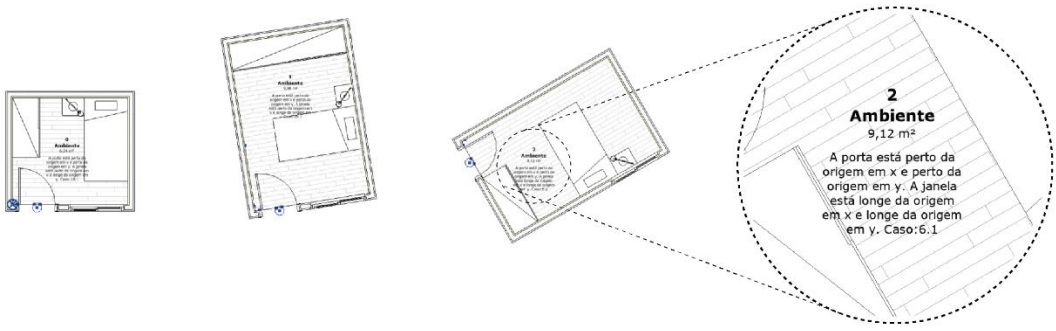
A próxima etapa foi estabelecer um novo plano UV com objetivo de inserir o layout independentemente da orientação do cômodo. O primeiro passo foi organizar as curvas do perímetro pela distância da porta e identificar qual ponto das extremidades da curva era mais próximo dela. Encontrado esse ponto em questão, um círculo foi inserido de modo que, diante da sua intersecção com as curvas de perímetro, o vetor U fosse criado entre a porta e a origem, já o vetor V entre a intersecção do círculo e a curva de perímetro, perpendicularmente a porta. O plano criado foi convertido em um novo sistema de coordenadas e, após uma análise das arestas, as dimensões de U e de V foram extraídas, obtendo os parâmetros de largura e comprimento.

Na sequência, o posicionamento das portas e das janelas foram analisados com o objetivo de classificar a tipologia dos cômodos. Para cada uma das aberturas foram realizadas operações trigonométricas, resultando em três listas de informações. A primeira delas classifica se a abertura está perto ou longe da origem dos eixos U e V. Já a segunda apresenta a distância em U e a última lista a distância em V. Essa classificação foi diretamente

relacionada com a colocação do mobiliário visto que direciona qual estratégia de disposição de elementos deveria ser utilizada.

Assim, ao início da etapa de parametrização, três tipos de famílias deveriam ser selecionados para integrarem o ambiente. No caso, se pré-fixou uma cama de solteiro, uma de casal e um guarda-roupa. É importante destacar que, caso o usuário estivesse utilizando o código pelo Reprodutor do Dynamo, seria possível controlar esses valores de entrada por lá, selecionando os tipos de família logo após a seleção dos ambientes a serem trabalhados. Feita a seleção do mobiliário, os parâmetros de largura e comprimento de cada um deles foram extraídos e utilizados na geração do layout. Deste modo, para cada combinação de posição de janela e de porta foram estabelecidas as posições dos elementos por meio de regras descritas em Python. Ao final, foram extraídas as posições, as novas larguras, os novos comprimentos e os ângulos de rotação que alimentariam os componentes que fazem a inserção de famílias em pontos na etapa de instanciação. No caso da rotação, não foi possível aplicar essa transformação como um comando. Percebeu-se ao longo do desenvolvimento que os valores utilizados em simulações anteriores ficavam armazenados dentro do componente, assim, a cada transformação somava-se o valor antigo. Para contornar essa limitação, as famílias utilizadas foram previamente modificadas, criando um parâmetro de rotação por instância de modo que, ao aplicar um novo valor para elas, as famílias seriam rotacionadas não no projeto, mas dentro de uma pré-configuração dentro delas mesmas. Por fim, os ambientes foram renumerados e foram acrescentados comentários automaticamente com a tipologia do ambiente no Revit (Figura 53).

Figura 53 – Layout gerado para qualquer orientação do cômodo e aplicação de comentário



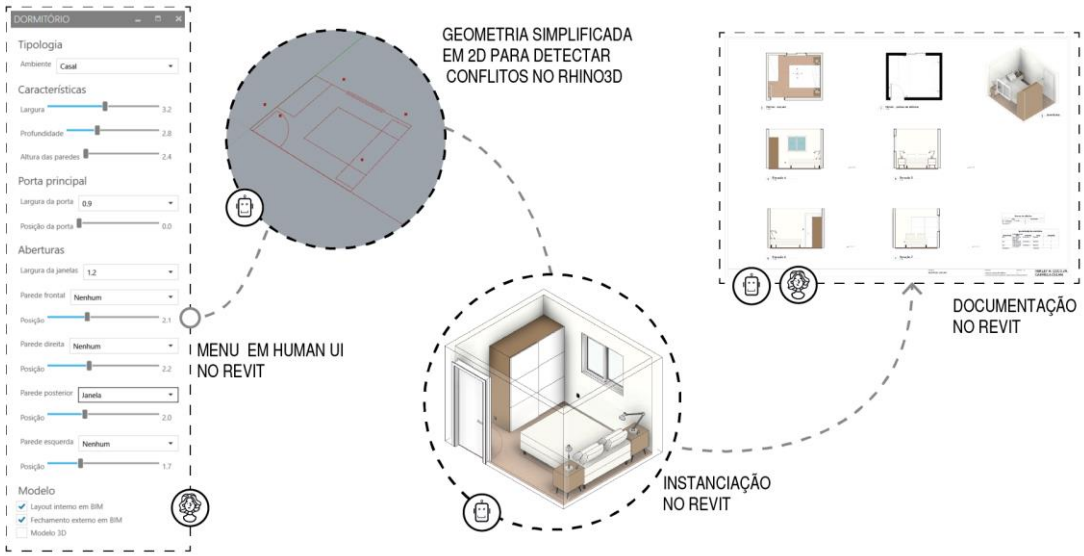
Fonte: elaborado pelo autor

4.2.2 Artefato 2[17]

Esta segunda instanciação visou a criação de um sistema de geração tanto arquitetônico quanto de mobiliário. O Revit 2021 foi utilizado em conjunto com o Rhino.Inside.Revit para a implementação.

A fase de leitura é iniciada pelo usuário que deve inserir os valores das variáveis importantes para o sistema previamente estabelecidas. Para tanto, um menu foi criado por meio do *Add-on* Human UI (Figura 54), sendo possível executar o código diretamente do ambiente do Revit, sem precisar abri-lo no Grasshopper. Em primeiro lugar, o usuário deveria determinar qual a tipologia de ambiente que seria criado, selecionando entre "solteiro" ou "casal". Em seguida, estabelecer as características dos ambientes, isto é, a "largura", a "profundidade" e a "altura das paredes". Essa valoração aconteceu por meio de *sliders* com intervalos específicos. O próximo passo foi determinar as características da porta principal, inserindo sua largura e posição ao longo da parede (neste caso, foi convencionado que a porta principal sempre ficasse na parede inferior). A quarta etapa foi escolher se as aberturas secundárias seriam portas ou janelas e qual as suas posições nas paredes em que se encontrariam aninhadas. Por fim, o usuário poderia escolher visualizar o "layout interno em BIM", o "fechamento externo em BIM" ou simplesmente "o modelo 3D". Esses parâmetros alimentariam todo o sistema.

Figura 54 – Menu, passagem pelo CAD, instanciação em BIM e desenho da prancha



Fonte: elaborado pelo autor

[17] O artefato pode ser visto em funcionamento em: https://www.youtube.com/watch?v=mIOSR80A8W4

No caso deste algoritmo, a etapa de análise também pode ser entendida como uma construção das geometrias e das relações paramétricas entre as variáveis. A primeira etapa foi criar uma *polyline* que marcaria o perímetro do ambiente. Logo em seguida, as geometrias de abertura foram desenhadas por meio de cinco *clusters* de códigos. Enquanto o algoritmo principal pode ser entendido como presente na camada 0, estes estariam na camada 1.1. Para a porta principal, um algoritmo foi desenvolvido para construir a geometria e mantê-la dentro dos limites da parede, ou seja, se o usuário atribuísse a posição da porta para um lugar que entre em conflito com as paredes, automaticamente ela seria deslocada para dentro dos limites desejáveis. Por fim, ela pode ser espelhada de modo que a sua abertura sempre se dê para uma das paredes laterais. Em essência, a sequência de procedimentos foi similar para as janelas: construção de geometrias e deslocamento da janela caso ela fosse colocada em um local inaceitável. No caso de uma eventual segunda abertura na mesma parede que a porta, o código interpretaria se há algum conflito entre as duas geometrias e, caso existisse, deslocaria a segunda.

Após essa fase de análise, seguiu a fase de parametrização com o desenho do mobiliário por meio de um componente em Python. Seis funções foram criadas no início do código: posição, dimensão, orientação, mobiliário (que reúne as três primeiras), ponto de elétrica e de ambiente. Essa última, em especial, teria como entrada a largura, profundidade e posição das aberturas, retornando se o ambiente seria largo ou profundo, pequeno, médio ou grande. Na sequência, analisaria ainda a posição das portas e das janelas, como por exemplo: "porta na esquerda, janela na esquerda e na direita", "porta na esquerda, nenhuma restrição para as aberturas" ou porta "na direita, janela na esquerda". Essas foram análises importantes para classificar a tipologia do ambiente em questão pela sua forma e posicionamento das aberturas. Neste momento, medidas gerais foram atribuídas para os mobiliários, são elas: largura, altura e profundidade da cama, largura da mesa de cabeceira e profundidade do guarda-roupa. Então, doze casos foram estabelecidos segundo a tipologia (largo ou profundo), tamanho (pequeno, médio ou grande) e posição da porta principal (esquerda ou direita). Para cada um deles, variáveis foram criadas de posição, dimensão e rotação para os equipamentos a serem inseridos. Na fase final, para a tipologia de casal, foram separados os doze casos definidos anteriormente e dentro de cada uma delas, cinco condições para disposição de mobiliário dependendo da posição da porta e das janelas.

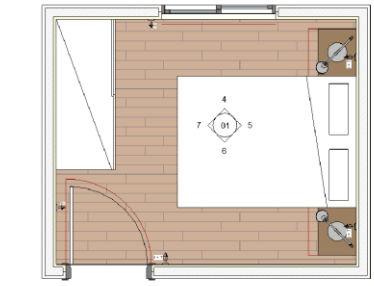
Os parâmetros obtidos no componente em Python foram importados em um *cluster* (camada 1.2) para desenho das geometrias em si. Na sequência, conflitos entre os elementos foram detectados por meio da intersecção de pelo menos duas curvas das geometrias criadas.

Feito isso, as geometrias passaram por uma estrutura semelhante à do Python descrita anteriormente, porém acrescentando a condicional caso existisse um conflito entre a cama e porta (outros conflitos também poderiam ser levados em consideração). Novamente, o código passou por um processo de análise de conflitos onde poderia existir a substituição de um equipamento por outro, por exemplo, uma cama com guarda-roupa embutido poderia ser trocada por uma cama normal. Por fim, esses resultados são armazenados em uma lista de dados estruturada em galhos.

Feito isso, ocorre o processo de instanciação no qual as curvas do perímetro recebem paredes, pontos recebem janelas, portas, mobiliário e tomadas. Importante ressaltar que as tomadas são famílias aninhadas em paredes (como ocorre com as famílias de aberturas), assim não é necessário entrar com parâmetros de rotação, como ocorre para um mobiliário, mas qual parede seria a hospedeira da família. Outro ponto a ser destacado é como ocorre a instanciação das famílias de mobiliário. O componente *Add family instance* tem como entrada a localização, o tipo da família, o nível e o hospedeiro. No caso do código desenvolvido, utilizou-se apenas os dois primeiros parâmetros para colocação do mobiliário. Na sequência, os parâmetros de largura, de profundidade e de rotação receberam novos valores segundo aqueles obtidos na fase anterior. Por uma limitação do Rhino.Inside.Revit, que ainda se encontra em desenvolvimento, a rotação teve que entrar como um parâmetro de instância na família e não como um comando, como usualmente se faz. Dessa forma, as famílias utilizadas nesta dissertação tiveram que ser preparadas com antecedência à semelhança da instanciação feita no Dynamo. Esse procedimento foi descrito no capítulo do Artefato 1.

Ao retornar para o Revit, ambiente no qual o usuário está desenvolvendo o projeto, o arquivo foi preparado para receber as soluções geradas pelo programa. Isto é, definiu-se elevações internas preposicionadas em planta, do mesmo modo tabelas para os pontos de elétrica e de mobiliário foram criadas e pré-configurações nas plantas foram estabelecidas para mostrar de diferentes maneiras uma planta, ora com o layout ora com os pontos de elétrica (Figura 55). Todas essas informações foram reunidas em uma prancha ao final.

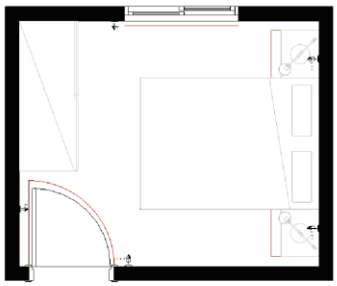
Figura 55 – Três diferentes resultados: layout arquitetônico, planta de elétrica e tabela com quantitativos



1 Térreo - Layout
1 : 25

Quantidade de mobiliário

Marca de tipo	Comentários de tipos	Fabricante	Custo	Quantidade
GU2	Guarda-roupa para cama embutida	Fabricante B	4000,00	1
CA2	Cama de casal 1380x1890 com criados-mudos	Fabricante A	5000,00	1
CA3	Cama de casal 1380x1880	Fabricante A	4000,00	1
Total geral: 3			13000,00	3



2 Térreo - pontos de elétrica
1 : 25

Pontos de elétrica

Tipo	Quantidade
01 - 1 Interruptor - 2 Tomadas	3
08 - 3 Tomadas	2
Total geral: 5	5

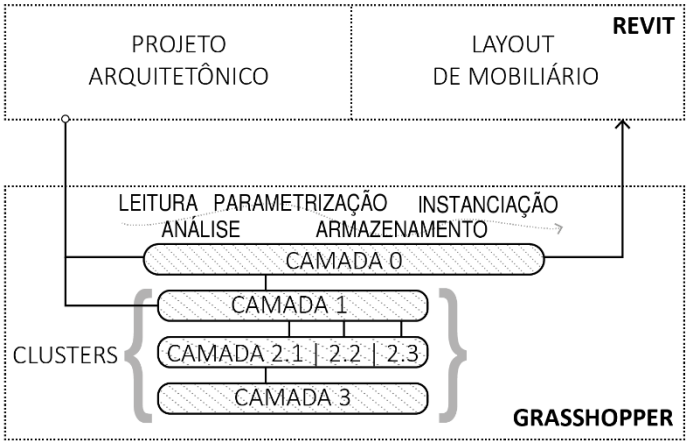
Fonte: elaborado pelo autor

4.2.3 Artefato 3[18]

Este artefato teve como objetivo inserir o mobiliário em um dormitório a partir de paredes, portas e janelas que já formavam um ou mais ambientes. O Revit 2021 foi utilizado como programa BIM e o Rhino.Inside.Revit como ponte entre ele e o Grasshopper, no qual foi realizada a criação dos algoritmos, seguindo a abordagem comensalista apresentada anteriormente. Antes de tudo é importante ressaltar que o código também foi estruturado por meio de *clusters* a fim de facilitar o desenvolvimento do algoritmo. Isso é como se existissem camadas de código. Assim, a camada mais externa será a chamada de 0, a seguinte de 1 e assim por diante (Figura 56).

[18] O artefato pode ser visto em funcionamento em: https://www.youtube.com/watch?v=CghS4jHA1j8

Figura 56 – *Workflow* do terceiro artefato com destaque para as camadas de desenvolvimento



Fonte: elaborado pelo autor

Na fase de leitura, o código foi iniciado por meio da seleção de todos os ambientes existentes no projeto arquitetônico. Para isso, inseriu-se um componente que varreu as categorias BIM e um filtro que extraiu uma lista de todos os ambientes criados no modelo. Em seguida, sua quantidade foi obtida, servindo como contador para o *looping* que viria na sequência. Essa estrutura foi construída por meio do *Add-on* chamado Anemone, no qual é possível realizar *loopings* mesmo com uma Linguagem de Programação Visual. O grande benefício deste procedimento foi a possibilidade de criar um código e depois aplicá-lo para quantos casos fossem necessários, otimizando o processo de criação e futura manutenção. O procedimento de leitura continuou dentro do *looping* e em uma camada inferior, a camada 1. Na sequência, todos os ambientes foram selecionados e convertidos em *boundary representations* (*Breps*). Um item dessa lista foi selecionado (item esse relativo ao contador do *looping*) e uma curva equivalente ao perímetro da geometria foi obtida. Isso foi feito por meio da seleção do seu centroide, da colocação de um plano horizontal neste ponto, do encontro da intersecção entre o volume e o plano, da projeção das curvas no plano de origem XY e, por fim, da sua conversão em *polyline*.

Paralelamente a esse processo, os elementos dentro das categorias de parede, porta e janela foram identificados de maneira semelhante ao realizado com os ambientes. Um ponto importante, porém, seria distinguir quais as relações topológicas entre eles, ou seja, dentre todas as paredes, portas e janelas presentes no modelo, o programa deveria selecionar apenas as que pertencessem a determinado ambiente. Assim, iniciou-se a etapa de análise. Um componente do próprio Grasshopper foi utilizado o qual identifica conflitos entre

geometrias. A partir disso, separou-se os elementos que "tocam" um ambiente (segundo o contador do *looping)*. Por fim, portas e janelas foram convertidos em *breps* também.

Neste momento, foi necessário extrair informações de posicionamento, largura, altura e peitoril das portas e janelas. Isso foi realizado por meio de mais um *cluster*, uma camada 2.1. Ele recebeu os *inputs* das geometrias e da curva de perímetro do ambiente descrita anteriormente. Em essência, o algoritmo agrupou os diversos componentes de uma porta em uma única geometria na qual foi base para a criação de uma *bounding box*. Essa nova geometria foi desconstruída e as suas faces foram reorganizadas segundo a área, a maior delas foi selecionada e foram extraídas as dimensões da curva menor e da maior, largura e altura nominal da porta e da janela. Para obtenção do ponto relativo ao posicionamento do elemento, o centroide foi extraído da *bounding box* e projetado sobre o ponto mais próximo pertencente a *polyline* construída anteriormente. Ao final, estas informações foram indexadas e estruturadas em galhos.

A etapa seguinte consistiu em estabelecer um novo sistema de coordenadas UV e identificar os vetores de largura e de comprimento do ambiente. O primeiro passo foi determinar o ponto de origem. Inicialmente, as curvas que formavam o perímetro foram organizadas segundo a distância da porta com maior largura (definida como acesso principal). A curva mais próxima, então, foi selecionada e os pontos extremos foram obtidos. Aquele que estiver mais próximo do ponto da porta seria definido como a origem. Um círculo foi inserido nesta posição, obtendo uma intersecção entre a curva mais próxima da porta e a curva perpendicular a ela. Na sequência, diversas operações com vetores foram realizadas, visando garantir que eles tivessem sempre a orientação voltada para o interior do ambiente e perpendiculares entre si. Isso foi feito por meio do estabelecimento do ponto de origem e criação de vetores entre ele e os pontos de intersecção. Esta etapa foi concluída com a extração do vetor U e V.

Feita essa construção de vetores, três operações foram realizadas novamente com as portas e as janelas. A primeira delas foi identificar qual a posição relativa das aberturas em U e em V. Isso foi feito pela criação de um vetor entre o ponto de origem do ambiente e da abertura seguida da sua projeção nos planos UZ e VZ. Na segunda operação, as aberturas foram classificadas segundo a posição das paredes, ou seja, se o elemento estivesse na parede de origem seria indexado como 0, na oposta como 1, na frente como 2 e na adjacente como 3. A terceira e última operação teve como objetivo criar o volume de abertura à frente de cada porta e janela, visando a detecção de conflitos na etapa de colocação de mobiliário. A

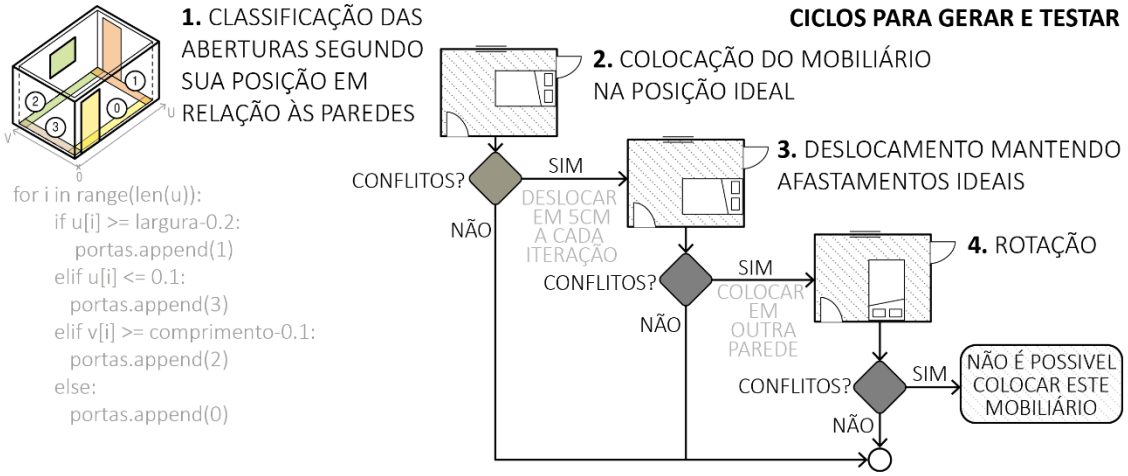
principal questão aqui foi manter essa "zona de circulação" dentro dos ambientes e alinhada com as faces adjacentes. Isso foi implementado por meio de uma sequência de operações geométricas dentro de um cluster, camada 2.2. Inicialmente, retângulos foram criados tomando como referência o ponto de intersecção dos elementos e a sua largura. Em seguida, uma análise foi feita para cada um deles na camada 3, verificando se os pontos dos cantos dos retângulos estariam ou não dentro do ambiente. Se existissem pontos fora, seria aplicada uma rotação de 90º até ele ficar na posição aceitável. Depois disso, as geometrias integraram uma lista e os retângulos iniciais foram extrudados, criando um volume referente a circulação.

Esses procedimentos de leitura e análise descritos até aqui seriam repetidos para cada um dos subambientes criados a partir de regras de subdivisão caso o cômodo tivesse um formato em "L". A subdivisão de um ambiente se desenvolveu a partir da curva de perímetro, vetores U e V e ponto de inserção das portas em um *cluster* na camada 2.3. Uma superfície foi criada, usando a *polyline* do perímetro. Em seguida, para cada ponto de canto foi criado uma sequência de linhas em U e V suficientemente grandes para cortar a superfície originando diversas superfícies retangulares menores. Essas, por sua vez, foram combinadas holisticamente, formando superfícies maiores. Ao final deste procedimento essas superfícies formaram uma lista por ordem de tamanho. De volta à camada 1, uma destas superfícies foi selecionada (pela maior área nesta dissertação ou por outra regra desejada). Ao término, esta geometria foi extrudada, repetindo o ciclo explicado anteriormente.

Todo o desenvolvimento no *cluster* da camada 1 culminou em um pacote de dados com as características do ambiente já citadas anteriormente, além do desenho em planta de um layout preliminar com o objetivo de detectar conflitos entre eles mesmos e entre as aberturas. Sendo que este último foi desenvolvido em Python e representou a transição entre a etapa de análise e de parametrização. No início, definições foram criadas para desenhar formas simples. Essas funções tiveram como entrada as coordenadas de localização, dimensões e orientação e retornaram uma lista com o ponto de inserção, largura, comprimento, altura, ângulo de rotação, curvas referentes às geometrias em si, a distância em U em V para cada um dos mobiliários desejados - dez parâmetros ao total. As funções foram chamadas dentro de um *looping* de atribuição de valores. Assim, para cada posição relativa das portas, isto é, aninhada na parede 0, 1, 2 ou 3 foram criadas condicionais para o mobiliário ser colocado na melhor posição. Caso não houvesse conflitos, o *looping* seria encerrado. Caso o elemento interseccione outro elemento, o ciclo se repetiria, porém, somando 1 ao contador e um fator de deslocamento "j" de 0.05m de deslocamento. Assim, o mobiliário poderia se

deslocar em U ou em V dependendo das posições das portas. Se nenhuma solução for encontrada, o mobiliário é transferido para uma outra parede (Figura 57).

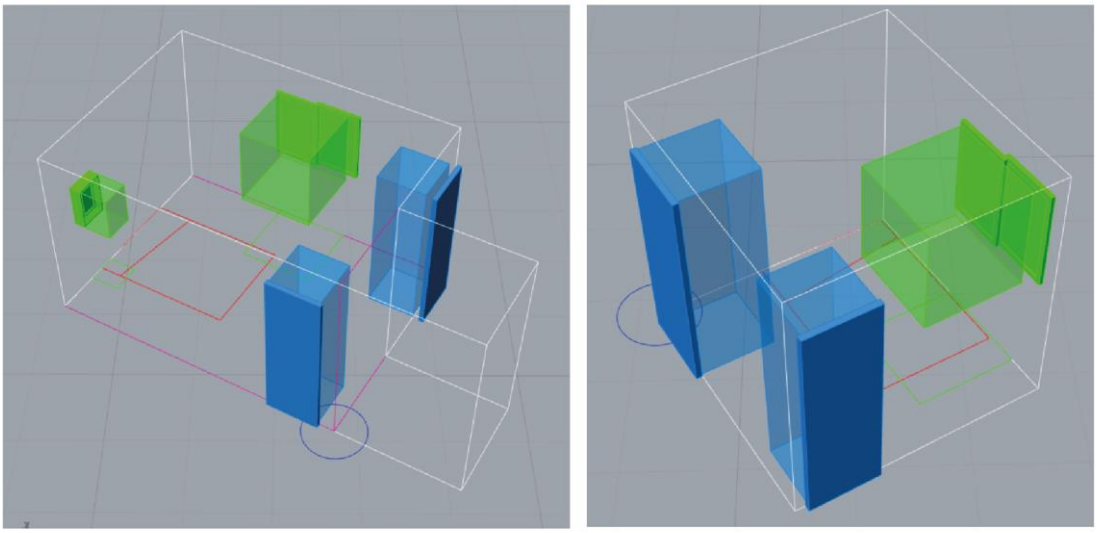
Figura 57 – Ciclos para gerar e testar alternativas de layouts



Fonte: elaborado pelo autor

De volta à camada 0, estes dados foram armazenados e estruturados em galhos. Já encaminhando para o final, na etapa de instanciação, por sua vez, os dados se bifurcaram. Por um lado, foi realizada uma representação em cores das geometrias criadas dentro do ambiente do Rhinoceros (Figura 58). Por outro lado, o resultado das operações forneceu as informações necessárias para a inserção das famílias do ambiente do Revit.

Figura 58 – Análises realizadas em ambientes em "L" e retangular



Fonte: elaborado pelo autor

4.2.4 Artefato 4[19]

O artefato 4 teve como objetivos aumentar a colaboração entre arquitetos e máquinas em uma orientação de protocooperação. Assim como o Artefato 3, foi utilizado o Rhino.Inside.Revit para criação do código.

Inicialmente, na fase de leitura, todos os ambientes presentes no projeto do Revit são importados para o Grasshopper e filtrados pelo uso que se deseja trabalhar. No caso desta dissertação, são sempre dormitórios. Esta seleção é realizada heuristicamente , ou seja, os ambientes dentro do Revit foram classificados manualmente com o uso "DORMITÓRIO" e o sistema em Grasshopper reconhece essa identificação e encaminha os dados para a inserção do layout. Na sequência, pela mesma lógica, são separados os dormitórios principais, que são dormitórios de casal, e os secundários que são dormitórios de solteiro. Feito isso, abre-se uma estrutura de *looping* e é analisado ambiente por ambiente do modelo.

Na etapa seguinte, são importadas todas as portas, janelas e paredes que estão inseridas no projeto. Com o objetivo de filtrar os elementos que pertencem ao cômodo que está sendo estudado, são selecionados aqueles que possuem algum conflito com o ambiente estudado. Por exemplo: se uma parede é adjacente ao volume do ambiente, então ela deve ser selecionada, caso contrário deve ser ignorado. Neste momento, é criado um sistema de coordenadas para o cômodo, determinando sua origem, área, curvas que o delimitam e os vetores x e y. Por fim, este ambiente é subdividido em ambientes menores, dos quais é possível extrair o espaço principal onde a cama será colocada, uma área de circulação e outra que receberá um armário, por exemplo. Assim, um ambiente de forma não retangular é simplificado em formas retangulares, potencializado o número de casos possíveis de aplicação. Além disso, as portas e janelas recebem a modelagem de volumes que representam o espaço necessário para a abertura e a área de circulação.

Neste momento, o algoritmo entra em uma fase crucial que é a classificação do ambiente e do subambiente (Figura 59). As seguintes classificações são feitas por meio de condicionais programadas em Python:

- Classificação do ambiente segundo a complexidade, sendo:
    - "Simples" se existir apenas 1 subambiente. Por exemplo: ambiente retangular;

[19] O artefato pode ser visto em funcionamento em: https://www.youtube.com/watch?v=PPV4LasUJvg

- De "média complexidade" se apresentar 2 subambientes. Por exemplo: ambiente em L ou closet;
- "Complexo" se apresentar mais que 2. Por exemplo: espaço para circulação, closet e área principal.

- Classificação do subambiente segundo a área:
  - Se a área for menor que 4m² é considerado pequeno;
  - Se a área estiver entre 4m² a 10m² é considerado médio;
  - Se a área for maior que 10m² é considerado grande.
- Classificação segundo as portas e passagens:
  - Se apresentar um único elemento, o subambiente é classificado como de acesso;
  - Se apresentar mais de um elemento, o subambiente é entendido como espaço de circulação.
- Classificação segundo as janelas, podendo ser:
  - De uma única janela;
  - De duas janelas;
  - Sem janelas.
- Classificação em tipologias:
  - Se o subambiente for pequeno, sem janela e com uma única porta ou acesso é classificado como espaço de apoio;
  - Se o subambiente for pequeno, sem janela, mas com duas portas é classificado como corredor;
  - Se as condições anteriores não forem atendidas, ele será o ambiente principal.
- Classificação segundo o posicionamento de portas e janelas:
  - Se as portas e janelas estão próximas da origem;
  - Se as portas e janelas estão em regiões centrais;
  - Se as portas e janelas estão em cantos afastados da origem.

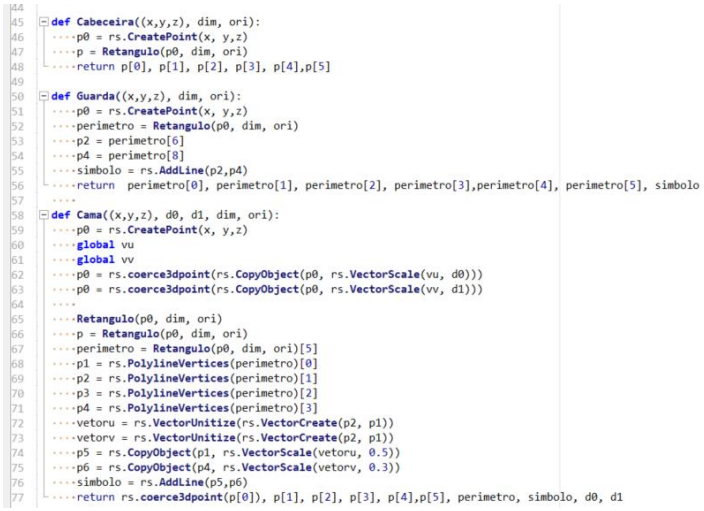
Figura 59 – Código em Python para classificação do ambiente e subambiente

```
import rhinoscriptsyntax as rs

#Classificação do ambiente segundo complexidade
if amb_quant_total == 1:
    forma = 0 #simples
elif amb_quant_total ==2:
    forma = 1 #média complexidade
else:
    forma = 2 #complexa


#Classificação do subambiente segundo a área:
if amb_area < 4:
    tamanho = 0 #pequeno
elif 4<=amb_area<=10:
    tamanho = 1 #médio
else:
    tamanho = 2 #grande


#Classificação segundo as porta e passagens
if len(porta_x) == 1:
    acesso = 1 # único
elif len(porta_x) >1:
    acesso = 2 #circulação
else:
    acesso = 0


#Classificação segundo as janela
if len(janela_x) == 1:
    janela = 1 # único
elif len(janela_x) ==2:
    janela = 2 #duas janela - ambiente principal
elif len(janela_x) > 2:
    janela = 3 #mais de duas janelas - ambiente principal
else: #sem janelas
    janela = 0 #apoio
```

Fonte: elaborado pelo autor

Após essa classificação, a parametrização é realizada em Python para a inserção do mobiliário. São necessários 13 *inputs* para o funcionamento do sistema: o ponto de origem, a curva que forma o perímetro, a classificação segundo a ocupação, tipologia e aberturas, vetor x e vetor y, comprimento em x e em y, tipologia das portas e janelas e, por fim, a geometria das aberturas. De início, utilizando Rhinoscript e a biblioteca ghphythonlib foram desenvolvidas definições que descrevem o mobiliário a ser inserido pelo seu ponto de inserção, dimensões e orientação (Figura 60). Na sequência, as variáveis do sistema são valoradas (Figura 61) e por fim ocorre a geração do layout.

Figura 60 – Definições em Python para a inserção do mobiliário



```
def Cabeceira((x,y,z), dim, ori):
    p0 = rs.CreatePoint(x, y,z)
    p = Retangulo(p0, dim, ori)
    return p[0], p[1], p[2], p[3], p[4],p[5]

def Guarda((x,y,z), dim, ori):
    p0 = rs.CreatePoint(x, y,z)
    perimetro = Retangulo(p0, dim, ori)
    p2 = perimetro[6]
    p4 = perimetro[8]
    simbolo = rs.AddLine(p2,p4)
    return  perimetro[0], perimetro[1], perimetro[2], perimetro[3],perimetro[4], perimetro[5], simbolo

def Cama((x,y,z), d0, d1, dim, ori):
    p0 = rs.CreatePoint(x, y,z)
    global vu
    global vv
    p0 = rs.coerce3dpoint(rs.CopyObject(p0, rs.VectorScale(vu, d0)))
    p0 = rs.coerce3dpoint(rs.CopyObject(p0, rs.VectorScale(vv, d1)))

    Retangulo(p0, dim, ori)
    p = Retangulo(p0, dim, ori)
    perimetro = Retangulo(p0, dim, ori)[5]
    p1 = rs.PolylineVertices(perimetro)[0]
    p2 = rs.PolylineVertices(perimetro)[1]
    p3 = rs.PolylineVertices(perimetro)[2]
    p4 = rs.PolylineVertices(perimetro)[3]
    vetoru = rs.VectorUnitize(rs.VectorCreate(p2, p1))
    vetorv = rs.VectorUnitize(rs.VectorCreate(p2, p1))
    p5 = rs.CopyObject(p1, rs.VectorScale(vetoru, 0.5))
    p6 = rs.CopyObject(p4, rs.VectorScale(vetorv, 0.3))
    simbolo = rs.AddLine(p5,p6)
    return rs.coerce3dpoint(p[0]), p[1], p[2], p[3], p[4],p[5], perimetro, simbolo, d0, d1
```

Fonte: elaborado pelo autor

Figura 61 – Valoração das variáveis em Python

```
camacasal_l = 1.38 + 2*0.5
camacasal_c = 1.88
camacasal_h = 0.5

camasolteiro_l = 0.88 + 2*0.5
camasolteiro_c = 1.88
camasolteiro_h = 0.5

camasolteiro2_l = 2.26
camasolteiro2_c = 1.905
camasolteiro2_h = 0.5

beliche_l = 0.88
beliche_c = 1.88
beliche_h = 2.2

guardaroupa_l = 2
guardaroupa_p = 0.6
guarda_roupa_h = 2.1

comoda_l = 1.5
comoda_p = 0.8
comoda_h = 0.9
```

Fonte: elaborado pelo autor

Neste último caso o algoritmo faz uma distinção inicial entre os ambientes simples, de média e de alta complexidade, além de identificar se o caso estudado é um corredor, área de circulação ou área principal. A partir de então são implementados *loopings* de até 100 iterações (k) para cada peça do mobiliário, garantindo que não haja conflitos com outras peças ou aberturas. Uma cama, por exemplo, é inserida em uma posição inicial ideal e o *looping* é interrompido se não houver conflitos. Caso contrário, ela irá para uma segunda posição e a cada iteração se moverá 1cm ao longo da parede hospedeira. Se K for maior que 10 e menor que 20, o mobiliário será colocado em uma outra parede. Caso ainda não haja uma solução, o ciclo se repetirá em mais uma tentativa (Figura 62). Ao concluir esta rotina para todas as peças do mobiliário, as informações são estruturadas em listas e devolvidas ao código principal, compondo a fase de armazenamento.

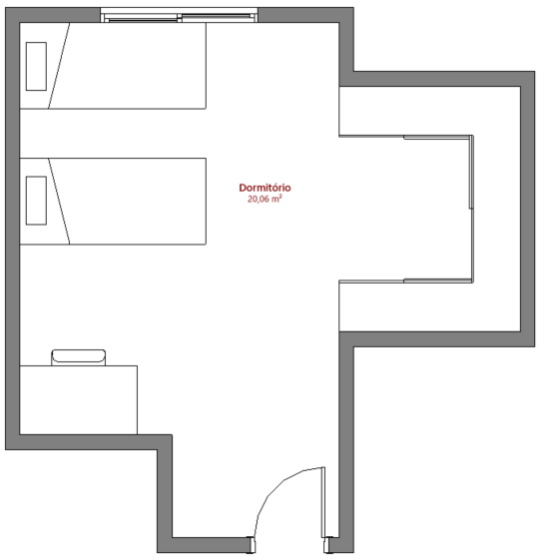
Por fim, na etapa de instanciação, os dados são representados por meio de geometrias dentro do ambiente do Rhinoceros e, simultaneamente, são inseridas famílias dentro do Revit. Neste último caso, o procedimento ocorre por meio de um componente que permite a locação de famílias em um ponto específico, seguido de outro componente para a rotação segundo o ângulo proveniente da etapa de parametrização. A Figura 63 traz um exemplo dos resultados e a Figura 64 do algoritmo no ambiente de Grasshopper.

Figura 62 – Parametrização do posicionamento do mobiliário em Python

```
elif tipologia ==2: #principal
    if ocupacao == 0: #casal
        if tp == 0 or tp == 1 or tp == 2 or tp ==3:
            k =0
            j = 0
            while k<100:
                ### cama
                if k ==0:
                    cama = Objeto(p,x, (y-(camacasal_l)/2), (camacasal_l,camacasal_c,camacasal_h),0)
                    dormir = mob_rot(cama)
                    dormir_volume = mob_extrude(dormir[1], camacasal_h)
                    dormir_clashes = clash(dormir_volume, aberturas)

                elif dormir_clashes == True and k< 10:
                    cama = Objeto(p,0, (y-(camacasal_l)/2)+j, (camacasal_l,camacasal_c,camacasal_h),0)
                    dormir = mob_rot(cama)
                    dormir_volume = mob_extrude(dormir[1], camacasal_h)
                    dormir_clashes = clash(dormir_volume, aberturas)

                elif dormir_clashes == True and 10<=k<20:
                    cama = Objeto(p,0, (camacasal_l/2)-j, (camacasal_l,camacasal_c,camacasal_h),0)
                    dormir = mob_rot(cama)
                    dormir_volume = mob_extrude(dormir[1], camacasal_h)
                    dormir_clashes = clash(dormir_volume, aberturas)

                elif dormir_clashes == True and 20<=k<30:
                    cama = Objeto(p,(x/2), y, (camacasal_l,camacasal_c,camacasal_h),0)
                    dormir = mob_rot(cama)
                    dormir_volume = mob_extrude(dormir[1], camacasal_h)
                    dormir_clashes = clash(dormir_volume, aberturas)
                print("k", k)

                situacaodormir = fora(perimetro, dormir)
                if situacaodormir == True and dormir_clashes == False:
                    break
                k+=1
                j+=0.01
```

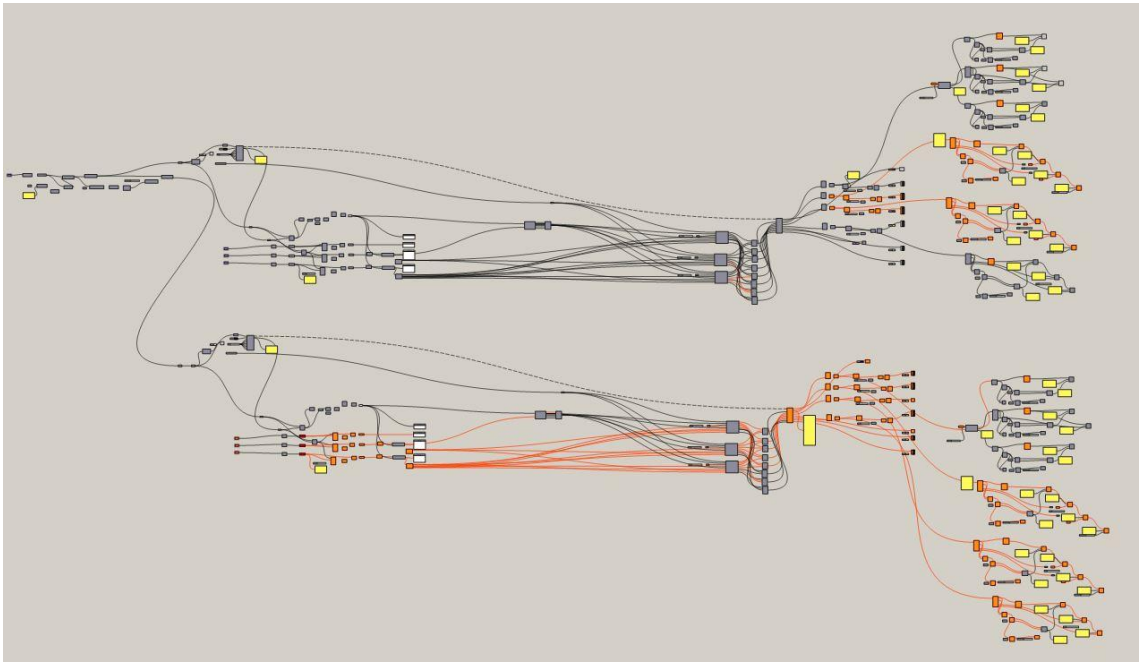
Fonte: elaborado pelo autor

Figura 63 – Exemplo de modelo gerado pelo artefato 4



Fonte: elaborado pelo autor

Figura 64 – Visão geral do algoritmo desenvolvido em Grasshopper.



Fonte: elaborado pelo autor

4.3. Considerações sobre as instanciações

Uma visão geral dos procedimentos e resultados dos artefatos pode ser vista na Tabela 5. Houve um desenvolvimento incremental e exploratório como prova de conceito da automatização da geração de layouts de mobiliário em BIM.

Tabela 5 – Resumo dos artefatos criados

		Artefato 1	**Artefato 2**	**Artefato 3**	**Artefato 4**
Tecnologia	**Software utilizado**	Revit, Dynamo	Revit, Grasshopper, Rhino.Inside.Revit	Revit, Grasshopper, Rhino.Inside.Revit	Revit, Grasshopper, Rhino.Inside.Revit
	Interface para o usuário	Reprodutor do Dynamo	Human UI	Nenhuma	Nenhuma
Entradas	**Forma prevista**	Retangular	Retangular	Retangular ou em L	Retangular, em L ou Complexa (hall, ambiente principal e closet)
	Prevê cômodo rotacionado?	Sim	Não	Sim	Não
	Quantidade de portas	Única	Até duas	Até duas	Ilimitado
	Quantidade de janelas	Única	Única	Única	Ilimitado
	Entrada manual de parâmetros de largura, comprimento e altura do dormitório	Não	Sim	Não	Não
Métodos	**Método para classificação da tipologia**	Posição da porta e da janela	Posição das portas e da janela	Posição das portas e da janela	Posição das portas e da janela, área, quantidade de acessos

	Método para parametrização	Função matemática com uma única solução para cada parametrização	Função matemática com uma única solução para cada parametrização	Função matemática com indexação das paredes, geração e teste para colocação de mobiliário, subdivisão de ambientes	Função matemática, geração e teste para colocação de mobiliário, subdivisão de ambientes
	Parâmetros de mobiliário para instanciação	Posição, dimensões, orientação	Posição, dimensões, orientação	Posição	Posição e dimensões
	Aninhamento de objetos	Sim	Sim	Sim	Sim
	Detecção de conflitos	Não	Sim	Sim	Sim
Saídas	**Novo mobiliário**	Cama de casal, cama de solteiro e guarda-roupa	Cama de casal, cama de solteiro e guarda-roupa, guarda-roupa embutido, cômoda, tomadas e interruptores	Desenho em CAD representando uma cama de casal	Cama de casal, cama de solteiro, guarda-roupa e escrivaninha
	Novos elementos arquitetônicos (paredes, portas, janelas e pisos)	Não	Sim	Não	Não
	Documentação em pranchas e quantitativos	Não	Sim	Não	Não

Fonte: elaborado pelo autor

O último artefato conseguiu abranger dormitórios com maior complexidade de forma e de função, isto é, foi possível prever espaços de apoio (como hall e closet), a circulação entre eles e duas ou mais portas e janelas. Em relação aos artefatos anteriores, um algoritmo mais enxuto, robusto e que pode abranger mais casos foi desenvolvido por meio da implementação de *loopings* e condicionais implementadas em *Python*. Em relação a uma metodologia que utiliza uma função matemática determinística (como o Artefato 1), considera-se o sistema de geração e teste uma melhor alternativa para trabalhos futuros. Neste caso, por exemplo, determinada mobília pode ser inserida na posição, dimensão e orientação ideal; caso haja algum conflito, porém, ele pode ser deslocado ao longo da parede hospedeira ou mesmo trocar de posição para uma outra parede.

Destaca-se também o Artefato 2 que, diferente dos outros três, possui uma implementação de dentro para fora, isto é, a partir do layout de mobiliário gerado segundo *inputs* dos usuários são modeladas automaticamente paredes, pisos, portas e janelas. Além disso, este modelo apresentou uma maior diversidade de elementos, como tomadas e interruptores e a substituição de um guarda-roupa, por exemplo, por um armário embutido ou cômoda dependendo da área do ambiente ou conflitos com aberturas. Por fim, o material gerado é organizado em uma prancha o que se mostra bem interessante na diminuição do tempo gasto entre concepção e projeto executivo, abrindo caminhos para maior exploração em outros trabalhos.

Os artefatos 1 e 3, por sua vez, trataram de desenvolvimentos preliminares para o segundo e o quarto. Sendo o primeiro o mais elementar de todos e o terceiro uma primeira experimentação para criar um algoritmo de geração e teste para abranger mais tipologias de dormitórios. Uma especificidade do artefato 1, porém, é a sua implementação por meio do Dynamo. A vantagem desta aplicação é não necessitar de outros softwares para a automatização, ficando no mesmo ambiente do Revit. Isso permite acessar diretamente a semântica do software e construir algoritmos sobre ela. Entretanto, isto também pode ser uma desvantagem, visto que há um maior gasto de processamento computacional e uma limitação de bibliotecas em relação ao que existe no momento para o Grasshopper. Em síntese, tanto o Dynamo tanto o Grasshopper (junto com o Rhino.Inside.Revit) podem responder bem às demandas de automatização, sendo que o Grasshopper se destaca pela flexibilidade, quantidade de bibliotecas e interoperabilidade com outros softwares BIM como o VisualArq e Archicad.

Os artefatos desenvolvidos devem ser vistos como complementares mesmo diante de seu progressivo refinamento. Considerando a metodologia de geração de layouts, o *workflow* desenvolvido, os modelos gerados e os resultados de documentação, trabalhos futuros podem utilizá-los como base para aplicação de um sistema de inteligência artificial, como *machine learning*, com o objetivo de se chegar a uma relação mutualística entre máquinas e projetistas. Para esta dissertação, o artefato 4 foi submetido à uma avaliação de desempenho, conforme relata o capítulo seguinte.

5 AVALIAÇÃO

A avaliação do desempenho do artefato final foi realizada por meio de um experimento inspirado nos trabalhos de Merrell *et al.* (2011), Kán e Kaufmann (2017) e Turing (1950), nos quais é medido o desempenho de máquinas exercendo ações humanas. Neste experimento, entretanto, buscou-se analisar o arranjo de mobiliário automatizado segundo o critério de funcionalidade em uma avaliação qualitativa dos resultados.

5.1 Objetivo

O principal objetivo foi verificar qual o nível de colaboração que o sistema desenvolvido permite entre arquiteto e máquina na fase de concepção projetual.

5.2 Hipóteses iniciais

As hipóteses iniciais foram:

- A geração automatizada equivalerá em qualidade à geração obtida manualmente, segundo critério de funcionalidade, isto é, de uma boa distribuição dos elementos no espaço (NEUFERT, 2012; PANERO; ZELNIK, 2016);
- A geração automatizada permitirá que se leve em consideração alternativas diferentes daquelas encontradas nas plantas utilizadas para o teste.

5.3 Procedimentos

A seguir se faz uma descrição das bases de dados utilizadas para o trabalho e em seguida a aplicação do algoritmo nesta base.

5.3.1. Base de dados

Um levantamento de plantas de apartamentos foi realizado como subsídio para a avaliação do algoritmo proposto. Inicialmente, as maiores construtoras nacionais foram listadas, em seguida seis delas foram selecionadas. Esse recorte foi feito em razão dessas empresas disponibilizarem em seus sites plantas de apartamentos para seus novos empreendimentos, bem como por cobrirem um intervalo de diferentes faixas de renda e cidades de atuação com o objetivo de cobrir possíveis diferenças culturais no layout.

As plantas foram modeladas no Revit 2021 a partir das imagens encontradas. Quando a escala ou as dimensões não estavam disponíveis, buscou-se ajustar a escala por meio de objetos de medidas conhecidas, como as camas. Foram modeladas 151 plantas como mostram a Tabela 6, a Figura 65 e a Figura 66.

Tabela 6 – Plantas de apartamentos modelados divididos em construtora e cidade

Construtora	Belém	Brasília	Campinas	Manaus	Porto Alegre	Recife	Ribeirão Preto	Rio de Janeiro	São Luis	São Paulo	Total
A							12				12
B	8							33	6	18	65
C		3		5	11	7		12		6	44
D			10							3	13
E										4	4
F										13	13
Total	8	3	10	5	11	7	12	45	6	44	151

Fonte: elaborado pelo autor

Cada uma das plantas recebeu um código de identificação incluindo a empresa, o local, o nome do empreendimento e o número da sequência da unidade. Assim, um apartamento projetado pela construtora A no Rio de Janeiro seria descrito como "A_RIO DE JANEIRO_EMPREENDIMENTO X_001". Em seguida, os apartamentos foram modelados em BIM (utilizando-se o software Revit) de modo simplificado, como seria feito nas etapas iniciais de um projeto. Foram utilizados dois tipos de paredes (uma de alvenaria de 15cm e outro drywall de 7cm), indicando acabamentos internos e externos; uma porta simples de abrir e uma porta balcão; dois tipos de janelas, sendo uma delas com duas folhas e a outra maxim-ar para banheiros e, por fim, dois tipos de pisos (um cerâmico e outro laminado de madeira). Antes de finalizar a modelagem, os ambientes receberam uma numeração, uma nomenclatura segundo o uso (sala de estar, cozinha, área de serviço ou dormitório) e uma classificação (contendo o nome da empresa, o número do empreendimento e uma sigla com duas letras referente ao uso).

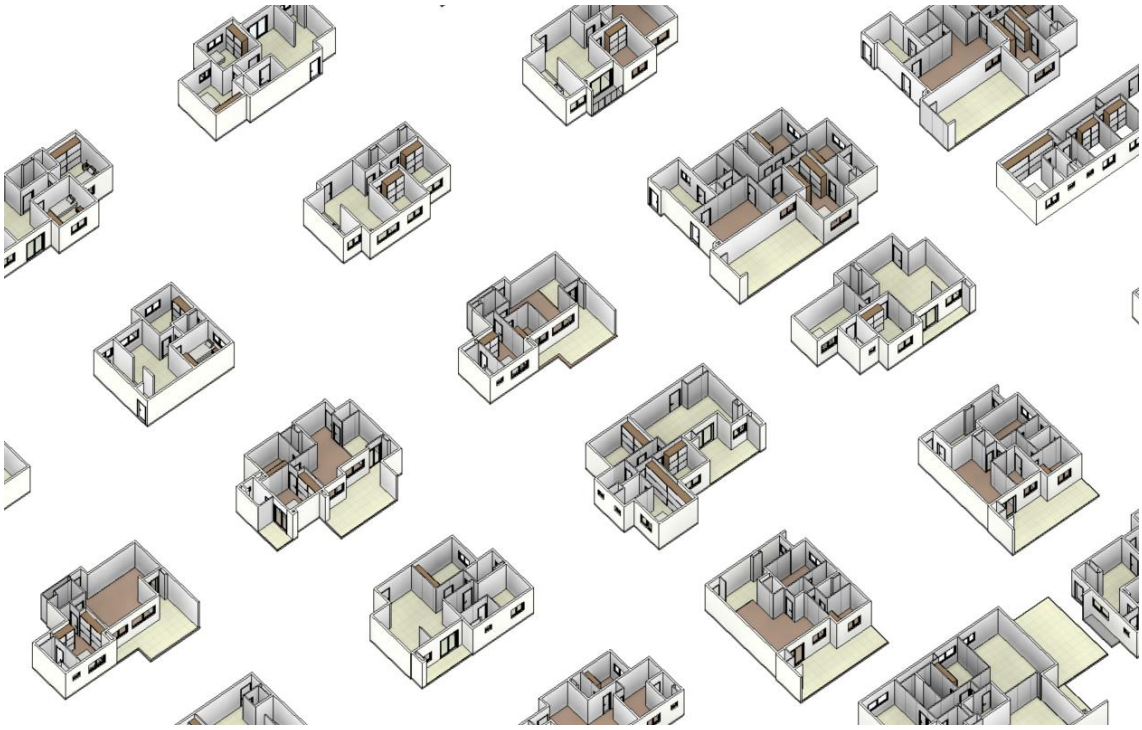
Figura 65 – As 151 plantas modeladas em Revit

Fonte: elaborado pelo autor

Já nesta etapa foi possível chegar a algumas conclusões:

- Apartamentos de uma mesma construtora possuem uma linguagem bem definida e é comum sua repetição em outros empreendimentos, mesmo em outras cidades;
- Quando se trata de dormitórios, há poucas variações de layout, seja para apartamentos mais econômicos seja para aqueles de luxo;
- As áreas dos dormitórios possuem pouca variação;
- Os dormitórios puderam ser agrupados em tipologias retangulares, em L e complexos, separados em dormitório de solteiro e de casal.

Figura 66 - Perspectiva de alguns dos apartamentos modelados em Revit



Fonte: elaborado pelo autor

3.2. Aplicação do algoritmo

O algoritmo denominado artefato 4 foi aplicado em dormitórios de solteiro e de casal de 31 apartamentos escolhidos aleatoriamente, totalizando 84 dormitórios, sendo 31 de casal e 53 de solteiro. Apesar do número de apartamentos modelados ter sido bem maior que os utilizados, no decorrer da experimentação se constatou que essa era uma quantidade suficiente para a amostra, levando em consideração a variabilidade limitada entre tipologias de dormitórios. Nos dormitórios de casal, o algoritmo inseria uma cama com mesas de cabeceira, um guarda-roupa e, havendo espaço, uma escrivaninha. Já no dormitório de solteiro, uma cama sem mesa de cabeceira, um guarda-roupa e, havendo espaço, uma escrivaninha.

5.4. Resultados e discussão

As soluções alcançadas foram classificadas em ruins, regulares ou boas. Por ruim, convencionou-se classificar os resultados absurdos, como mobiliários fora do ambiente ou duplicados. Aqueles regulares dizem respeito a uma disposição plausível, mas sendo necessários pequenos ajustes. Já os resultados considerados bons foram aqueles em que nenhum ajuste ou apenas ajustes manuais mínimos foram requeridos. No caso da função inicial, de identificação do tipo de dormitório (dormitório de casal ou de solteiro), houve acerto em 100% dos casos. Primeiramente, esse sistema de classificação foi realizado por meio da atribuição heuristicamente de uma chave que identificava a função do cômodo. Em seguida, o algoritmo interpreta esse código atribuído e aplica o layout relativo à cada espaço. Os resultados podem ser vistos na Tabela 7.

Tabela 7 - Resultados da aplicação do artefato 4, sem exclusão de erros de lógica

Análise geral dos 84 dormitórios						
	Ruim		Regular		Bom	
	Total	Percentual	Total	Percentual	Total	Percentual
Geral	**26**	**30,59%**	**25**	**29,41%**	**33**	**38,82%**

Análise dos 32 dormitórios de casal						
	Ruim		Regular		Bom	
	Total	Percentual	Total	Percentual	Total	Percentual
Retangulares	2	6,45%	0	0,00%	6	19,35%
Em L	5	16,13%	3	9,68%	4	12,90%
Complexos	4	12,90%	7	22,58%	0	0,00%
Total parcial	**11**	**35,48%**	**10**	**32,26%**	**10**	**32,26%**

Análise dos 53 dormitórios de solteiro (layout para 1 pessoa)						
	Ruim		Regular		Bom	
	Total	Percentual	Total	Percentual	Total	Percentual
Retangulares	14	26,42%	13	24,53%	22	41,51%
Em L	0	0,00%		0,00%	1	1,89%
Complexos	1	1,89%	2	3,77%		0,00%
Total parcial	**15**	**28,30%**	**15**	**28,30%**	**23**	**43,40%**

Fonte: elaborado pelo autor

Os resultados gerais mostram que o algoritmo teve um desempenho aceitável em 68,23% dos casos, sendo que em 38,82% não foi necessário nenhum trabalho adicional ou foi necessário apenas algum ajuste mínimo. Quando os ambientes são divididos pelo uso, os

dormitórios de solteiro apresentam um melhor resultado que os de casal, 71,7% contra 64,52% (entre bons e regulares). Cabe observar aqui que dentro dos resultados ruins é possível fazer uma distinção entre erros de lógica e erros de adequação. O primeiro caso ocorre pelo algoritmo criado não ter coberto todas as possibilidades do mundo real, assim acaba gerando soluções absurdas que claramente nenhum projetista faria, por exemplo: colocação de duas camas de casal em um ambiente ou um armário parte dentro, parte fora, do ambiente. No segundo caso, por sua vez, o algoritmo acerta a colocação da mobília, mas isso não reflete as melhores práticas de projeto. Um caso observado, por exemplo, foi concentrar o mobiliário em uma região do cômodo e deixar outra vazia. Dentro deste contexto, ao se rever os resultados, seriam excluídos 19 casos de erro de lógica. Assim, as novas porcentagens para dormitórios de solteiro seriam 54% de bons resultados, 35% de regulares e 11% de ruins e para dormitórios de casal 45% de bons, 46% de regulares e 9% de ruins.

Quanto às hipóteses iniciais, a geração automatizada atendeu aos critérios de funcionalidade, em 38,82% dos casos. Ela conseguiu, também, mais resultados com menos esforço, considerando que muitos ambientes foram analisados e tiveram o layout colocado de uma vez, porém menos acertados que um procedimento tradicional diante de pequenos ajustes serem necessários e determinados casos não terem conseguido resultados plausíveis. Experimentos futuros podem medir a eficiência do sistema medindo o tempo que os projetistas levam para inserir o mobiliário manualmente e em relação a outros que fazem o mesmo procedimento utilizando o algoritmo criado. Por fim, a geração automatizada resultou em algumas soluções inesperadas, parte delas pouco precisas.

Alguns exemplos podem ser vistos da Figura 67, Figura 69, Figura 71 e Figura 73. A Figura 67 mostra boas soluções geradas tanto para dormitórios de casal quanto de solteiro. Já a Figura 69 demonstra resultados inesperados, como a divisão do ambiente por meio de um guarda-roupa no dormitório de solteiro, e no de casal a escrivaninha ao lado da cama, sendo que ele foi perfeitamente encaixado onde era previsto na planta. A Figura 71 demonstra um erro de lógica da duplicidade da cama de casal bem como a Figura 73 com a cama saindo para fora do ambiente.

A Figura 68, Figura 70, Figura 72 e Figura 74 mostram as plantas dos exemplos anteriores com o layout original. Quando é feita uma análise qualitativa entre as plantas geradas automaticamente, que obtiveram sucesso, e a original, os resultados demonstram tanto a capacidade de reproduzir o que já havia sido feito quanto de apresentar novas

soluções plausíveis. Na Figura 70, por exemplo, é possível ver um dormitório de casal no qual a cama foi colocada em uma outra orientação, mesmo que pequenos ajustes possam ser feitos manualmente.

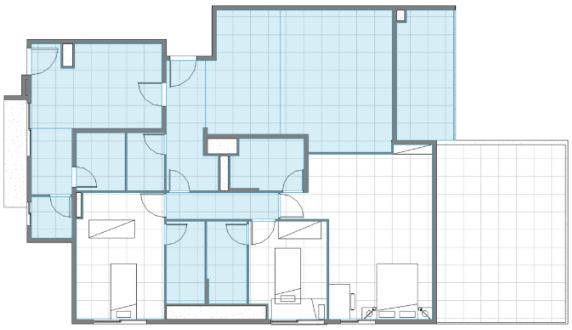
Figura 67 - Exemplo 1 de resultado da geração automatizada. Dormitório de casal e de solteiro classificados como bons resultados

Fonte: elaborado pelo autor

Figura 68 - Planta original do exemplo 1

Fonte: banco de dados de plantas de apartamentos elaborado pelo autor

Figura 69 - Exemplo 2 de resultado da geração automatizada. Dormitório de casal e um de solteiro classificados como regulares e um dormitório de solteiro como ruim

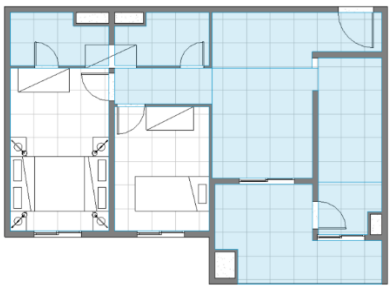


Fonte: elaborado pelo autor

Figura 70 - Planta original do exemplo 2

Fonte: banco de dados de plantas de apartamentos elaborado pelo autor

Figura 71 - Exemplo 3 de resultado da geração automatizada. Dormitório de casal classificado como ruim e o de solteiro como bom resultado

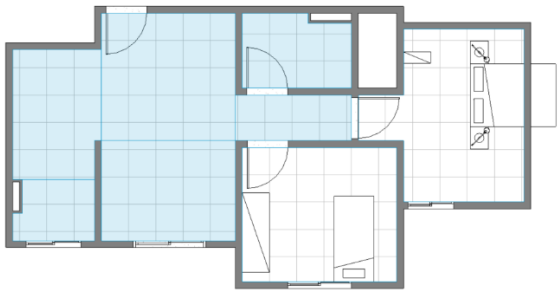


Fonte: elaborado pelo autor

Figura 72 - Planta original do Exemplo 3

Fonte: banco de dados de plantas de apartamentos elaborado pelo autor

Figura 73 - Exemplo 4 de resultado da geração automatizada. Dormitório de casal classificado como ruim e o de solteiro como bom resultado



Fonte: elaborado pelo autor

Figura 74 - Planta original do Exemplo 4

Fonte: banco de dados de plantas de apartamentos elaborado pelo autor

Em linhas gerais, os resultados demonstram o desafio de automatizar processos intuitivos do projeto arquitetônico quando os problemas vão ganhando maior indefinição, como foi problematizado na revisão da literatura a partir dos trabalhos de Mitchell (1975), Eastman (1969) e Reitman (1965 *apud* MITCHELL, 1975). Isso é demonstrado pela diferença nos resultados obtidos nas diferentes tipologias. Dormitórios de solteiro, por exemplo, tendem a apresentar uma forma mais simples, retangular e sem um banheiro anexo; já os dormitórios maiores podem apresentar uma variedade maior de peças de mobiliário e disposição dos elementos. Assim, o algoritmo apresentou melhores resultados nos cômodos com maior previsibilidade das condicionantes. Segundo nossa análise qualitativa, ambientes menores e com menos elementos tendem a representar problemas mais definidos que ambientes maiores e com mais elementos, permitindo uma maior eficiência de aplicação do algoritmo, sem muitos ciclos de geração e análise, o que está de acordo com o modelo descrito por Eastman (1969).

Uma propriedade importante a ser destacada é como a utilização do BIM trouxe benefícios para a disposição automática de mobiliários nos ambientes. Em primeiro lugar, seus benefícios estão presentes na facilidade de filtrar elementos por categorias ou qualquer outro parâmetro que seja relevante para o sistema. Essa estruturação da informação aliada com a semântica embutida no BIM permitiu, por exemplo, uma rápida identificação do "volume" entre paredes, que denominamos "dormitórios", como recorte para o trabalho. Em seguida, por meio de um procedimento heurístico, essa informação de nomenclatura determinou se o uso deveria ser de casal ou de solteiro, quais elementos deveriam ser inseridos ali e de que maneira deveria ser feita essa parametrização. Cada elemento inserido, por sua vez, poderia conter informações como vida útil, fabricante, custo etc. Deste modo, quaisquer dados poderiam ser tabelados e, no caso de custo, compor uma planilha de

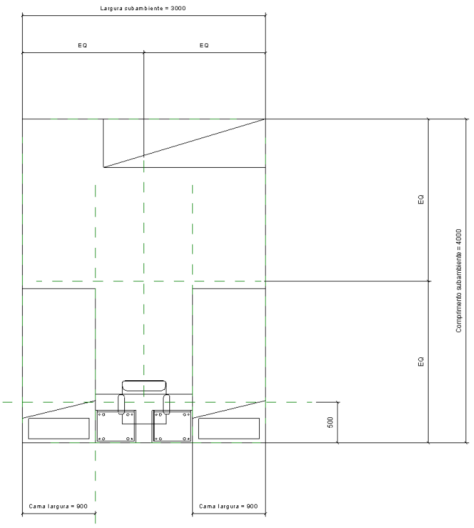
orçamento. Esse rápido fluxo de geração beneficiaria os arquitetos por permitir uma rápida análise de cenários, cruzando informações. Há aqui, portanto, mais do que uma disposição física de objetos no espaço, mas uma algoritmização das informações do projeto arquitetônico em BIM.

Em segundo lugar, o "aninhamento" de elementos trouxe um grande benefício para o desenvolvimento de automatizações no projeto. Dentro do BIM e, especificamente dentro do Revit, é possível inserir famílias (elementos) dentro de outras famílias. Por exemplo, no geral, uma cama de casal e uma mesa de cabeceira sempre são colocadas uma ao lado da outra. Assim, uma família pode ser feita em BIM de modo que contenha estas duas peças do mobiliário com uma restrição de adjacência. Essa propriedade tem um impacto significativo na codificação do algoritmo, visto que ao invés de aplicar um conjunto de regras para cada família, o procedimento é feito apenas uma vez. Isto nos conduz a uma constatação qualitativa de que quanto mais famílias estiverem aninhadas (isto é, agrupadas umas dentro das outras), maior será a facilidade de codificação. A consequência direta desta afirmação é que seria mais eficiente criar o mínimo possível de famílias por ambiente a partir de uma leitura sobre as tipologias possíveis de layouts. Uma sistemática de aninhamento preservaria a quantidade de peças do mobiliário, ou seja, para o usuário ele estará inserindo uma única família, como um dormitório de solteiro que contém, uma cama, uma mesa de cabeceira, um tapete, um abajur, por exemplo, enquanto para o BIM seriam diversas famílias agrupadas, sendo facilmente separadas por categorias e consequentemente uma estruturação automática das informações.

Um possível desdobramento desta metodologia, é uma nova abordagem na relação entre projetos de interiores e de Arquitetura ou ainda no equilíbrio entre forma e função. No experimento realizado nesta dissertação, plantas de apartamentos foram construídas de início e depois os dormitórios foram populados com famílias. O mesmo procedimento poderia ser replicado para outros ambientes seguindo a mesma metodologia. Entretanto, os resultados alcançados nos permitem inferir que seria possível elaborar uma planta de apartamento (ou até de outro uso) automaticamente de dentro para fora. Isto seria semelhante ao artefato 2, mas neste caso, haveria ali um número reduzido de famílias, porém com inúmeros elementos aninhados com relações paramétricas previamente construídas (Figura 75, Figura 76 e Figura 77). A partir da mudança de parâmetros gerais como largura e comprimento destes elementos que compõem o ambiente (Figura 78), paredes poderiam ser erguidas e janelas e portas poderiam ser inseridas, tudo de maneira automática, segundo regras estabelecidas previamente. Portanto, a atual categorização de elementos em BIM já estaria ultrapassada,

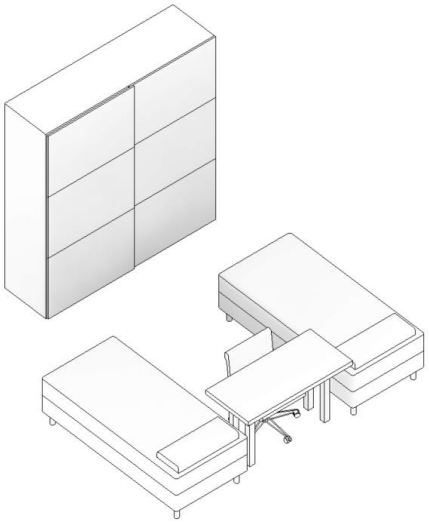
visto que não se estaria trabalhando simplesmente com a categoria “mobiliário” ou “elementos de iluminação”, “janelas”, por exemplo, mas com categorias de “ambientes”. Poderiam ser criadas categorias como “dormitório”, “cozinha”, “sala de estar”, “banheiro” e assim por diante.

Figura 75 – Relações paramétricas construídas dentro de uma família com famílias aninhadas



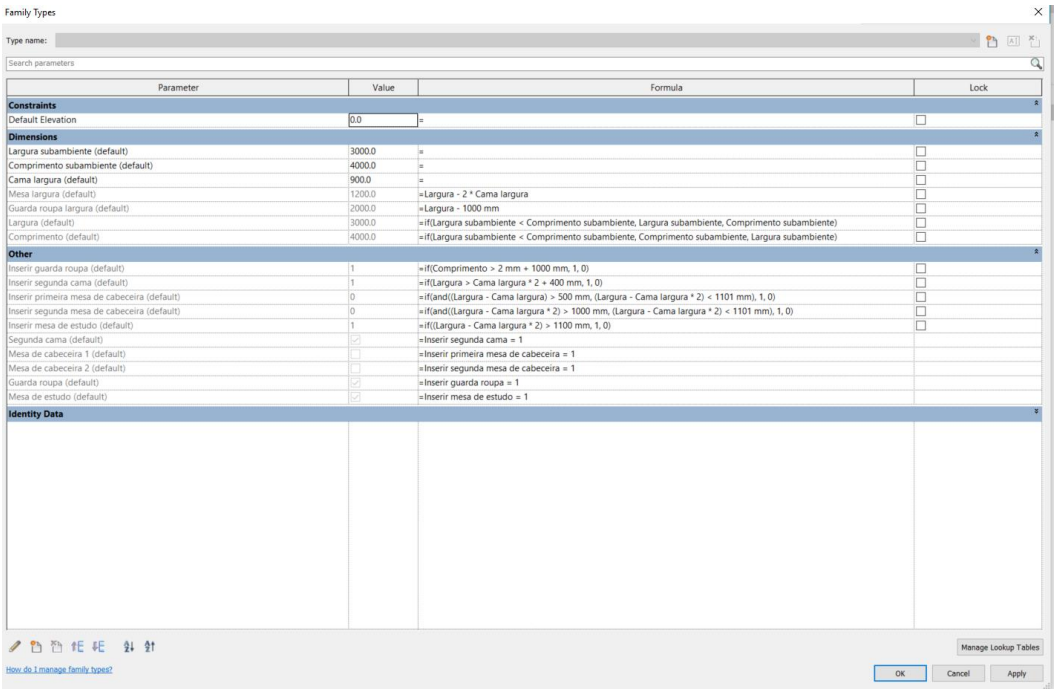
Fonte: elaborado pelo autor

Figura 76 – Uma única família contendo diversas famílias



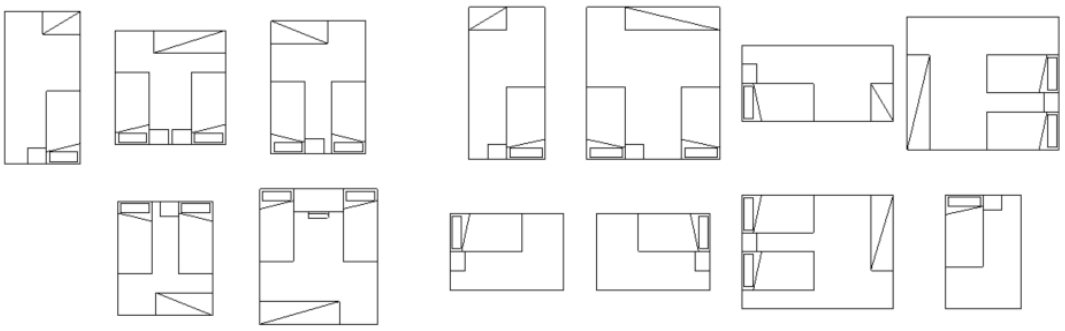
Fonte: elaborado pelo autor

Figura 77 – Relações matemáticas para boa distribuição do mobiliário dentro do cômodo



Fonte: elaborado pelo autor

Figura 78 – Uma única família com famílias aninhadas e com relações paramétricas possibilita diversas variações



Fonte: elaborado pelo autor

Esta é uma propriedade inexplorada pelos autores trabalhados na revisão da literatura sobre *Space Planning*. Entretanto, seu conceito é claramente discutido por Kieran e Timberlake (2003). Os autores defendem que devemos olhar para a Arquitetura como conjuntos de módulos, partes e montagens, mais do que enxergar elementos isoladamente, à semelhança da fabricação dos navios, automóveis e aeronaves. Os benefícios disso são o maior potencial de melhorias em cada conjunto separadamente, a maior adequabilidade de substituição de peças e uma grande abertura para a industrialização da construção. Assim, essa "modularização" aliada à sistemática de aninhamento presente no BIM parece ter um grande

potencial para a industrialização de edifícios inteiros e de seus interiores, proporcionando, inclusive, maior intersecção com a indústria 4.0.

Deste modo, constata-se que é plausível a aplicação de algoritmos para a disposição de mobiliário com o objetivo de reduzir o esforço entre concepção e projeto executivo. O algoritmo desenvolvido pode ser aprimorado para aumentar a sua eficiência e expandido para atender a outros ambientes residenciais, comerciais ou institucionais. Além disso, esses algoritmos podem ser uma ponte para a industrialização da Arquitetura.

6 CONCLUSÃO

O aprimoramento das técnicas de Inteligência Artificial tem desafiado as tradicionais estruturas socioeconômicas a romper paradigmas e a se adaptar a um mundo em rápida transformação (ALPAYDIN, 2016). A Arquitetura não é uma exceção nessa revolução tecnológica. Isso é demonstrado pelo crescimento exponencial de artigos que relacionam AEC com IA (DARKO *et. al*, 2020) e pelos governos ao redor do mundo, que têm incluído em seu plano estratégico e legislado sobre tecnologias centrais para essa transformação na Arquitetura, como tem ocorrido com o BIM e a pré-fabricação (ABANDA; TAH; CHEUNG, 2017; ESTRATÉGIA BIM BR, 2018) .

Neste contexto, é importante destacar que, mesmo atualmente, grande parte dos computadores são subutilizados pelos arquitetos. Isto é, eles são utilizados apenas como uma ferramenta para a criação de planilhas, imagens, desenhos e modelos tridimensionais, mas é ignorado o potencial de automatização de processos de geração e de análise desde o começo do desenvolvimento do projeto. Assim como Mitchell (1975) e Negroponte (1970) já propunham desde o início da utilização dos computadores na Arquitetura, devemos visar uma simbiose maior entre máquinas e projetistas. Negroponte (1970) ilustra essa transformação pelo abandono de uma relação entre escravo e senhor, e na construção de uma relação de duas espécies colaborando para a evolução de ambas. Deste modo, se extrai o melhor de cada um, permitindo ampliar o conjunto de soluções perceptíveis e viáveis para determinado problema.

Esta dissertação explorou essas mudanças estruturais nas fases iniciais do projeto arquitetônico. Isso foi feito por meio de desenvolvimento e aplicação de algoritmos na geração de layouts visando diminuir trabalhos repetitivos e o esforço de transição entre a concepção e o desenho executivo. Visto que a maioria dos atuais trabalhos levantados na revisão da literatura focam mais em como desenvolver um sistema que simule os resultados humanos, ao invés de ser uma ferramenta de automatização de etapas específicas do projeto arquitetônico.

Quatro artefatos foram desenvolvidos de modo incremental a fim de explorar as aplicações de algoritmos de *Space Planning* no projeto de interiores, sendo que o último deles foi submetido a um experimento para medir sua eficiência segundo os critérios de funcionalidade. Os 68,23% de resultados regulares e bons indicam a plausibilidade do

método, tendo em vista que trabalhos futuros podem ser feitos para aumentar a eficiência do algoritmo e que a mesma metodologia pode ser aplicada para outros tipos de ambientes.

Plantas de apartamentos de diferentes partes do Brasil foram analisadas com o objetivo de compor um amplo e diverso espaço amostral, levando em consideração possíveis especificidades regionais, como mobília, dimensões e distribuição de cômodos. Assim, o layout de mobiliário dos dormitórios dos apartamentos estudados são uma evidência do que se está construindo no país. Consequentemente, o algoritmo criado reproduz heuristicamente essas características.

Diante do método desenvolvido, três desdobramentos são possíveis para trabalhos futuros. Primeiramente, a mesma lógica procedural pode compreender outros cômodos do apartamento. Na sequência, poderão ser realizados estudos levando em consideração a interface entre eles e as suas relações de área e complexidade. Por fim, novas pesquisas podem ser realizadas, explorando a geração de ambientes arquitetônicos a partir de sistemas com um alto nível de aninhamento de peças de mobiliário, isto é, das relações paramétricas entre os elementos. Espera-se que, deste modo, novas plantas de apartamentos possam ser criadas a partir do layout da mobília, contribuindo para a otimização dos espaços.

Outra linha possível a ser seguida em trabalhos futuros é implementar Inteligência Artificial desde o levantamento de dados até o desenho. Em uma fase inicial, por exemplo, o sistema pode fazer uma busca sistemática por plantas disponíveis na internet, substituindo a procura manual. Em continuidade, técnicas de *machine learning* poderiam ser utilizadas para analisar as informações obtidas, definir padrões e gerar soluções.

Neste contexto, entende-se que a associação de algoritmos em BIM integrada ao projeto arquitetônico pode contribuir mais do que para a produtividade da Arquitetura, mas para a análise da lógica intuitiva dos processos associada à semântica construtiva, isto é, uma autocrítica de como fazemos e como pensamos Arquitetura. Assim, ainda que aplicações didáticas não tenham feito parte do escopo desta dissertação, o algoritmo desenvolvido poderia ser aplicado ao ensino de Arquitetura, permitindo visualizar rapidamente as possíveis soluções de layout para diferentes dimensões de dormitórios. O ensino de Arquitetura por meio de algoritmos procedurais e de Inteligência Artificial pode ser uma ferramenta para contribuir para o entendimento da lógica por trás de procedimentos intuitivos na prática projetual, bem como de instigação para a construção de processos mais eficientes. Nesta abordagem, há uma problematização sobre os processos, antes de se obter os resultados. Quando essa lógica é associada ao BIM, por sua vez, há a construção de códigos sobre o

modelo digital da futura construção, uma simulação do real que permite o fomento da colaboração entre profissionais e integração com novas tecnologias.

Deste modo, as máquinas podem evoluir de simples ferramentas que recebem ordens de projetistas para um sistema de Inteligência Artificial essencial no processo de projeto. Embora, não se tenha aplicado tal procedimento nesta dissertação, uma etapa futura seria implementar no algoritmo desenvolvido um sistema de *machine learning* que permitisse que o código se aperfeiçoasse à medida em que os usuários lhe dissessem que ele havia acertado ou errado o layout. Haveria, assim, uma colaboração contínua entre projetistas e máquinas.

7 REFERÊNCIAS

ABANDA, F. H.; TAH, J. H. M.; CHEUNG, F. K. T. BIM in off-site manufacturing for buildings. **Journal of Building Engineering**, v. 14, p. 89–102, 2017. Disponível em: https://www.sciencedirect.com/science/article/pii/S2352710217301237. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1016/j.jobe.2017.10.002

ABDELMOHSEN, S. *et al.* A Heuristic Approach for the Automated Generation of Furniture Layout Schemes in Residential Spaces. In: DESIGN COMPUTING AND COGNITION DCC'16, 2016, Chicago. **Anais**. Charlotte: Springer, 2016, pp. 483-502. Disponível em: https://www.researchgate.net/publication/297032365_A_Heuristic_Approach_for_the_Automated_Generation_of_Furniture_Layout_Schemes_in_Residential_Spaces. Acesso em: 08 ago. 2020.

AKASE, R.; OKADA, Y. Automatic 3D Furniture Layout Based on Interactive Evolutionary Computation. In: 2013 SEVENTH INTERNATIONAL CONFERENCE ON COMPLEX, INTELLIGENT, AND SOFTWARE INTENSIVE SYSTEMS, 2013, Taichung. **Anais eletrônicos**. Taichung: IEEE, 2013, pp. 726-731. Disponível em: https://ieeexplore.ieee.org/document/6603980. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1109/CISIS.2013.130

ALPAYDIN, E. **Machine learning** – The New AI. Cambridge: MIT Press, 2016. 172 p.

ANDERSON, C. *et al.* Augmented space planning: Using procedural generation to automate desk layouts. **International Journal of Architectural Computing**, v. 16, p. 164–177, 2018. Disponível em: https://journals.sagepub.com/doi/full/10.1177/1478077118778586. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1177/1478077118778586

ARCHITECTURE 2030. **The 2030 Challenge**, 2018. Disponível em: https://architecture2030.org/2030_challenges/2030-challenge/. Acesso em: 27 jan. 2020.

ARVIN, S. A.; HOUSE, H. Modeling architectural design objectives in physically based space planning. **Automation in Construction**, v. 11, n. 2, p. 213–225, 2002. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0926580500000996. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1016/S0926-5805(00)00099-6

AS, I.; PAL, S.; BASU, P. Artificial intelligence in architecture: Generating conceptual design via deep learning. **International Journal of Architectural Computing**, v. 16, n. 4, p. 306–327, 2018. Disponível em: http://journals.sagepub.com/doi/10.1177/1478077118800982. Acesso em: 08 ago. 2020.

BHARGAVA, A. Y. **Entendendo Algoritmos**: Um Guia Ilustrado Para Programadores e Outros Curiosos. São Paulo: Novatec Editora, 2017. 264 p. Versão eletrônica.

BAZALO, F.; MOLETA, T. RESPONSIVE ALGORITHMS An investigation of computational processes in early stage design. In: INTERNATIONAL CONFERENCE OF THE ASSOCIATION FOR COMPUTER-AIDED ARCHITECTURAL DESIGN RESEARCH IN ASIA (CAADRIA), 2015, Daegu. **Anais eletrônicos**. Hong Kong: CAADRIA, 2015. Disponível em: http://papers.cumincad.org/cgi-

bin/works/paper/caadria2015_237. Acesso em: 08 ago. 2020. DOI: 10.13140/RG.2.2.23500.41605

BEGON, M.; TOWNSEND, C; HARPER, J. **Ecology**: From Individuals to Ecosystems [4ed.]. Malden: Blackwell Publishing, 2006. 759 p.

BELÉM, C.; SANTOS, L.; LEITÃO, A. On the Impact of Machine learning. Architecture without Architects?". In: CAAD FUTURES 2019, 2019, Seul. **Anais eletrônicos**. Daejeon: CAAD Futures, 2019, pp. 148-167. Disponível em: http://papers.cumincad.org/cgi-bin/works/paper/cf2019_020. Acesso em: 08 ago. 2020. ISBN 978-89-89453-05-5]

BIAGINI, C.; DONATO, V.; PELLIS, D. Preliminary Design Through Graphs: A Tool for Automatic Layout Distribution. In: **ICONARP International Journal of Architecture and Planning**, [S.l.], v. 2, n. 2, p. 1-13, feb. 2015. ISSN 2147-9380. Disponível em: http://iconarp.selcuk.edu.tr/iconarp/article/view/60. Acesso em: 08 ago. 2020.

BIANCONI, F.; FILIPPUCCI, M.; BUFFI, A. Automated design and modeling for mass-customized housing. A web-based design space catalog for timber structures. **Automation in Construction**, v. 103, p. 13–25, 2019. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0926580518305302. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1016/j.autcon.2019.03.002

BUCHANAN, R. Wicked Problems in Design Thinking. **Design Issues**, v. 8, n. 2. Cambridge: MIT Press, 1992. Pp.5-21. Disponível em: https://web.mit.edu/jrankin/www/engin_as_lib_art/Design_thinking.pdf. Acesso em: 08 ago. 2020.

BUTTERFIELD, A.; NGONDI, Gerard E.; KERR, A. (Orgs.). **A Dictionary of Computer Science**. Edição: 7. Oxônia: OUP Oxford, 2016. 641 p.

CALIXTO, V.; CELANI, G. A literature review for space planning optimization using an evolutionary algorithm approach: 1992-2014. In: XIX CONGRESSO DA SOCIEDADE IBERO-AMERICANA DE GRÁFICA DIGITAL, 2015, Florianópolis **Anais eletrônicos**. Florianópolis: Editora Edgard Blücher, 2015, p. 662–671. Disponível em: http://www.proceedings.blucher.com.br/article-details/22382. Acesso em: 26 jul. 2020. DOI: 10.5151/despro-sigradi2015-110166

CALIXTO, V. **Geração automatizada de layouts com o uso de algoritmos evolutivos: aplicações em arquitetura e urbanismo**. Dissertação (Mestrado em Arquitetura e Urbanismo) – Universidade Estadual de Campinas, Campinas, 2016.

CAMBRIDGE DICTIONARY. Cambridge: Cambridge University Press, 2020. Disponível em: https://dictionary.cambridge.org/pt/. Acesso em: 08 ago. 2020.

CASTAGNINO, S *et al.* **The Transformative Power of Building Information Modeling**. The Boston Consulting Group**,** 2016. Disponível em: https://www.bcg.com/en-ch/publications/2016/engineered-products-infrastructure-digital-transformative-power-building-information-modeling.aspx. Acesso em: 28 jan. 2020.

_____. **6 ways the construction industry can build for the future**. World Economic Forum, 2018. Disponível em: https://www.weforum.org/agenda/2018/03/how-construction-industry-can-build-its-future/. Acesso em: 28 jan. 2020.

CELANI, G. **Beyond analysis and representation in CAD: a** new computational approach to design education. Thesis, Massachusetts Institute of Technology, 2002. Disponível em: https://dspace.mit.edu/handle/1721.1/8016. Acesso em: 23 jan. 2020.

CELANI, G *et al.* The Future of the Architect's Employment: To Which Extent Can Architectural Design Be Computerised? In: CAAD FUTURES 2015, 2015, São Paulo. **Anais**. Heidelberg: Springer Berlin Heidelberg, 2015. [s.n.]

CHAILLOU, S. **AI + Architecture** | Towards a New Approach. Dissertação (Mestrado em Arquitetura). Harvard University, 2019. Disponível em: https://www.academia.edu/39599650/AI_Architecture_Towards_a_New_Approach. Acesso em: 13 abr. 2020.

CHANG, J. HyperCell: A Bio-inspired Design Framework for Real-time Interactive Architectures. **A+BE | Architecture and the Built Environment**, [S.l.], n. 1, p. 1-250, jan. 2018. ISSN 2214-7233. Disponível em: https://journals.open.tudelft.nl/abe/article/view/1947. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.7480/abe.2018.1.1947

CÔCO JÚNIOR, V. H.; CELANI, G. From the automated generation of layouts to fabrication with the use of BIM: a new agenda for Architecture in the 21st century. In: XXII CONGRESSO INTERNACIONAL DA SOCIEDADE IBEROAMERICANA DE GRÁFICA DIGITAL (SIGRADI), 7 nov. 2018, São Carlos. **Anais eletrônicos.** São Carlos: Blucher Design Proceedings, 7 nov. 2018. p. 23–30. Disponível em: https://www.proceedings.blucher.com.br/article-details/-29680. Acesso em: 11 nov. 2018. DOI 10.5151/sigradi2018-1302

COLMAN, A. M**. A Dictionary of Psychology.** Oxônia: OUP Oxford, 2015. 896 p.

CORMEN, T.; LEISERSON, C. E.; RIVERST, R. L.; STEIN, C. **Algoritmos**: Teoria e Prática. [s.l.]: Gen LTC, 2012

CORREIA, R.; DUARTE, J.; LEITÃO, A. GRAMATICA: A general 3D shape grammar interpreter targeting the mass customization of housing. In: 30th ECAADE CONFERENCE, 2012, Praga. **Anais da 30th eCAADe Conference** - Volume 1 / ISBN 978-9-4912070-2-0. Praga, 2012, pp. 489–496. Disponível em: http://papers.cumincad.org/cgi-bin/works/Show?ecaade2012_273. Acesso em: 08 ago. 2020.

DAMÁSIO, António. **Em busca de Espinosa**. Ed. 1. São Paulo: Companhia das Letras, 2004. 358 p.

DARKO *et al.* Artificial intelligence in the AEC industry: Scientometric analysis and visualization of research activities. **Automation in Construction**, v. 112, p. 103081, 2020. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S092658051930651X. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1016/j.autcon.2020.103081

DAS *et al.* Space Plan Generator: Rapid Generation & Evaluation of Floor Plan Design Options to Inform Decision Making. In: ACADIA // 2016: POSTHUMAN FRONTIERS: DATA, DESIGNERS, AND COGNITIVE MACHINES. **Anais da 36th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA) ISBN 978-0-692-77095-5] Ann Arbor 27-29 October, 2016, pp. 106-115**.

[s.l.]: CUMINCAD, 2016. Disponível em: http://papers.cumincad.org/cgi-bin/works/Show?acadia16_106. Acesso em: 25 abr. 2020.

DINO, I. An evolutionary approach for 3D architectural space layout design exploration. **Automation in Construction**, v. 69, p. 131–150, 2016. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0926580516301005. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1016/j.autcon.2016.05.020

DOWNEY, A. B. **Pense em Python**: Pense Como um Cientista da Computação. São Paulo: Novatec, 2016. 312 p.

DRESCH, A.; LACERDA, D. P.; ANTUNES JR, J. A. V. **Design Science Research: A Method for Science and Technology Advancement**. Nova York: Springer, 2014.

DU, T. *et al.* Gaps and requirements for automatic generation of space layouts with optimised energy performance. **Automation in Construction**, v. 116, p. 103132, 2020. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0926580519307496. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1016/j.autcon.2020.103132

EASTMAN, C. M. **Explorations of the cognitive processes in design**. Pittsburgh: Carnegie Mellon University, 1968. Disponível em: https://www.researchgate.net/publication/235072413_EXPLORATIONS_OF_THE_COGNITIVE_PROCESSES_IN_DESIGN. Acesso em: 20 ago. 2019.

_____. **On the Analysis of Intuitive Design Processes**. Pittsburgh: Carnegie-Mellon University, 1968b. Pp 21-35. Disponível em: http://users.metu.edu.tr/baykan/arch586/Readings/Cognition/Eastman.pdf. Acesso em: 08 ago. 2020.

_____. Cognitive processes and ill-defined problems: A case study from design. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1969, Washington. **Anais da International Joint Conference on Artificial Intelligence**. Washington: Eds Walker, 1969, pp. 669–690. Disponível em: https://www.semanticscholar.org/paper/Cognitive-Processes-and-Ill-Defined-Problems%3A-A-Eastman/d1c8dd5bfcd1270af88f9fb151a377a6051ad2a3. Acesso em: 08 ago. 2020.

_____. Automated space planning. **Artificial intelligence**, Elsevier, v. 4, n. 1, p. 41–64, 1973. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/0004370273900088. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1016/0004-3702(73)90008-8

EASTMAN, C *et al.* **Manual de BIM**: um guia de modelagem da informação da construção para arquitetos, engenheiros, gerentes, construtores e incorporadores. Porto Alegre: Bookman, 2014. 481 p.

ESTRATÉGIA BIM BR. **Estratégia Nacional de Disseminação do Building Information Modeling - BIM**. Brasília: Ministério da Indústria, Comércio Exterior e Serviços, 2018. Disponível em: http://www.mdic.gov.br/images/REPOSITORIO/sdci/CGMO/26-11-2018-estrategia-BIM-BR-2.pdf. Acesso em: 20 ago. 2019.

FU, Q. *et al.* Adaptive synthesis of indoor scenes via activity-associated object relation graphs. **ACM Transactions on Graphics**, v. 36, n. 6, p. 201:1–201:13, 2017. Disponível em: https://dl.acm.org/doi/10.1145/3130800.3130805. Acesso em: 08 ago. 2020.

GERMER, T.; SCHWARZ, M. Procedural Arrangement of Furniture for Real-Time Walkthroughs. **Computer Graphics Forum**, v. 28, n. 8, p. 2068–2078, 2009. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2009.01351.x. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1111/j.1467-8659.2009.01351.x

GHAFFARIAN, M.; FALLAH, R.; JACOB, C. Organic architectural spatial design driven by agent-based crowd simulation. In: SIMAUD '18. **Anais do Symposium on Simulation for Architecture and Urban Design (SIMAUD '18)**. Delft: Society for Computer Simulation International, 2018, p. 1–8. Disponível em: https://dl.acm.org/doi/10.5555/3289750.3289767. Acesso em: 08 ago. 2020.

GILL, Karamjit S. Human Machine Symbiotics: On Control and Automation in Human Contexts. In: 14TH IFAC WORKSHOP ON INTERNATIONAL STABILITY AND SYSTEMS ENGINEERING. **IFAC Proceedings Volumes**, v. 45, n. 10, p. 91–96, 2012. Disponível em: https://www.sciencedirect.com/science/article/pii/S1474667015337174. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.3182/20120611-3-IE-4029.00019

GROPIUS, W. Computers for Architectural Design. **Architecture and the Computer Conference**, Boston Architectural Center. Serge Chermayeff Archive, Avery Archive, Columbia University. NY, NY. 4 de dezembro de 1964. Boston. Pp. 91-92 *apud* ROCHA, A. J. M. *Architecture theory, 1960-1980: emergence of a computational perspective*. Tese (Doutorado) — Massachusetts Institute of Technology, 2004.

HERRE, E. A.; WEST, S. A. Conflict of interest in a mutualism: documenting the elusive fig wasp-seed trade-off. **Anais do Royal Society B: Biological Sciences**, v. 264, n. 1387, p. 1501–1507, 1997 *apud* BEGON, M.; TOWNSEND, C; HARPER, J. Ecology: From Individuals to Ecosystems [4ed.]. Malden: Blackwell Publishing, 2006. 759 p.

HILS, D. Visual languages and computing survey: Data flow visual programming languages. **Journal of Visual Languages & Computing**, v. 3, n. 1, p. 69–101, 1992. Disponível em: http://fab.cba.mit.edu/classes/S62.12/docs/Hils_visual.pdf. Acesso em: 08 ago. 2020.

HUANG, W.; ZHENG, H. Architectural Drawings Recognition and Generation through Machine learning. In: 38TH ANNUAL CONFERENCE OF THE ASSOCIATION FOR COMPUTER AIDED DESIGN IN ARCHITECTURE (ACADIA), 2018, Cidade do México. **Anais da 38th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)**, 2018.

KÁN, P.; KAUFMANN, H. Automated interior design using a genetic algorithm. In: PROCEEDINGS OF THE 23RD ACM SYMPOSIUM ON VIRTUAL REALITY SOFTWARE AND TECHNOLOGY. **Anais do 23rd ACM Symposium on Virtual Reality Software and Technology**. Gothenburg, Sweden: Association for Computing Machinery, 2017, p. 1–10. (VRST '17). Disponível em: https://doi.org/10.1145/3139131.3139135. Acesso em: 1 ago. 2020.

KÁN, P.; KAUFMANN, H. Automatic Furniture Arrangement Using Greedy Cost Minimization. In: 25TH IEEE CONFERENCE ON VIRTUAL REALITY AND 3D

USER INTERFACES. **Anais eletrônicos 25th Ieee Conference On Virtual Reality And 3d User Interfaces**, 2018, p. 491–498. Disponível em: https://ieeexplore.ieee.org/xpl/conhome/8423729/proceeding. Acesso em: 08 ago. 2020.

KIERAN, S.; TIMBERLAKE, J. **Refabricating Architecture**: How Manufacturing Methodologies are Poised to Transform Building Construction. 1ª Edição. Nova York: McGraw-Hill Education, 2003. 193 p.

KJØLAAS, Kari Anne Høier. **Automatic furniture population of large architectural models**. Thesis, Massachusetts Institute of Technology, 2000. Disponível em: https://dspace.mit.edu/handle/1721.1/86455. Acesso em: 25 abr. 2020.

KNECHT, K; KOENIG, R. Generating floor plan layouts with k-d trees and evolucionary algorithms. In: GA2010 - 13TH GENERATIVE ART CONFERENCE, 2010, Milão. **Anais da GA2010-13TH Generative Art Conference**, 2010. Pp 238-253. Disponível em: https://www.researchgate.net/publication/256471356_Generating_Floor_Plan_Layouts_with_K-d_Trees_and_Evolutionary_Algorithms. Acesso em: 08 ago. 2020.

KNUTH, D. E. Computer Science and Its Relation to Mathematics. **The American Mathematical Monthly**; Vol. 81, No. 4 (Apr., 1974), pp. 323-343.KOENIG, R.; KNECHT, K. Comparing two evolutionary algorithm based methods for layout generation: Dense packing versus subdivision. **AI EDAM**, v. 28, n. 3, p. 285–299, 2014. Disponível em: https://www.cambridge.org/core/journals/ai-edam/article/comparing-two-evolutionary-algorithm-based-methods-for-layout-generation-dense-packing-versus-subdivision/CF1818B1DDA0BB4B15A72EF21F68E6AB. Acesso em: 08 ago. 2020. **DOI: https://doi.org/10.1017/S0890060414000237**

KRONSJÖ, L. **Algorithms**: Their Complexity and Efficiency. Nova Jersey: Wiley, 1987 378 p.

LACERDA, D. *et al.* Design Science Research: método de pesquisa para a engenharia de produção. **Gestão & Produção**, v. 20, n. 4, p. 741–761, 2013. Disponível em: https://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2013000400001&lng=pt&nrm=iso. Acesso em: 09 ago. 2020.

LANGENHAN, C. *et al.* Graph-based retrieval of building information models for supporting the early design stages. **Advanced Engineering Informatics**, v. 27, n. 4, p. 413–426, 2013.Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S1474034613000499. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1016/j.aei.2013.04.005

LANGRISH, J. Z. The Design Methods Movement: From Optimism to Darwinism. In: DRS 2016, DESIGN RESEARCH SOCIETY 50TH ANNIVERSARY CONFERENCE, 2016, Brighton. **Anais eletrônicos Drs 2016, Design Research Society 50th Anniversary Conference, 2016.** Disponível em: https://www.drs2016.org/222. Acesso em: 08 ago. 2020.

LEACH, N. Parametrics Explained. In: LEACH, N; Yuan, P. (Eds.). **Scripting the Future**. Xangai: Tongji University Press, 2012. pp. 9-13

LEITE, R. M. **Personalização em série para o projeto e a produção de espaços flexíveis**. Dissertação (Mestrado em Arquitetura e Urbanismo) - Universidade Estadual de

Campinas, Campinas, 2020. Disponível em: http://repositorio.unicamp.br/handle/REPOSIP/343534. Acesso em: 08 ago. 2020.

LOPES *et al.* A constrained growth method for procedural floor plan generation. In: 11TH INTERNATIONAL CONFERENCE ON INTELLIGENT GAMES AND SIMULATION, 2010, Leicester. **Anais da 11th International Conference on Intelligent Games and Simulation**. Leicester, Reino Unido, 2010. pp. 13-23. Disponível em: https://www.researchgate.net/publication/265988238_A_constrained_growth_method_for_procedural_floor_plan_generation. Acesso em: 08 ago. 2020.

MAI, L.; OWL, M.; KERSTING, M. **The Cambridge Dictionary of Human Biology and Evolution**. Cambridge: Cambridge University Press, 2005. 648 p.

MARTIN, E.; RUSE; M.; HOLMES, E. [Eds]. **A Dictionary of Biology** [3ed]. Nova York: Oxford University Press, 1996. 552 p.

MENGES, Achim. The New Cyber-Physical Making in Architecture: Computational Construction. **Architectural Design**, v. 85, 2015. Disponível em: https://www.researchgate.net/publication/281521762_The_New_Cyber-Physical_Making_in_Architecture_Computational_Construction. Acesso em: 08 ago. 2020.

MERRELL, P. *et al.* Interactive Furniture Layout Using Interior Design Guidelines. In: SIGGRAPH '11, 2011, Vancouver. **Anais SIGGRAPH '11**, v. 30, 2011. Disponível em: https://dl.acm.org/doi/10.1145/1964921.1964982. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1145/1964921.1964982

MICHALEK, J.; CHOUDHARY, R.; PAPALAMBROS, P. Architectural layout design optimization. **Engineering Optimization**, v. 34, n. 5, p. 461–484, 2002. Disponível em: https://www.cmu.edu/me/ddl/publications/2002-Michalek,Choudhary,Papalambros-EO-ArchLayout.pdf. Acesso em: 08 ago. 2020.

MITCHELL, W. J. The Theoretical Foundation of Computer-Aided Architectural Design: **Environment and Planning B: Planning and Design**, 1975. Disponível em: https://journals.sagepub.com/doi/10.1068/b020127. Acesso em: 25 abr. 2020.

MOHAMAD, K.; AMORIM, S. **BIM**: Building Information Modeling no Brasil e na União Europeia. Brasília: Ministério do Desenvolvimento, Indústria e Comércio Exterior e o Ministério do Planejamento, Orçamento e Gestão, 2015. Disponível em: http://sectordialogues.org/sites/default/files/acoes/documentos/bim.pdf. Acesso em: 10 ago. 2019.

MORIN, E. **Introdução ao Pensamento Complexo**. Edição: 4. Porto Alegre: Sulina, 2015. 120 p.

MYERS, B. Taxonomies of visual programming and program visualization. **Journal of Visual Languages & Computing**, v. 1, n. 1, p. 97–123, 1990. Disponível em: https://www.sciencedirect.com/science/article/pii/S1045926X05800369#:~:text=There%20has%20been%20great%20interest,and%20understanding%20of%20computer%20systems.&text=These%20systems%20are%20organized%20by%20classifying%20them%20into%20three%20different%20taxonomies.. Acesso em: 08 ago. 2020.

NEGROPONTE, N. **The Architecture Machine: Toward a More Human Environment**. Cambridge, Mass: The MIT Press, 1970.

NEUFERT, P. Arte de projetar em arquitetura. 17ª ed. São Paulo: Gustavo Gilli, 2012. 618 p.

NORBERG-SCHULZ, C. O fenômeno do lugar. Architectural Association Quarterly 8, n.4, 1976. Pp 4-10. *In*: NESBITT, Kate [Org.]. **Uma Nova Agenda Para a Arquitetura - Coleção Face Norte**. Edição: 2ª. São Paulo: Cosac & Naify, 2008. pp. 443–461.

OESTERREICH, T.; TEUTEBERG, F. Understanding the implications of digitisation and automation in the context of Industry 4.0: A triangulation approach and elements of a research agenda for the construction industry. **Computers in Industry**, v. 83, pp. 121–139, 2016. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0166361516301944. Acesso em: 08 go. 2020. DOI: https://doi.org/10.1016/j.compind.2016.09.006

PALLASMAA, Juhani; SALVATERRA, Alexandre. **Habitar**. Edição: 1. Barcelona: Editora Gustavo Gili, 2017. 125 p.

PANERO, J.; ZELNIK, M. Dimensionamento humano para espaços interiores. Gustavo Gili, 1ª Edição, 2016. 320 p.

PREIDEL, C; BORRMANN, A. Towards code compliance checking on the basis of a visual programming language. **Journal of Information Technology in Construction (ITcon)**, v. 21, n. 25, pp. 402–421, 2016. Disponível em: https://www.itcon.org/paper/2016/25. Acesso em: 08 ago. 2020.

RACEC, E.; BUDULAN, S.; VELLIDO, A. **Computational Intelligence in architectural and interior design : a state-ofthe-art and outlook on the field**. Barcelona: Universitat Politècnica de Catalunya, 2016. Disponível em: https://www.cs.upc.edu/~avellido/research/RacecBudulanVellido_CCIA16.pdf. Acesso em 08 ago. 2020.

REITMAN, W. R., Cognition and Thought. Nova York: John Wiley, 1965. *apud* MITCHELL, W. J. The Theoretical Foundation of Computer-Aided Architectural Design: **Environment and Planning B: Planning and Design**, 1975. Disponível em: https://journals.sagepub.com/doi/10.1068/b020127. Acesso em: 25 abr. 2020.

RITTEL, H. W. J. On the Planning Crisis: Systems Analysis of the First and Second Generations. In **Bedrifsøkomen,** 8, 1972. pp. 390-396. Disponível em:http://www.ask-force.org/web/Discourse/Rittel-Planning-Crisis-First-Second-Generation-1972.pdf. Acesso em: 25 abr. 2020.

RITTEL, H. W. J.; WEBBER, M. M. Dilemmas in a General Theory of Planning. **Policy Sciences**, v. 4, n. 2, p. 155–169, 1973. Disponível em: https://link.springer.com/article/10.1007/BF01405730. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1007/BF01405730

ROCHA, A. J. M. **Architecture theory, 1960-1980: emergence of a computational perspective**. Tese (Doutorado) — Massachusetts Institute of Technology, 2004. Disponível em: https://dspace.mit.edu/handle/1721.1/28316. Acesso: 09 ago. 2020.

RODRIGUES, E; GASPAR, A; GOMES, Á. An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, part 1: methodology. **Comput. Aided Des. 45,** 2013. Pp 887-897. Disponível em:

https://www.sciencedirect.com/science/article/abs/pii/S0010448513000031?via%3Dihub . Acesso: 04 jul. 2021. DOI: https://doi.org/10.1016/j.cad.2013.01.001

RODRIGUES, E; GASPAR, A; GOMES, Á. An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, part 2: validation and performance tests. **Comput. Aided Des. 45,** 2013. Pp 898–910. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0010448513000055?via%3Dihub . Acesso: 04 jul. 2021. DOI: https://doi.org/10.1016/j.cad.2013.01.003.ROWE, P. **Design Thinking**. Boston: MIT Press, 1991, 227 p.

RUSSEL, S; NORVIG, P. **Artificial Intelligence**: A modern approach. Artificial Intelligence. Prentice-Hall, Englewood Cliffs, Citeseer, v.25, 1995. 928 p.

SANCHEZ, S. *et al.* Constraint-based 3d-object layout using a genetic algorithm. **Intelligenza Artificiale - IA**, 2003. Disponível em: https://www.researchgate.net/publication/245776169_Constraint-based_3d-object_layout_using_a_genetic_algorithm. Acesso em: 08 ago. 2020.

SCHWAB, K. **The Fourth Industrial Revolution: what it means, how to respond**. World Economic Forum, 2016. Disponível em: https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/. Acesso em: 28 jan. 2020.

SIMON, H. A. **The Sciences of the Artificial, Third Edition | The MIT Press**. 3. ed. Massachusetts: The MIT Press, 1996. Disponível em: https://mitpress.mit.edu/books/sciences-artificial. Acesso em: 26 abr. 2020.

ŠIJAKOVIĆ, M.; PERIĆ, A. Symbiotic architecture: Redefinition of recycling design principles. **Frontiers of Architectural Research**, v. 7, n. 1, p. 67–79, 2018. Disponível em: https://www.sciencedirect.com/science/article/pii/S2095263518300013. Acesso em: 08 ago. 2020.

STINY, G; MITCHELL, W J. The Palladian grammar. **Environment and Planning B: Planning and Design**, v. 5, n. 1, p. 5–18, 1978. Disponível em: https://journals.sagepub.com/doi/10.1068/b050005. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1068/b050005

SÖNMEZ, N. O. A review of the use of examples for automating architectural design tasks. **Computer-Aided Design**, v. 96, pp. 13–30, 2018. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0010448517301781. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1016/j.cad.2017.10.005

SUTHERLAND, I. E. Sketch pad a man-machine graphical communication system. *In*: **Proceedings of the SHARE design automation workshop**. New York, NY, USA: Association for Computing Machinery, 1964, pp. 6.329–6.346. (DAC '64). Disponível em: https://doi.org/10.1145/800265.810742. Acesso em: 26 jul. 2020.

TESLA. **Autopilot and Full Senf-Driving Capability**. Disponível em: https://www.tesla.com/support/autopilot. Acesso em: 11 de abril de 2021.

_____. **Future of Driving**. Disponível em: https://www.tesla.com/autopilot. Acesso em: 11 de abril de 2021.

TURING, A. M. I. **Computing Machinery And Intelligence**. Mind, v. LIX, n. 236, pp. 433–460, 1950. Disponível em: https://www.csee.umbc.edu/courses/471/papers/turing.pdf. Acesso em 08 ago. 2020.

VELOSO, P.; CELANI, G.; SCHEEREN, R. From the generation of layouts to the production of construction documents: An application in the customization of apartment plans. **Automation in Construction**, v. 96, p. 224–235, 2018. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0926580518304734. Acesso em: 08 ago. 2020.

WAHBEH, W. Building skins, parametric design tools and BIM platforms. In: 12TH CONFERENCE OF ADVANCED BUILDING SKINS, 2017, Bern. **Anais de conferência da 12th Conference of Advanced Building Skins**, 2017, pp. 1104–1111. Disponível em: https://www.researchgate.net/publication/320244444_Building_skins_parametric_design_tools_and_BIM_platforms. Acesso em: 19 out. 2019.

WANG, L.; GAO, R.; VÁNCZA, J.; et al. Symbiotic human-robot collaborative assembly. **CIRP Annals**, v. 68, n. 2, p. 701–726, 2019. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0007850619301593. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1016/j.cirp.2019.05.002

WEF - WORLD ECONOMIC FORUM. **Shaping the Future of Construction – A Landscape in Transformation: An Introduction**. Report: Committed to Improving the State of the World. Geneva, 2016. Disponível em: http://www3.weforum.org/docs/WEF_Shaping_the_Future_of_Construction.pdf. Acesso em: 28 jan. 2020.

WONG, S. S. Y.; CHAN, K. C. C. EvoArch: An evolutionary algorithm for architectural layout design. Computer-Aided Design, v. 41, n. 9, pp. 649–667, 2009. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0010448509001109?via%3Dihub. Acesso em: 08 ago. 2020. DOI: https://doi.org/10.1016/j.cad.2009.04.005

YIN, X *et al.* Building information modelling for off-site construction: Review and future directions. **Automation in Construction**, v. 101, p. 72–91, 2019. Disponível em: https://www.researchgate.net/publication/330566168_Building_Information_Modelling_for_Off-site_Construction_Review_and_Future_Directions. Acesso em: 08 ago. 2020. DOI: 10.1016/j.autcon.2019.01.010

YU, L. *et al.* Make it home: automatic optimization of furniture arrangement. . In: SIGGRAPH '11, 2011, Vancouver. **Anais SIGGRAPH '11**, v. 30, n. 4, 2011, pp. 86:1–86:12, 2011. Disponível em: https://dl.acm.org/doi/10.1145/2010324.1964981. Acesso em: 08 ago. 2020.