PALAVRAS-CHAVE
Otimização,
Design computacional,
Caixa-preta,
Código aberto,
Desempenho

PALABRAS CLAVE
Optimización,
Diseño computacional,
Caja negra,
Código abierto,
Performance

KEYWORDS
Optimization,
Computational design,
Black-box,
Open-source,
Performance

ABRIR A CAIXA-PRETA E REFLETIR SOBRE MÉTODOS DO FAZER. OTIMIZAÇÃO DE PROJETO ORIENTADO AO DESEMPENHO EM ARQUITETURA

ABRIR LA CAJA NEGRA Y REFLEXIONAR SOBRE LOS MÉTODOS DE HACER. OPTIMIZACIÓN DEL DISEÑO BASADO EN DESEMPEÑO EN LA ARQUITECTURA

OPEN THE BLACK BOX AND THINK ABOUT THE METHODS. OPTIMIZATION IN PERFORMANCE-BASED ARCHITECTURAL DESIGN

> **DYEGO DA SILVA DIGIANDOMENICO[1], GABRIELE DO ROSARIO LANDIM[2] E CLÁUDIO FABIANO MOTTA TOLEDO[3]**

1 Universidade Católica de Pernambuco
2 Universidade Federal da Paraíba
3 Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

RECEBIDO
10 DE MAIO DE 2022

ACEITO
20 DE MARÇO DE 2023

EL CONTENIDO DE ESTE ARTÍCULO ESTÁ BAJO LICENCIA DE ACCESO ABIERTO CC BY-NC-ND 2.5 AR

> **CÓMO CITAR ESTE ARTIGO** *(NORMAS APA):*
Digiandomenico, D. da S., Landim, G. do R. e Motta Toledo, C. F. (Noviembre de 2022 – Abril de 2023). Abrir a caixa-preta e refletir sobre métodos do fazer. Otimização de projeto orientado ao desempenho em arquitetura. [Archivo PDF]. *AREA, 29*(1), pp. 1-18. Recuperado de https://www.area.fadu.uba.ar/wp-content/uploads/AREA2901/2901_digiandomenico_et_al.pdf

RESUMO

Algoritmos de otimização integrados ao projeto computacional têm despertado cada vez mais interesse em arquitetura e urbanismo. Chamamos atenção para uma prática frequente na área de projeto computacional: publicar pesquisas em otimização sem explicitar, detalhar os compartilhar os conjuntos de procedimentos que levaram aos resultados otimizados. A falta de compartilhamento dos métodos e procedimentos aplicados à otimização de projetos de arquitetura pode produzir conclusões inseguras, desamparar discussões sobre as soluções e inviabilizar sua replicabilidade, além de dificultar a análise crítica do processo. Discutimos detalhes emergidos ao transitar entre a aplicabilidade técnica de um algoritmo de otimização aberto e o que aprendemos deste processo de maneira mais ampla e crítica para a produção no sul global.

RESUMEN

Los algoritmos de optimización integrados al diseño computacional han despertado cada vez más interés en arquitectura y urbanismo. Llamamos la atención sobre una práctica frecuente en diseño computacional: publicar investigaciones sobre optimización sin explicar, detallar o compartir los conjuntos de procedimientos que llevaron a los resultados optimizados. La falta de compartición de métodos y procedimientos aplicados a la optimización de proyectos arquitectónicos puede producir conclusiones inseguras, abandonar las discusiones sobre las soluciones y hacer inviable su replicabilidad, además de dificultar el análisis crítico del proceso. Discutimos los detalles que surgieron al moverse entre la aplicabilidad técnica de un algoritmo de optimización abierto y lo que aprendemos de este proceso de manera más amplia y crítica para la producción en el sur global.

ABSTRACT

Optimization algorithms integrated into computational design have aroused more and more interest in architecture and urbanism. We draw attention to a frequent practice in computational design: publishing research on optimization without explaining, detailing, or sharing the sets of procedures that led to the optimized results. The lack of sharing of methods and procedures applied to the optimization of architectural projects can produce insecure conclusions, abandon discussions about the solutions and make their replicability unfeasible, making the critical analysis of the process difficult. We discuss details that emerged when moving between the technical applicability of an open optimization algorithm and what we learn from this process more broadly and critically for production in the global south.

> SOBRE OS AUTORES E A AUTORA

Dyego da Silva Digiandomenico. Antes de tudo, cria da periferia. Mestre em Arquitetura, Urbanismo e Tecnologia pela Universidade de São Paulo (IAU-USP). Professor na Universidade Católica de Pernambuco (UNICAP). Foi coordenador do curso de Pós-Graduação Internacional em Tecnologias do Design (UNICAP). Foi membro do grupo de pesquisa Núcleo de Estudos das Espacialidades Contemporâneas (NEC-USP). Realizou estágio de pesquisa no grupo Algorithmic Design for Architecture (ADA) do Instituto Superior Técnico da Universidade de Lisboa. Foi Docente Convidado no curso de graduação de Arquitetura e Urbanismo da Universidade de São Paulo (IAU-USP) nas disciplinas Arquitetura, Design Paramétrico e Fabricação Digital e Projeto 3A.
✉ <dyego.sd@gmail.com>

Gabriele do Rosario Landim. Doutoranda na pós-graduação em Arquitetura e Urbanismo pela Universidade Federal da Paraíba (UFPB). Mestre pelo Instituto de Arquitetura e Urbanismo da Universidade de São Paulo (IAU-USP) com pesquisa na área do Projeto Computacional e arquiteta e urbanista pelo Centro Universitário Belas Artes de São Paulo como bolsista PROUNI (Programa Universidade para Todos). Atualmente é docente nos cursos de Pós-graduação em Arquitetura Digital e Projetos Paramétricos da Belas Artes de São Paulo e Design Paramétrico em Arquitetura da PUC Minas. Foi professora (2019-2021) no curso de Arquitetura e Urbanismo da Universidade Federal da Integração Latino-Americana (UNILA). Entre os principais temas de atuação, pesquisa e interesse estão: design computacional, programação, projetos orientados ao desempenho, teoria crítica da tecnologia e pensamentos não-hegemônicos.
✉ <gabriele.landim@gmail.com>

Claudio Fabiano Motta Toledo. Doutor em Engenharia Elétrica pela Universidade Estadual de Campinas e Livre Docente em Sistemas de Computação pela Universidade de São Paulo. Realizou pós-doutorado no Computer Science and Artificial Intelligence Laboratory (CSAIL) do Massachusetts Institute of Technology (MIT) (2014-2015). Professor na Universidade de São Paulo desde 2011. Tem experiência na área de Ciência da Computação com ênfase em Sistemas Evolutivos e Programação Matemática aplicados a problemas de planejamento de produção, planejamento de rotas para veículos aéreos não tripulados, geração procedural de conteúdos para jogos digitais, remoção de ruídos em imagens, entre outros.
✉ <claudio@icmc.usp.br>

Introdução

Um dos usos que mais geram interesse em pesquisadores e arquitetos em relação a aplicação de algoritmos e da computação na área, é a possibilidade de testar muitas alternativas de projeto e verificar suas eficiências. Este fluxo de trabalho integra modelagem paramétrica, simulação e otimização para definir quais as melhores alternativas de projeto para um problema que se quer verificar solução. É possível verificar desempenhos em simulações estruturais, de consumo energético, ou de análises de iluminância, por exemplo. É comum nas etapas do projeto orientado ao desempenho (ou performance), o uso de algoritmos de *otimização caixa-preta*. Na matemática, uma otimização *caixa-preta* se refere ao uso de *softwares* que podem simular um comportamento que deseja-se verificar –como por exemplo a tensão e deslocamento de forças em uma viga– sem que o projetista necessariamente descreva o problema de simulação em uma fórmula matemática. Desta maneira, os mecanismos presentes nos *softwares* calculam as simulações e fornecem os resultados em valores numéricos para os projetistas, sem suas fórmulas matemáticas correspondentes (Wortmann e Nannicini, 2017).
No entanto, com uma vista mais abrangente deste processo de projeto e para além desta especificidade técnica, chamamos de *caixa-preta*, um método tomador de decisão que não explicita seu algoritmo ou detalhes precisos de como operam os dados que retornam como soluções "ótimas". Por exemplo, é popular o uso do otimizador Galapagos, nativo da linguagem de programação visual *Grasshopper*.
O *plugin* permite aplicar um método de recozimento simulado (*simulated annealing*) e um evolucionário (*evolutionary solver*). Existem variedades de decisões e estratégias na definição destes algoritmos que podem impactar tanto os resultados da otimização quanto sua adequação ao problema de projeto. No caso do algoritmo evolucionário é possível tomar decisões sobre tamanho inicial da população, indivíduos iniciais descartados, taxa de mutação, estratégias de cruzamento, taxa de elitismo, entre outros. Em um algoritmo caixa preta, não podemos verificar ao certo como opera o método escolhido.
Este artigo chama atenção para o fato de que as pesquisas que exploram o projeto orientado ao desempenho em arquitetura –e projetos do design computacional em geral– têm frequentemente publicado trabalhos sem compartilhar ou detalhar os conjuntos de procedimentos que levaram a tais resultados. Há uma emergência aparente de positivismo sobre esses projetos. Mesmo onde resultados inovadores de desempenho são atingidos, a não abertura dos métodos fornece pouca ou nenhuma informação para que a comunidade científica possa desenvolver uma análise crítica sobre os procedimentos empregados e resultados apresentados. Destacam-se as inovações, e sem informações abertas sobre os algoritmos, dificulta-se a reflexão crítica sobre a coerência da tecnologia aplicada na resolução do problema que se deseja solucionar.
Tal condição impossibilita reflexões seguras sobre as características dos processos e otimizações produzidas, torna-se reduzida a reflexão em arquitetura sobre quais métodos são mais apropriados para a otimização de determinados problemas do projeto arquitetônico. Além disso, desampara o fluxo de trocas de conhecimentos, uma vez que a possibilidade de replicar o experimento e discutir aprofundadamente a respeito das soluções encontradas tornam-se reduzidas. Também, acentua-se o cenário de que pesquisas de ponta são produzidas em centros de inovação, geralmente localizados no norte global, enquanto países no capitalismo dependente, como o Brasil, reproduzem as técnicas *caixa-preta* (ou de código fechado) sem terem desenvolvido olhar crítico sobre a adequação entre métodos e busca por soluções de problemas.
Diferente dos métodos não-computacionais de processo de projeto em arquitetura, nos quais o objeto arquitetônico se manifesta através da representação pelo desenho –baseado em conhecimentos implícitos– no design computacional existe a explicitação dos processos cognitivos, baseados na capacidade de configurar, programar e implementar os algoritmos do fluxo de projeto, ao mesmo tempo que interage com sua representação visual. Esta transição entre paradigmas amplia consideravelmente a capacidade de compartilhamento dos processos e soluções de projetos, agora declarados como algoritmos. Contudo, esta cultura não se manifesta na maior parte das pesquisas publicadas na área.
Diante deste contexto de escassez de referências que abrem seus processos com

transparência, este artigo apresenta um experimento comparativo de otimização de projeto orientado ao desempenho em arquitetura. Apresentamos uma reimplementação do projeto *Vancouver Academic Building*, do escritório *Perkins+Will* Atlanta, que utiliza o método *Design Space Construction* (Haymaker et al., 2018) e a comparação deste com um algoritmo genético, através do *plugin ArchOptimum,* desenvolvido pelos autores (Landim, Digiandomenico, Amaro, Pratschke, Tramontano e Toledo, 2017). O objetivo aqui é transitar entre a aplicabilidade técnica de um algoritmo de otimização aberto, e o que se aprendeu deste processo de maneira mais ampla e crítica frente a área do projeto orientado ao desempenho. Nossa finalidade inicial foi identificar quais as potencialidades e limites dos métodos e qual deles é mais eficiente na otimização do projeto de arquitetura estudado. Além disso, buscamos saber quais reflexões emergem quando os projetistas desenvolvem ou operam métodos de otimização de código-fonte aberto. Em outras palavras, neste trabalho tentamos abrir certas caixas-pretas e ver o que elas nos despertam a respeito do design computacional e otimização de projetos em arquitetura. Assim, a partir das observações apresentadas até aqui, nos perguntamos se estimular a abertura dos métodos de otimização no design computacional em arquitetura pode tornar a otimização mais eficiente, verificável e reproduzível.

Otimização de projeto em arquitetura

O fluxo de trabalho que integra modelagem paramétrica, simulações de desempenho e algoritmos de otimização, conhecido como projeto orientado ao desempenho –em inglês *performance-driven architectural design*– é definido como um método capaz de estabelecer relações de compensação entre objetivos concorrentes e testar seus impactos em muitas alternativas de projetos. Dessa maneira, a otimização integrada de múltiplas variáveis pode guiar e aperfeiçoar o projeto (Shi e Yang, 2013). A literatura internacional têm apontado que a otimização tem grande relevância na arquitetura, engenharia e construção (AEC), pois ela pode aumentar consideravelmente os padrões de eficiência dos resultados nas respostas às diversas demandas e objetivos do projeto (Li, Liu e Peng, 2020). São geralmente usadas em estudos de massa, orientação e implantação de edificações, projeto de fachadas, sistemas estruturais, qualidade do ar, eficiência energética, sistemas de aquecimento, ventilação e ar-condicionado, iluminação natural e artificial, logística, acústica, ciclo de vida, custos, entre outros (Shi, 2010; Machairas, Tsangrassoulis e Axarli, 2014; Touloupaki e Theodosiou, 2017). O projeto orientado ao desempenho se conecta com os aspectos que podem ser quantificáveis. Vale lembrar que este método reduz uma série de combinações possíveis a inadequadas ou insuficientes, e em certa medida desconsidera que a proposta de uma solução ótima ignora o estado impermanente e subjetivo que compõem o caráter qualitativo da resposta arquitetônica.

Considerações sobre a escolha do método de otimização: Algoritmo Genético

Entre as metaheurísticas –heurísticas para resolver de forma genérica problemas de otimização– os Algoritmos Genéticos (AGs) é um dos métodos mais populares em problemas de arquitetura. Algoritmos Evolutivos, especialmente AGs, são conhecidos por serem robustos na exploração de espaço de design para uma ampla gama de problemas de otimização (Attia, Hamdy, O'Brien e Carlucci, 2013). No entanto, existem outros métodos de otimização baseados em algoritmos que dependem de menos aleatoriedade e mais de fundamentos matemáticos, portanto, podem ser mais eficientes, uma vez que encontram soluções com menor número de iterações (Holmström, 2008; Rios, 2009; Costa, Nannicini, Schroepfer e Wortmann, 2015). O que se pode afirmar é que nenhum método de otimização tem o melhor desempenho em todos os problemas possíveis. Assim, é relevante apontar que o AG é um processo heurístico, portanto, pode não necessariamente gerar soluções ótimas globais. A vasta documentação do conjunto de procedimentos do AG, assim como a quantidade de exemplos práticos em arquitetura aplicando esse método em seu processo de projeto, motivaram o enfoque desta pesquisa.

Experimento

A partir do uso da computação, o processo de projeto ganha complexidade, uma vez que tenta obter respostas eficientes às crescentes solicitações na AEC, dentre elas, a maximização da eficiência energética do projeto em relação a sua forma e componentes construtivos. Os sistemas de classificação de sustentabilidade, desenvolvidos em todo o mundo nas últimas décadas, alavancaram pesquisas de otimização em AEC por envolverem requisitos de projeto difíceis de calcular, onde muitas vezes é necessário balancear fatores concorrentes como, por exemplo, a maximização do uso da luz natural e a minimização do calor por radiação solar. Diante desse panorama, muitas vezes a varredura do espaço de design, sem a aplicação de um método de otimização, torna-se complexo e custoso para as equipes de projeto. Considerando este contexto, o presente experimento apresenta reflexões a partir do desenvolvimento e aplicação do método em código aberto que propõe realizar três contribuições:

1. Demonstrar a implementação do método de otimização por AG como recurso de buscas dentro de um grande espaço de design, operando junto com simuladores de eficiência energética.
2. Abrir os métodos de otimização implementados, assim como detalhar e comparar os dois métodos de otimização aplicados no mesmo estudo de caso, abrindo a possibilidade dos métodos investigados serem verificados e adaptados.
3. Fornecer recursos para a reflexão prática, teórica e crítica a respeito dos potenciais e limites da otimização de projetos orientados ao desempenho.

Problema arquitetônico

O problema corresponde ao estudo de caso *Vancouver Academic Building*, do escritório de arquitetura *Perkins+Will* da cidade de Atlanta, Geórgia, nos Estados Unidos, *é* balancear objetivos conflitantes e atender pré-requisitos de certificações internacionais de sustentabilidade. São eles, a carga de resfriamento e aquecimento *Standard* da *International Passive House Association* (IPHA) e a maximização do uso da luz natural disponível definido pelo *Leadership in Energy and Environmental Design* (LEED). Portanto, a otimização do projeto se concentra na fachada do edifício compondo a distribuição de painéis pré-moldados, brises e a definição dos valores de ganho de calor aceitáveis para o vidro e para as paredes opacas. A tradução dos objetivos da otimização em sua forma quantificável está descrita na tabela abaixo:

Tabela 1. Objetivos da otimização

METAS	OBSERVAÇÕES		MÉTRICAS	RESTRIÇÕES
Quais são as metas experienciais, ecológicas ou econômicas específicas? Qualitativas.			Para combinar várias Metas e Restrições em um único valor, elas devem ser quantificáveis.	Restrições são requisitos. Alternativas que violam uma restrição não são viáveis.
Iluminância dentro do aceitável	% de área dentro do intervalo aceitável (entre 300 e 3.000 lux/m^2).	Maximizar.	%	Implícitas nas próprias amplitudes de variáveis.
Iluminância acima do aceitável	% de área acima do intervalo aceitável (entre 300 e 3.000 lux/m^2).	Minimizar.	%	Aplicadas via DoE. (*Design of Experiments*)
Iluminância abaixo do aceitável	% de área abaixo do intervalo aceitável (entre 300 e 3.000 lux/m^2).	Minimizar.	%	
Resfriamento (kWh/m^2)	Eficiência energética.	Minimizar o consumo de energia.	kWh/m^2	
Aquecimento (kWh/m^2)	Eficiência energética.	Minimizar o consumo de energia.	kWh/m^2	
WWR (*Window Wall Ratio*)	Divisão da a área total de vidro (janela) pela área total da parede. Influencia nos cálculos do Valor U. Uma janela média, com vidro simples, tem U de 5,0 W/m^{2}C, ou 6,0 se tiver uma estrutura de metal. Enquanto paredes externas tem valores de 0,5 a 2,0. Quanto menor o valor, menos calor é transferido pelo material. Isso significa que as janelas estão deixando passar mais que o dobro da energia do que as paredes.	Não usa na função objetivo. Dado para observação. Provável índice requerido na Passive House.	Número inteiro.	

Fonte: elaboração própria.

Os autores envolvidos no experimento tiveram autorização do escritório *Perkins+Will* para a utilização do projeto, que concedeu acesso a parte dos dados do processo de projeto. A autorização foi concedida pelos arquitetos pesquisadores John Haymaker, diretor de pesquisa da *Perkins+Will* San Francisco, professor na *Georgia University of Technology*, Estados Unidos, e Marcelo Bernal, diretor de pesquisa da *Perkins+Will* Atlanta e professor na *Universidad* Técnica Federico Santa *María*, Chile. Tais dados englobam o fluxo de trabalho do processo, informações sobre os critérios de decisão do projeto, métricas de desempenho e os métodos de otimização. A implementação do estudo parte do modelo paramétrico, compartilhado no material disponibilizado pelo escritório de arquitetura *Perkins+Will* no *workshop Design Space Construction: Defining, optimizing, and communicating performance-based building design spaces*, realizado no congresso *Association for Computer-Aided Design in Architecture,* ACADIA 2017.
O projeto *Vancouver Academic Building* foi selecionado como estudo de caso por integrar concepção, simulação e otimização de desempenho em arquitetura, assim, possui características processuais que colaboram com as investigações da presente pesquisa.

Abordagem Design Space Construction (DSC)

O fluxo de trabalho aplicado no estudo de caso se baseia na abordagem *Design Space Construction* (DSC). A metodologia oferece um modelo computacional que dá suporte para definição de objetivos de projeto, geração de modelos paramétricos e simulações de alternativas de projeto. Essa estrutura guia o projetista por meio de processos de formulação de problemas, geração de alternativas de projetos, análise de impacto e avaliação de valor das alternativas obtidas (Haymaker et al., 2018). O estudo de caso segue o quadro conceitual de implementação proposto por Haymaker et al. (2018), descrito a seguir.

Modelo Paramétrico

Dentre o material disponibilizado, os itens fundamentais são, um arquivo *Rhinoceros* (.3dm) e uma arquivo *Grasshopper* (.gh), que geram o modelo paramétrico do projeto. O arquivo *Rhinoceros* contém a representação computacional volumétrica do projeto do estudo de caso. Os dados do projeto estão reunidos em cinco conjuntos de camadas, que reúnem o volume dos elementos das lajes, áreas ocupadas, volumes internos, fachadas estáticas e curvas da silhueta do volume.
O modelo paramétrico do projeto se refere à geração da geometria dos diferentes tipos de painéis e seus possíveis posicionamentos em cada trecho de fachada. O modelo paramétrico será responsável por gerar iterativamente as alternativas de projeto e o algoritmo de simulação atrelado à ele avalia o modelo analítico com os atributos necessários para cada análise de simulação.

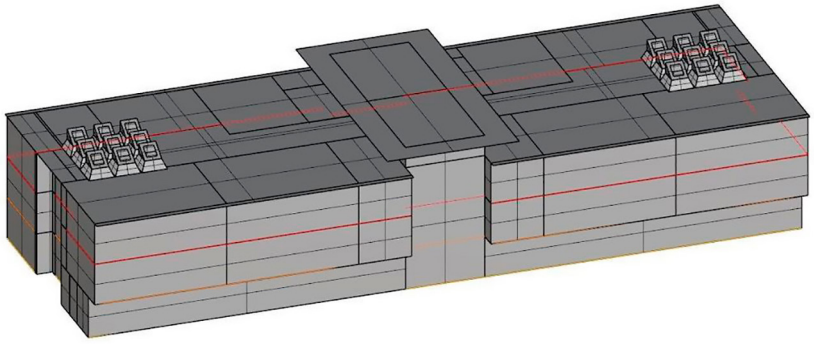


Figura 1
Modelo base no *Rhinoceros* para a criação do modelo paramétrico no *Grasshopper*.
Fonte: elaboração própria.

Simuladores de performance

Ladybug e Honeybee, são *plugins* de código aberto para *Grasshopper*. Integram o *Grasshopper* com conjuntos de dados ambientais validados e motores de simulação com a finalidade de explorar e avaliar o desempenho ambiental. Portanto, são responsáveis por intermediar a comunicação entre o modelo paramétrico no *Grasshopper* com o software que calcula a simulação definida.
O *Ladybug* torna possível a importação de arquivos padrão do *WeatherPlus Weather* (.EPW). O *Honeybee* permite a realização de simulações de iluminação natural com o software Radiance e DAYSIM, modelos de energia usando o *OpenStudio* e *EnergyPlus* e fluxos de calor através *do Berkeley Lab Therm*.
EnergyPlus é um programa de simulação de energia capaz de avaliar consumo energético para aquecimento e resfriamento, ventilação, iluminação e cargas. Além disso, o programa é capaz de incorporar durante a simulação questões como cálculo de horas específicas, sendo que as etapas

de tempo podem ser pré-definidas para análise de zonas térmicas do ambiente. O EnergyPlus é gratuito e de código aberto, financiado e desenvolvido pelo Building Technologies Office (BTO) do United States Department of Energy (DOE).
Open Studio reúne um conjunto de programas que suportam a modelagem de energia do edifício através do EnergyPlus e Radiance acrescendo a interface gráfica de visualização no seu funcionamento, que também pode ser inicializado por entradas em linguagens textuais, como C++, Ruby e C#, é gratuito e de código aberto, também financiado e desenvolvido pelo Building Technologies Office (BTO) do United States Department of Energy (DOE).
Radiance, reúne um conjunto de programas de análise e visualização focado na iluminação, utilizado por profissionais da construção e pesquisadores para simulação de luminosidade e qualidade visual. Os cálculos realizados pelo programa incluem índices de brilho, iluminância e cor, ou seja, radiância espectral e irradiância. Radiance foi desenvolvido por Greg Ward no Building Technologies Program na Environmental Energy Technologies Division do Lawrence Berkeley National Laboratory em Berkeley.
DAYSIM, é um programa de análise capaz de modelar a iluminação natural dentro de um período específico, medindo a quantidade de luz no interior e entorno dos edifícios operando baseado no Radiance. O desenvolvimento do programa foi coordenado por Christoph Reinhart e as implementações foram realizadas no Fraunhofer Institute for Solar Energy Systems (ISE), Harvard University, Massachusetts Institute of Technology (MIT) e no National Research Council (NRC) do Canada.

Otimizador

No projeto original, o escritório utiliza uma função matemática que atribui pesos a cada um dos objetivos. O resultado dessa função é o valor *fitness* de cada alternativa de projeto. Essa função é escrita na planilha de dados resultantes das simulações de todas as alternativas de projeto.
O objetivo do método é explorar a combinação de parâmetros que melhor atende o balanceamento dos seguintes itens: Iluminação (*lighting*) dentro da edificação, o resultado equivale a porcentagem da área dentro da faixa aceitável; Resfriamento (*cooling*) em kWh/m²; e Aquecimento (*heating*) em kWh/m².

Para poder comparar diretamente os resultados das simulações que estão em diferentes unidades de medida, é necessário normalizar os resultados (quando todos os valores ficam na mesma escala, de 0 a 1) através da função *f ator – min(f ator) / max(f ator) – min(f ator)*. A função objetivo aplicada da otimização considera então:

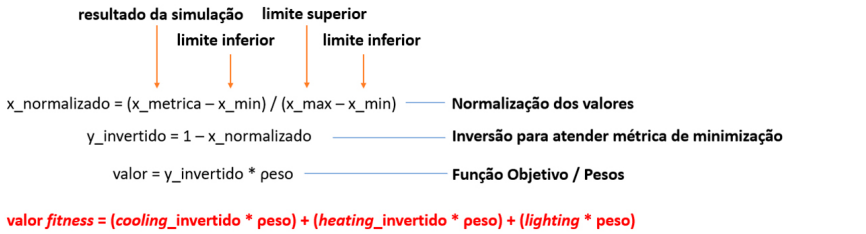


Figura 2
Função Objetivo.
Fonte: elaboração própria.

Os pesos dos fatores são definidos em um acordo entre a equipe de projeto e clientes. Por exemplo: 0,3 para *cooling*, 0,2 para *heating*, 0,3 para *lighting*, pesos somam cem por cento.

Visualização de dados

Após a realização das simulações de todas as alternativas de projeto definidas na planilha, o conjunto de dados e imagens de cada alternativa (gravados pelo *plugin Colibri* em um arquivo .csv e imagens .jpg do resultado de cada iteração) podem ser carregados na ferramenta web Designer Explorer. Design Explorer é uma ferramenta de código aberto desenvolvida pelo CORE Studio, para a exploração da representação do espaço de busca. A partir do arquivo data.csv a ferramenta gera a visualização 2D do espaço de busca do projeto através da plotagem do Gráfico de Coordenadas Paralelas (*Parallel Coordinates Plot*) (Figuras 10 e 11 na página 14).

Conceito do fluxo da implementação

A partir dos recursos apresentados, o fluxo da implementação do DSC pode ser conceitualmente descrito em quatro etapas:

Etapa 1. Início do fluxo da implementação, onde são definidos os objetivos e parâmetros de entradas de projeto, atrelados a geometria volumétrica do arquivo *Rhinoceros* (.3dm) e a modelagem algorítmica no *Grasshopper*. Nessa etapa, também são estabelecidas as zonas e os intervalos da análise energética.
Etapa 2. O projeto computacional gera as alternativas através de modelos analíticos tridimensionais incluindo os painéis

paramétricos que serão alocados na fachada do edifício. Os modelos analíticos são encaminhados para a etapa posterior.
Etapa 3. São realizadas as simulações para análises e validações das alternativas de projeto.
Etapa 4. Os resultados de desempenho das análises são computados na função objetivo que classifica as melhores alternativas de projeto.

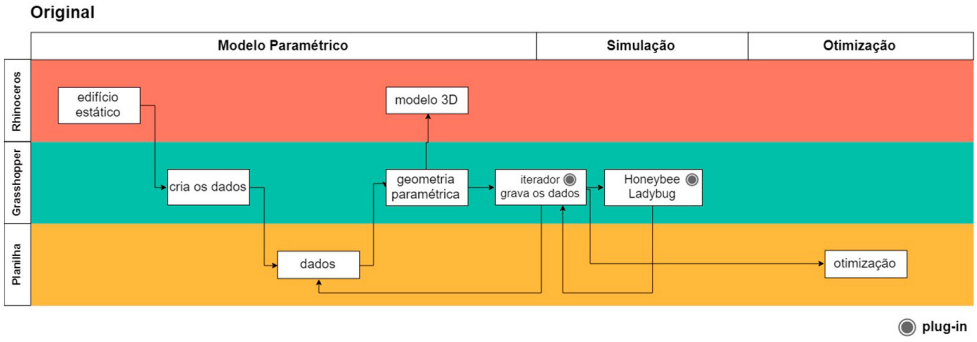
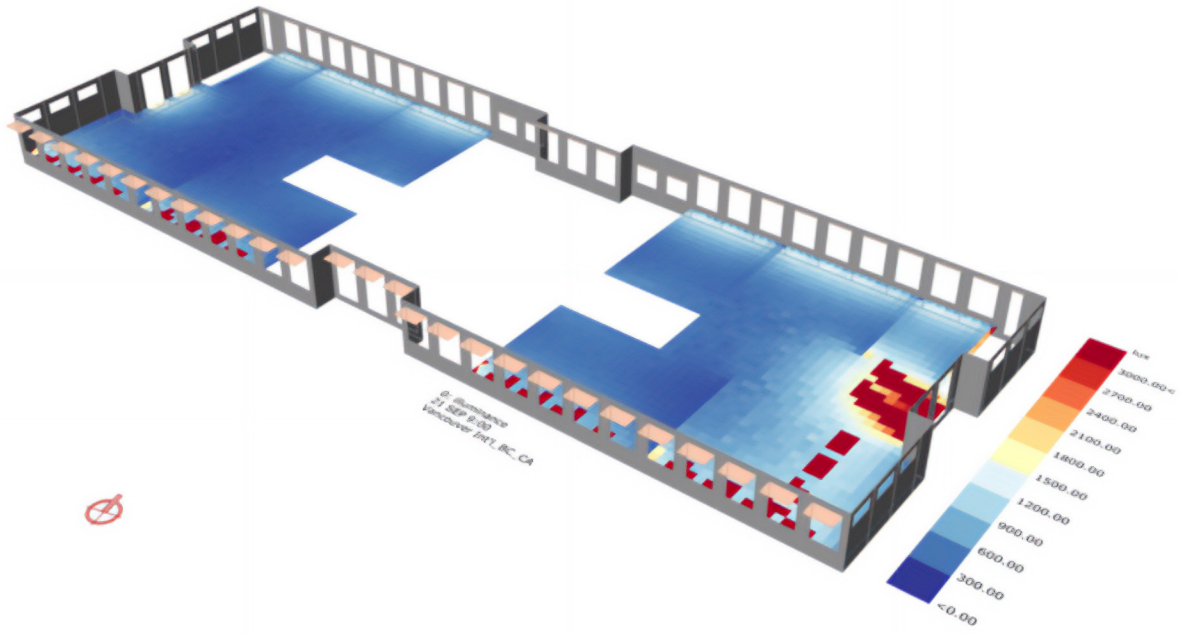


Figura 3
Fluxo de trabalho computacional implementado originalmente pelo escritório *Perkins+Will*.
Fonte: Landim (2019).

Implementação do estudo de caso

A partir desse conjunto de programas e procedimentos, é composto o fluxo de trabalho do estudo de caso. No entanto, mesmo com todos os programas instalados e verificados, o código cedido pelo escritório, desenvolvido no *Grasshopper*, apresentou erros de operação que precisaram ser adaptados e corrigidos. Na fase de implementação foram superados erros causados por versionamento da interface de programação e outros comportamentos intermitentes do código, como rodar em uma máquina e em outra não. Este tipo de erro é identificado na literatura de linguagens de programação visual: "A falta de mecanismos para checar o funcionamento do código progressivamente é um ponto negativo da linguagem [de programação visual]" (Landim, 2019, p. 130). O motivo não foi declarado nas mensagens de erro do *Grasshopper* e o próprio programa não possui suporte a nenhum método automatizado de revisão e testes de códigos.
Foi necessário recriar e avaliar o código por inteiro. Os erros encontrados possuem fontes diversas, entre eles um arquivo solicitado pelos *plugins Ladybug* e *Honeybee* ao programa *EnergyPlus* que não era encontrado. O código fonte dos *plugins* desenvolvido em *Python* foram atualizados manualmente para buscar outra versão do arquivo dentro da biblioteca do programa *EnergyPlus*, sanando esse problema. O código original disponibilizado pelo escritório correspondia à simulação e otimização de apenas um pavimento do edifício. Isso significa que no código cedido não era considerada a interferência da arquitetura de um pavimento no outro. Desse modo, o desempenho do edifício não poderia ser simulado e otimizado como um todo.

Figura 4
Processo DSC na avaliação de uma parte de um pavimento do edifício.
Fonte: Haymaker et al. (2018).



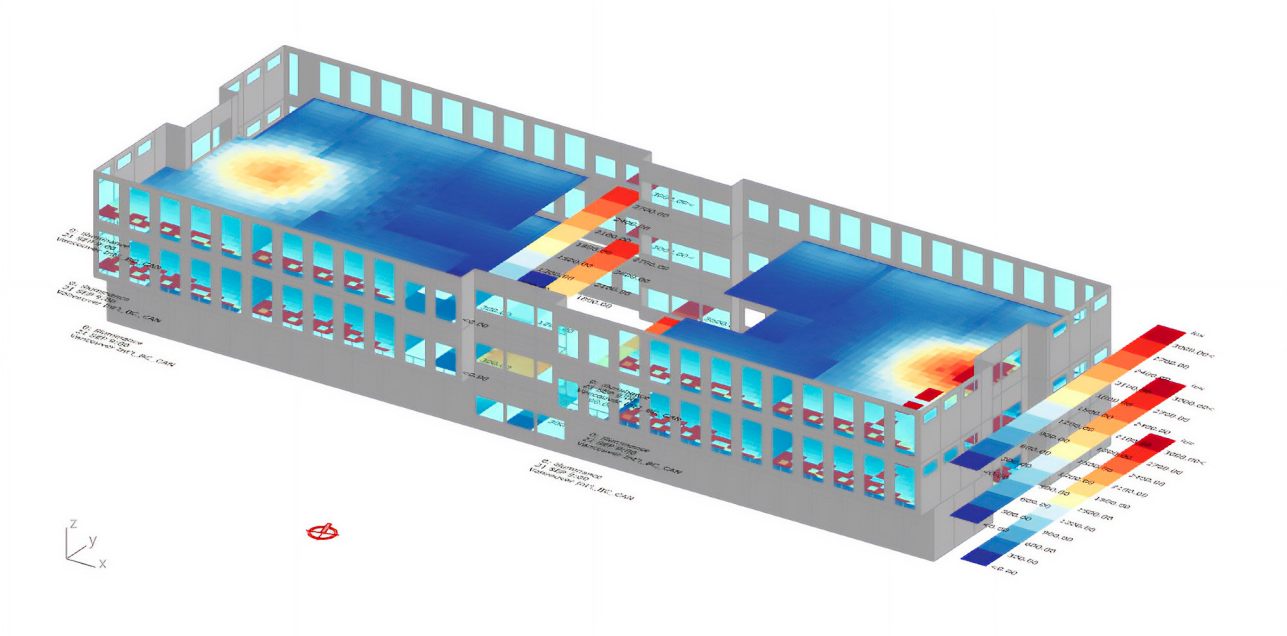


Figura 5
Processo DSC adaptado para realizar a avaliação de todos os pavimentos do edifício.
Fonte: elaboração própria.

Para proceder o estudo de caso do edifício integralmente, é necessário que o código seja capaz de realizar as simulações de eficiência energética em aquecimento, resfriamento, ventilação e iluminação de todos os pavimentos simultaneamente (Figura 5). Foi realizada a reestruturação do código para que a complexidade da simulação e otimização de todos os pavimentos do edifício fosse considerada sem custos computacionais desnecessários por limitações da linguagem de programação visual *Grasshopper*.

Parâmetros do processo de projeto

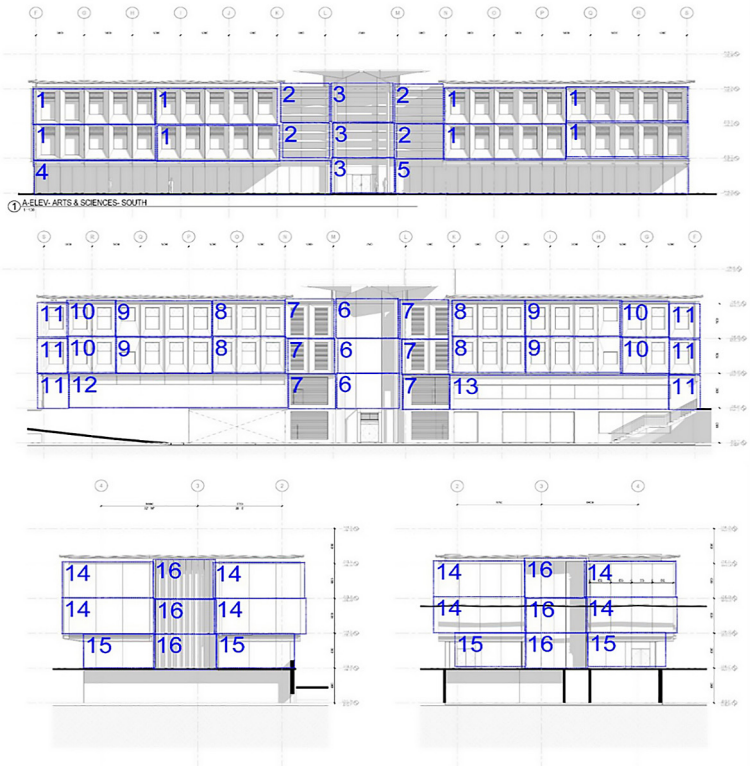
Os parâmetros definidos na modelagem paramétrica do projeto são:

Painéis paramétricos: variação de oito tipos de painéis, que deverão ser distribuídos em 15 grupos de áreas da fachada do edifício. A equipe de projeto definiu qual área de fachada poderia receber quais tipos de painéis. Por exemplo, uma área de fachada pode receber apenas dois tipos de painéis disponíveis, e há partes da fachada que não recebem nenhum tipo de painel paramétrico, como áreas de circulação vertical (Figura 6).
Além dos tipos de painéis nos grupos das fachadas, outros parâmetros aplicados no processo de projeto DSC, e diretamente relacionados à eficiência energética são:

R-Value, representa a resistência térmica de um material isolante, ou seja, sua habilidade de condução da energia térmica. Assim, quanto maior o valor *R*, maior a eficácia do material isolante, que varia de acordo com a sua espessura e densidade do material, e em alguns casos depende também da sua temperatura, envelhecimento e acúmulo de umidade.
A equipe de projeto definiu a representação dos parâmetros de *R-Value* permitidos para o projeto com os números os valores 5, 6, e 7, que correspondem a uma representação simplificada da tolerância da resistência térmica do material medida em watts por metro quadrado kelvin.

U-Value, representa a transferência térmica de um material isolante, ou seja, sua habilidade de transferência de calor. Assim,

Figura 6
Grupos da fachada do edifício *Vancouver Academic Building*, os números são um exemplo da alocação dos diferentes tipos de painéis.
Fonte: *Perkins+Will* (2017).



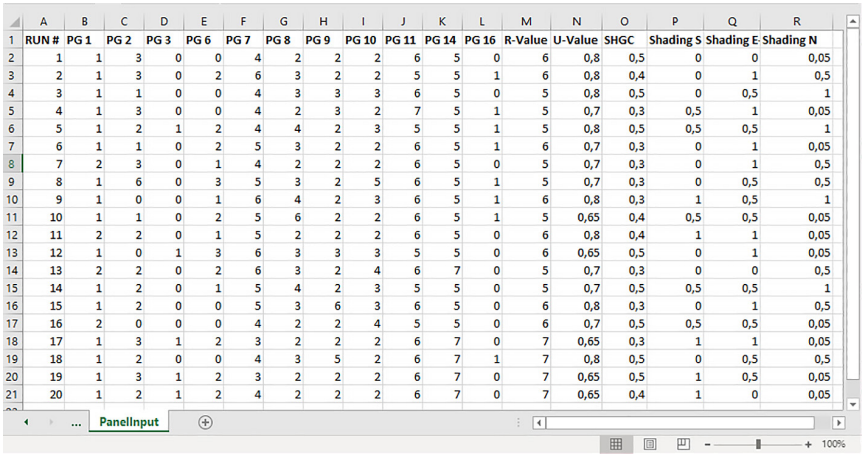
quanto menor o valor de *U*, maior a eficácia do material isolante, que varia de acordo com a sua capacidade de transferir energia térmica da massa quente para uma massa mais fria, configurando a troca de energia calórica entre dois sistemas de temperatura. Os parâmetros *U-Value* interessantes ao projeto podem variar entre 0,65; 0,7; 0,8. Correspondem a uma representação simplificada da tolerância da resistência térmica do material medida em watts por metro quadrado por kelvin. SHGC, representa a fração de radiação solar incidente transmitida, absorvida e liberada para dentro do ambiente através de uma janela. Assim, quanto menor for o coeficiente SHGC menor será o calor solar transmitido. Os valores podem variar entre 0,3; 0,4; 0,5 que correspondem à tolerância de SHGC expressa entre o domínio de 0 e 1. Também são posicionados em cada painel de brises que servem como anteparo para a incidência da luz solar, posicionados na verga superior dos painéis das fachadas norte, leste e oeste da edificação. Os valores são a largura em metros: 0; 0,5; 1 para fachada sul, 0,05; 0,5 para fachada norte e 0; 0,5; 1 para as fachadas leste e oeste. Abaixo apresentamos uma tabela com a sistematização destas variáveis, seus tipos de dados e amplitudes:

Tabela 2. Parâmetros do projeto sistematizados em variáveis

VARIÁVEIS	TIPO DE DADO	AMPLITUDE	OBSERVAÇÕES
Correspondem ao conjunto de entradas de dados que combinados, formam as alternativas de projeto. As variáveis vão desde aquelas que desencadeiam a variação geométrica do modelo paramétrico até aquelas que especificam atributos requeridos pelas análises como propriedades do material.		Intervalos e amplitudes das variáveis. Definição dos limites inferiores e superiores, intervalos.	
Painel Groups	Inteiro (número do painel).	16 grupos com combinações diferentes.	
R-Value	Decimal (valor).	5,0; 6,0; 7,0	*R-Value* (ou valor *R*) é uma medida de resistência térmica usada para análises térmicas em edificações. O valor é a razão da diferença de temperaturas entre um isolante térmico e o fluxo de calor. Quanto maior o número, melhor a eficiência do isolamento da edificação. Valor *R* é o inverso do Valor *U*.
U-Value	Decimal (valor).	0,65; 0,7; 0,8	*U-Value* (valor *U* ou transmitância térmica) é a taxa de transferência de calor (em W/m^2) dividida pela diferença de temperatura na estrutura.
SHGC	Decimal (valor).	0,3; 0,4; 0,5	O ganho solar é o aumento da energia térmica de um espaço que absorve a radiação solar incidente. A quantidade de ganho solar que um espaço experimenta é uma função da irradiância solar incidente total e da capacidade de qualquer material interveniente para transmitir ou resistir à radiação.
Brise Sul	Decimal (metros).	0,0; 0,5; 1,0	
Brise Leste Oeste	Decimal (metros).	0,0; 0,5; 1,0	
Brise Norte	Decimal (metros).	0,05; 0,5; 1,0	

Fonte: elaboração própria.

Os parâmetros são definidos dentro da modelagem paramétrica e atribuídos em uma combinação aleatória que gera cada indivíduo (ou alternativa de projeto). Os parâmetros atribuídos são gravados em uma planilha com o *plugin TTToolbox* e a estruturação dos dados gravados são organizadas através do *plugin Colibri* (Figura 7). A partir dos parâmetros estabelecidos são geradas as alternativas de projeto, que irão compor a tabela de *input*. Para compô-la, são aplicados dois métodos, o primeiro, um método randômico, definido no código do *Grasshopper,* que adota algum dos valores dentro dos parâmetros estabelecidos. O segundo, realizado através de um *Design of Experiments* (ou Projeto de Experimentos), no qual se testa um espaço de variáveis de projeto e suas métricas de desempenho a fim de perceber se há significância estatística em manter o domínio de parâmetros, ou seja, é um recurso para determinar a importância de uma variável sobre o resultado final de uma avaliação. O objetivo aqui é reduzir a quantidade de combinações possíveis de variáveis, e por sua vez, reduzir a quantidade de alternativas possíveis. No projeto original, esta fase é executada com o software proprietário JMP, e é parte importante na etapa de otimização.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	RUN #	PG 1	PG 2	PG 3	PG 6	PG 7	PG 8	PG 9	PG 10	PG 11	PG 14	PG 16	R-Value	U-Value	SHGC	Shading S	Shading E	Shading N
2	1	1	3	0	0	4	2	2	2	6	5	0	6	0,8	0,5	0	0	0,05
3	2	1	3	0	2	6	3	2	2	5	5	1	6	0,8	0,4	0	1	0,5
4	3	1	1	0	0	4	3	3	3	6	5	0	5	0,8	0,5	0	0,5	1
5	4	1	3	0	0	4	2	3	2	7	5	1	5	0,7	0,3	0,5	1	0,05
6	5	1	2	1	2	4	4	2	3	5	5	1	5	0,8	0,5	0,5	0,5	1
7	6	1	1	0	2	5	3	2	2	6	5	1	6	0,7	0,3	0	1	0,05
8	7	2	3	0	1	4	2	2	2	6	5	0	5	0,7	0,3	0	1	0,5
9	8	1	6	0	3	5	3	2	5	6	5	1	5	0,7	0,3	0	0,5	0,5
10	9	1	0	0	1	6	4	2	3	6	5	1	6	0,8	0,3	1	0,5	1
11	10	1	1	0	2	5	6	2	2	6	5	1	5	0,65	0,4	0,5	0,5	0,05
12	11	2	2	0	1	5	2	2	2	6	5	0	6	0,8	0,4	1	1	0,05
13	12	1	0	1	3	6	3	3	3	5	5	0	6	0,65	0,5	0	1	0,05
14	13	2	2	0	2	6	3	2	4	6	7	0	5	0,7	0,3	0	0	0,5
15	14	1	2	0	1	5	4	2	3	5	5	0	5	0,7	0,5	0,5	0,5	1
16	15	1	2	0	0	5	3	6	3	6	5	0	6	0,8	0,3	0	1	0,5
17	16	2	0	0	0	4	2	2	4	5	5	0	6	0,7	0,5	0,5	0,5	0,05
18	17	1	3	1	2	3	2	2	2	6	7	0	7	0,65	0,3	1	1	0,05
19	18	1	2	0	0	4	3	5	2	6	7	1	7	0,8	0,5	0	0,5	0,5
20	19	1	3	1	2	3	2	2	2	6	7	0	7	0,65	0,5	1	0,5	0,05
21	20	1	2	1	2	4	2	2	2	6	7	0	7	0,65	0,4	1	0	0,05

Figura 7
Imagem da tabela de *Input* dos parâmetros de projeto. Cada linha representa os parâmetros de uma alternativa de projeto, ou seja, de uma solução no espaço de busca no processo de projeto.
Fonte: Digiandomenico (2019).

Otimização proposta - caixa aberta

O método de otimização operado neste experimento é calculado pelo *plugin ArchOptimum*, desenvolvido inicialmente por Gabriele Landim, Dyego Digiandomenico, Jean Amaro, Anja Pratschke, Marcelo Tramontano e Claudio Toledo (2017). O *plugin ArchOptimum* foi desenvolvido em C# para *Grasshopper*, e comunica com uma aplicação externa desenvolvida em C++ através da troca de informações via arquivo. A aplicação avalia os valores obtidos nas simulações de eficiência energética e atribui um valor de desempenho para cada solução de projeto analisada.

Algoritmo genético implementado

O *ArchOptimum* emprega um algoritmo genético (AG), no processo de otimização das alternativas de projeto. O AG gera uma população de indivíduos a partir dos parâmetros de entrada fornecidos, os mesmos estabelecidos pelo escritório *Perkins+Will*. As soluções são codificadas como vetores *n* dimensionais $v = (v1, v2, \ldots, vi, \ldots, vn)$, onde cada entrada corresponde a $vi \in [min, max]$. Os parâmetros *min* e *max* representam o intervalo de variação ponderada do resultado obtido nas simulações. Assim, cada vetor *v* representa um indivíduo no AG e a população inicial de indivíduos é gerada assumindo uma distribuição uniforme na determinação das entradas v_i. A avaliação dos indivíduos utiliza a mesma função *fitness* aplicada pelo escritório, para efeito de comparação do desempenho dos dois métodos. Assim, a função *fitness* mensura três resultados de desempenho, são eles:

Lighting, resultado das simulações das distribuições anuais de iluminação diurna dentro dos ambientes do projeto.
Cooling, resultado das simulações de resfriamento dos ambientes do projeto.
Heating, resultado das simulações de aquecimento dos ambientes do projeto.

Apesar das simulações gerarem muitos outros resultados de eficiência energética, esses são os três valores que compõem a função objetivo definida pelo escritório *Perkins+Will* no seu processo de otimização. Contudo, os três resultados de desempenho dos indivíduos são normalizados, pois são valores de natureza e grandeza distintas, assim, a normalização e função *fitness* aplicada foi:

x = resultado atual.
n = resultado normalizado de 0 a 1, em que 0 é o menor e 1 o mais alto.
i = invertido (o baixo consumo de energia é o alvo, buscando evitar números altos).
min = resultado mínimo dentro da amostra.
max = resultado máximo dentro da amostra.
w = fator de ponderação de acordo com as preferências da equipe de projeto (onde $w1 = 30$, $w2 = 50$, $w3 = 20$).
v = valor.
$n = x - / (max - min)$

$$i = 1 - n$$

$$v = w1 * ni\ Cooling + w2 * ni\ Heating + w3 * n\ Ill\ Witnin$$

Durante a etapa de reprodução, uma recombinação das informações codificadas em dois indivíduos $v1$ e $v2$ ($parent1$ e $parent2$, que equivalem aos pais da na abordagem evolutiva do AG) gera uma nova solução v ($child$). O operador *blend alpha crossover* (BLX – α) foi aplicado, onde duas versões foram implementadas como se segue:

$$BLX - \alpha 1: v_i = \alpha v_i^1 + (1 - \alpha) v_i^2 \text{ onde } \alpha \in [0,1]$$

$$BLX - \alpha 2: v_i = \alpha_1 v_i^1 + (1 - \alpha) v_i^2 \text{ onde } \alpha_i \in [0,1]$$

No *crossover* $BLX - \alpha 1$, o parâmetro α, aleatoriamente selecionado no intervalo [0,1], é o mesmo aplicado a todas as entradas das soluções v^1 e v^2 durante o processo de recombinação. No $BLX - \alpha 2$, um valor αi diferente pode ser aplicado na recombinação que gera cada entrada v_i.
Durante a fase de mutação do AG, alterações em cada entrada v_i podem ocorrer, caso a taxa de mutação λ seja satisfeita. Isso significa que, para cada entrada v_i da nova solução, um número aleatório $\beta \in [0,1]$ é gerado e, se $\beta < \lambda$, um novo valor $v_i \in [min, max]$ é definido.
Uma vez criada a nova solução a partir do *crossover* e mutação, um valor de *fitness* é atribuído, esse valor é atualizado a cada verificação de $v_i \in [min, max]$ nos valores de *Lighting, Cooling e Heating*.
Se a nova solução v apresentar valor de *fitness* melhor que v^1 ou v^2, ela assume o lugar daquele com pior valor de *fitness*.
Foi implementado um gatilho de genocídio, pelo qual um número é determinado pelo usuário e representa o limite da quantidade de gerações sem melhora no valor do *fitness*. Caso esse limite seja atingido, o melhor indivíduo é preservado –que representa a melhor solução encontrada dentro do espaço de busca– todo o restante da população é eliminado e uma nova população é iniciada. Isso evita que a população fique estagnada em um mínimo local por várias gerações. Nos parâmetros do *ArchOptimum* também foi implementado um contador para o limite de tempo da execução dado em segundos e um indicador do tamanho da população inicial.

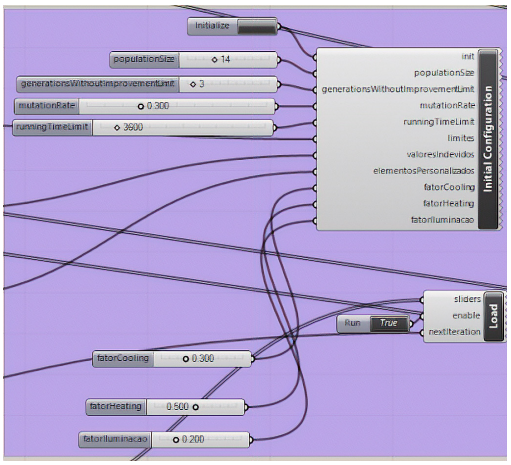


Figura 8
Imagem do ArchOptimum com seus parâmetros de funcionamento.
Fonte: Digiandomenico (2019).

Também foram implementados os controladores para taxa de mutação, tamanho da população inicial e a ponderação de *Lighting, Cooling e Heating*, assim os parâmetros podem ser ajustados pelo usuário na interface do *plugin*.

Comparação entre os métodos

Os dois métodos descritos compuseram o experimento comparativo entre o método de otimização originalmente implementado pelo escritório de arquitetura *Perkins+Will* e o do *plugin ArchOptimum*. Foi verificado qual dos dois métodos apresentou o melhor resultado de função *fitness*. Não foram considerados índices qualitativos ou quantitativos de certificações de eficiência energética na comparação dos resultados, apenas a comparação direta do desempenho entre os métodos experimentados.
O AG proposto foi executado oito vezes em um computador com processador i7-7700HQ, 2.8GHz, quatro núcleos físicos e oito núcleos lógicos, 16GB de RAM, com sistema operacional Windows 10 de 64 bits, versão do kernel 10.0.14393. Os programas

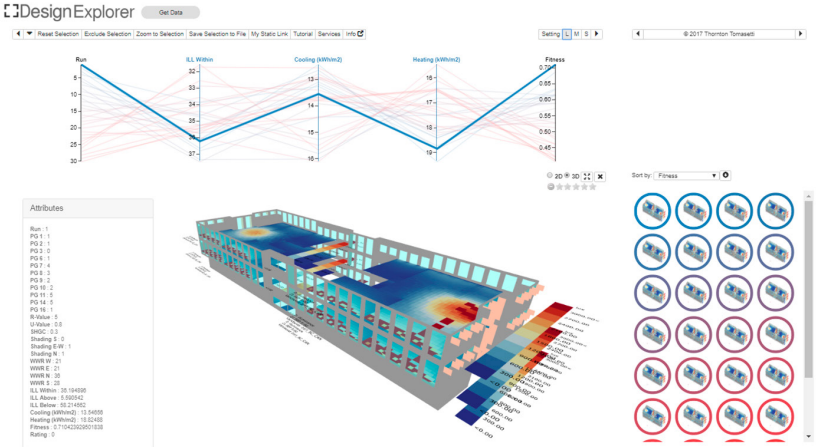
Rhinoceros 6 *Version* 6 (6.1.18023.13161, 01/23/2018) e *Grasshopper* versão 1.0 também foram utilizados. O código C++ foi desenvolvido no *Visual Studio* 2015 Enterprise, atualização 3, compilador Visual C++ 14.0, 64 bits.
Os parâmetros definidos para as execuções do *ArchOptimum*, nos respectivos campos, foram:

> *populationSize*, tamanho da população inicial, definida em seis indivíduos.
> *generationsWithoutImprovementLimit*, gatilho do genocídio da população em caso de estagnação, definido em três gerações.
> *mutationRate*, taxa de mutação empregada, definida em 0,3.
> *runningTimeLimit*, limite do tempo máximo de cada execução, definida em 3 horas de execução.
> O espaço de design explorado pelo método é composto por todos os parâmetros estabelecidos nos painéis paramétricos, *R-Value, U-Value,* SHGC, *shading*, sem a redução dos parâmetros executada pelo escritório com o programa JMP. Os valores ponderados dos componentes do cálculo da *função de fitness* preservaram a especificação do estudo de caso:
> *fatorCooling*, definido em 0,3.
> *fatorHeating*, definido em 0,5.
> *fatorLighting*, definido em 0,2.

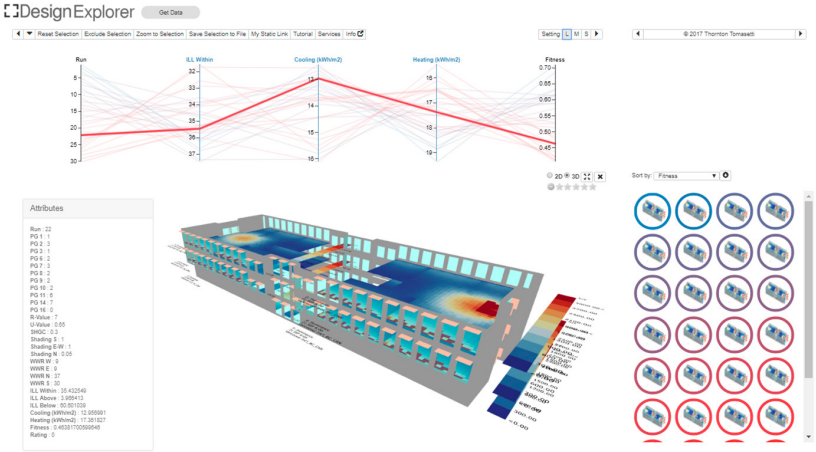
O método do *Perkins+Will* foi executado duas vezes, com as mesmas 64 alternativas já determinadas após a redução do espaço de *design*. Foi utilizada a mesma configuração de *software* e *hardware*.
O tempo médio para cada simulação analisada foi de 7 minutos, e o tempo médio de execução de todas 64 avaliações totalizou 7 horas 50 minutos.
Assim, o método do escritório *Perkins+Will* teve duas vantagens a seu favor, o espaço de busca já reduzido por sua estratégia de otimização e mais que o dobro de tempo para sua execução.
Durante a execução do experimento foi identificado um ruído nos resultados das análises produzidas pelos simuladores que poderia interferir nos resultados. Foram realizados diversos testes e pesquisas em fóruns e manuais dos desenvolvedores dos simuladores utilizados, todos os programas envolvidos no processo foram reinstalados e executados em diferentes computadores.
O erro se tratava de uma intermitência nos simuladores que não poderia ser facilmente identificado. Em algumas simulações, o mesmo modelo apresentou valores diferentes nos resultados, alterando o valor de *fitness* do indivíduo avaliado.
Contudo, a alteração só acontecia quando o programa de simulação era acionado novamente no terminal do *Grasshopper*, após todas as execuções predefinidas já terem sido executadas, ou seja, os simuladores apresentavam instabilidade de funcionamento a cada execução.
Além disso, os valores dos *fitness* não poderiam ser comparados entre execuções diferentes uma vez que na função de *fitness* estabelecida pelo escritório os valores máximos e mínimos aplicados na normalização dos resultados eram atualizados pelos valores obtidos em cada uma das execuções.
Para contornar tal situação e validar os valores de desempenho dos métodos, foi modelada a estratégia na qual o *ArchOptimum* foi rodado inicialmente oito vezes, e em cada execução foi gerada e avaliada uma média de 25 indivíduos. Os indivíduos foram salvos, e os três melhores indivíduos de cada uma das sete primeiras execuções e quatro indivíduos da última execução foram agrupados, totalizando um grupo com os 25 melhores indivíduos de todas as execuções do *ArchOptimum*. Esse grupo foi inserido na tabela que já continha todos os indivíduos gerados pelo método da *Perkins+Will*. Em sequência, todos os indivíduos foram avaliados numa mesma execução dos simuladores, contornando a variação de valores do ruído ao reiniciar a execução da simulação. Os valores máximos e mínimos utilizados no cálculo de *fitness* considerou os valores de todos os indivíduos da tabela.
O resultado obtido no experimento de comparação entre os métodos demonstrou que entre as 30 melhores soluções de projeto, as 21 melhores foram produzidas através do método do *ArchOptimum*. O resultado confirmou o desempenho favorável do *plugin ArchOptimum* em comparação ao método aplicado pelo escritório (Figura 9).
Após obtermos do otimizador os valores de resultado da função objetivo (*fitness*), o conjunto completo de dados foi carregado na aplicação web Designer Explorer.
O gráfico representa a correlação entre as colunas de alternativas de projeto, os objetivos propostos,os índices de desempenho de cada alternativa e suas respectivas funções *fitness.*

Indiv	Fitness
1	0,71042393
2	0,687830655
3	0,635317417
4	0,63194942
5	0,617228123
6	0,616548542
7	0,614352944
8	0,613735921
9	0,603772484
10	0,602215447
11	0,590470948
12	0,588957856
13	0,573913386
14	0,551725748
15	0,542376698
16	0,52897754
17	0,528325989
18	0,513514592
19	0,492946232
20	0,48449132
21	0,477783033
22	0,463817006
23	0,460735865
24	0,451107423
25	0,438774803
26	0,423307369
27	0,422712283
28	0,413764587
29	0,412177513
30	0,407466289

Figura 9
Ranque dos 30 melhores indivíduos, com grifo em azul os indivíduos gerados com o *ArchOptimum*.
Fonte: Digiandomenico (2019).

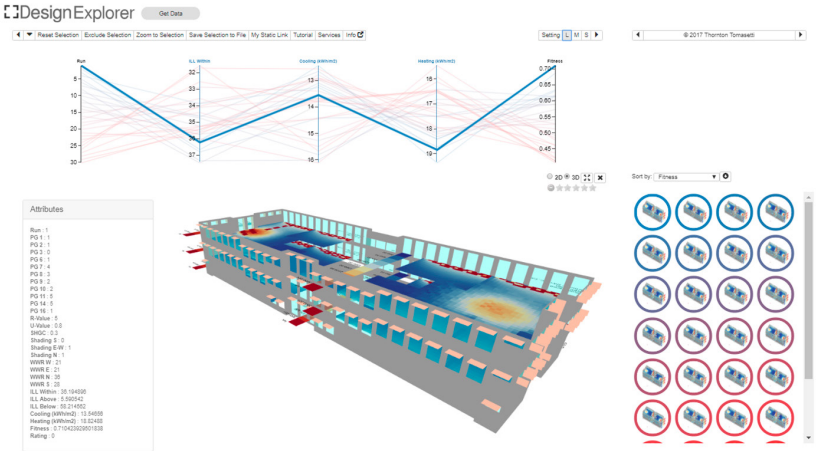


10

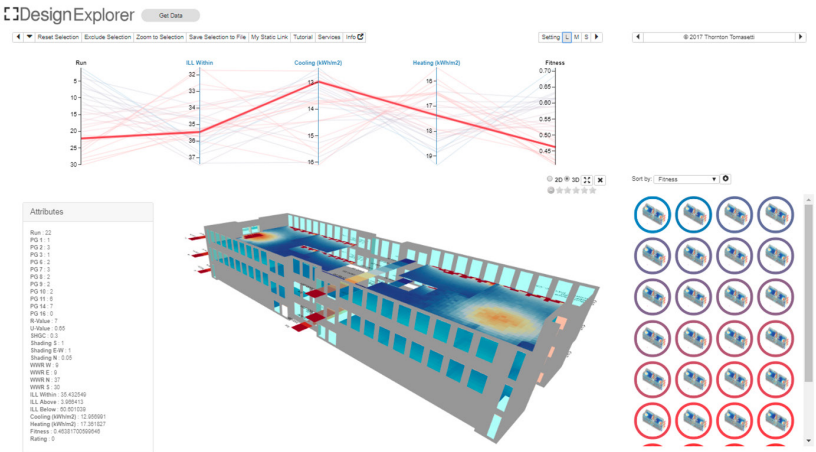


11

Figuras 10 e 11
Respectivamente a fachada Sul dos melhores indivíduos (alternativas de projeto) obtidos através do ArchOptimum e do método do escritório *Perkins+Will*.
Fonte: Digiandomenico (2019).



12



13

Figuras 12 e 13
Respectivamente a fachada Norte dos melhores indivíduos (alternativas de projeto) obtidos através do ArchOptimum e do método do escritório *Perkins+Will*.
Fonte: Digiandomenico (2019).

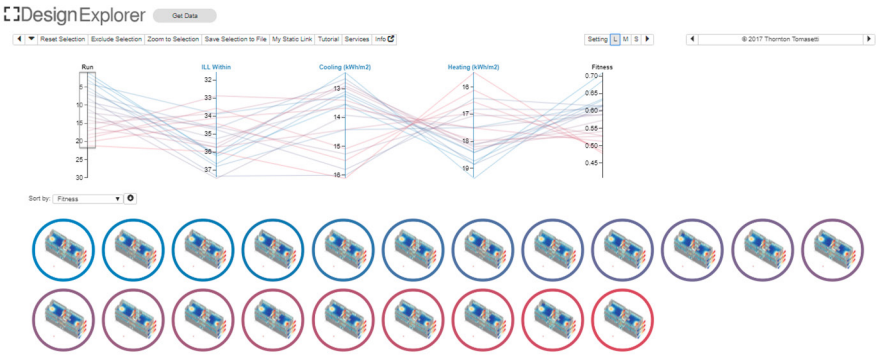


Figura 14
Gráfico de Coordenadas Paralelas dos parâmetros utilizados no cálculo do *fitness* e o respectivo resultado dos indivíduos gerados através do *ArchOptimum*. A cor azul representa os melhores resultados, a vermelha os piores resultados.
Fonte: Digiandomenico (2019).

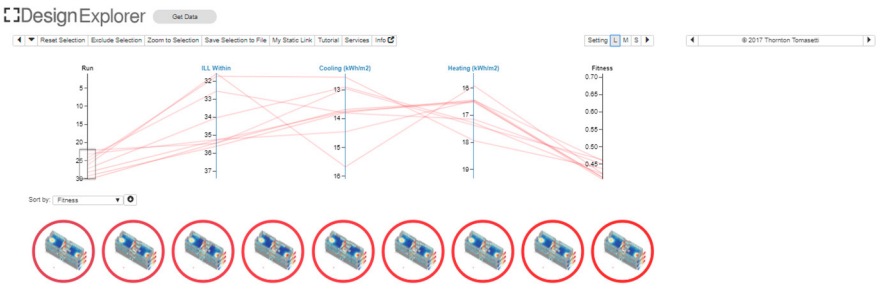


Figura 15
Gráfico de Coordenadas Paralelas dos parâmetros utilizados no cálculo do *fitness* e o respectivo resultado dos indivíduos gerados através do método do escritório *Perkins+Will*. A cor azul representa os melhores resultados, a vermelha os piores resultados.
Fonte: Digiandomenico (2019).

Discussão e conclusões do experimento

A abertura e comparação dos dois métodos explorados durante o experimento demonstraram que o resultado obtido pelo *ArchOptimum* superou os do projeto original. Todas as operações implementadas no *plugin* permaneceram abertas para intervenções ao longo do experimento, possibilitando adaptações e aperfeiçoamentos. Também foi demonstrada a capacidade de adaptabilidade do *plugin* ao se encaixar no processo de projeto DSC desenvolvido pelo escritório. O experimento evidenciou a complexidade técnica de se trabalhar com simuladores e otimização de maneira eficiente. O tempo necessário para executar as simulações das alternativas de projeto podem ser maiores caso o algoritmo não esteja definido com estruturas de dados eficazes, por exemplo. O experimento demonstrou a capacidade do *ArchOptimum* de atingir um resultado satisfatório de otimização mesmo com uma população inicial pequena, ou seja, com pouca variabilidade de indivíduos iniciais. Além disso, em um algoritmo de código fonte aberto, é possível aferir melhor o impacto das taxas de mutação, elitismo e estratégia de cruzamento de alternativas, no caso do AG, o que deixa o desempenho melhor adaptado à função objetivo em teste. Outros detalhes de funcionamento do AG que estão fora do escopo deste artigo, estão descritos em Landim, Digiandomenico, Amaro, Pratschke, Tramontano e Toledo (2017). O método obteve melhor desempenho em relação ao resultado da função implementada pelo escritório, pois metaheurísticas são conhecidas por aprenderem com os resultados históricos de buscas anteriores para se guiar pelo espaço de pesquisa, obtendo mais probabilidade de escapar de paradas em ótimos locais. O processo de aprender o funcionamento pormenorizado de implementar um AG do zero, permite entender e conhecer o espaço de busca por soluções de uma maneira que pode ser difícil de visualizar na fase inicial do projeto. Gerar soluções otimizadas em ferramentas de código fonte fechado, quando o programa informa ao projetista qual as melhores alternativas de projeto sem que se entenda o que de fato está sendo computado, pode criar confiança cega em um método de otimização pouco adequado para o problema que se tenta resolver. O projetista também fica suscetível à falsa percepção de que métodos computacionais

e de otimização são adequados para qualquer tipo de problema arquitetônico. Conduzimos a investigação a partir de um processo de projeto pronto, previamente definido e pensado por uma equipe de projetistas, pesquisadores e demais interessados do escritório que concedeu os dados do experimento. Para além do objetivo de testar a adaptabilidade e ganhos de performance do algoritmo de otimização de código aberto desenvolvido pelos autores, um dos maiores questionamentos resultantes no final do processo foi: quais teriam sido nossas métricas e propostas frente ao objetivo inicial do projeto, caso fossemos a equipe de projetistas? Abre-se aqui algumas ordens de discussão. No nível da técnica, os arquitetos mais interessados nos problemas de otimização podem se perguntar: qual o melhor método de otimização podemos usar para este problema? Devo usar um método estocástico, um modelo matemático ou um método exato? O que é possível tangenciar dos conhecimentos de otimização quando se transita entre as áreas de arquitetura, engenharia e computação? Para além de métodos mais orientados à problemas da natureza das engenharias, como métodos baseado em modelos (*model-based*) ou algoritmos de busca direta (*direct search*) (Wortmann e Schroepfer, 2019), é difícil estabelecer que algum método estocástico possui melhor desempenho que outros. Métodos estocásticos geram distribuição de probabilidade e oferecem recursos para criar previsões da melhor solução. No nível da natureza do projeto de arquitetura e dos problemas iniciais do projeto emergem questões como: e se as decisões e métricas iniciais foram pouco adequadas para o problema? Ficaram de fora do escopo inicial materiais ou soluções outras que possam ser mais adequadas ao projeto do que as encontradas via método de otimização? Existem uma série de fatores objetivos e quantificáveis que podem ser bem resolvidos com a ajuda de métodos de otimização, mas precisamos questionar as crenças de que um método quantitativo pode oferecer a melhor solução em qualquer caso ou tipo de problema. É preciso antes de tudo, estabelecer uma relação crítica com o processo. Após focar em fluxos de trabalho baseados em dados e na exatidão computacional, atesta-se que podemos obter o melhor resultado possível dentro do desenho do experimento. Epistemologias, valores e visões de mundo em como abordar um determinado problema estão naturalmente implícitas no projeto deste fluxo de trabalho. Quando vale a pena usar um método computacional e em quais situações? Há pouca discussão sobre como os contextos tecnocientíficos locais, de países que estão fora dos centros de desenvolvimento de tecnologias hegemônicas, podem dialogar em simetria com os avanços tecnológicos dos países do centro do capitalismo. Como saberes, ambientes, visões de mundo e tecnociências produzidas com outros recursos podem dialogar com o que tem sido considerado soluções apropriadas e tecnológicas?

Considerações Finais

A aplicação de métodos de otimização na busca do balanceamento ideal de parâmetros presentes desde a concepção até a fabricação do projeto requer conhecimentos específicos, que muitas vezes, não são comuns à formação do arquiteto. Contudo, recursos do design computacional estão apoiando os profissionais a inserir a otimização em seus processos através de modelos paramétricos produzidos com linguagens de programação visual e textual. Os modelos paramétricos são analisados por simuladores, que são também fundamentais ao processo de projeto orientado ao desempenho, pois os dados que produzem informam as decisões de otimização dentro do espaço de busca por soluções. Porém, a construção do próprio espaço de busca é uma competência da modelagem do processo de projeto. Ou seja, o arquiteto é responsável pela construção do espaço de busca que será analisado e para isso precisa entender com nitidez não só as exigências do programa arquitetônico, mas também os potenciais e limites das ferramentas que irão apoiar sua busca. Nesse contexto, é crescente na arquitetura o emprego de programas computacionais, tanto de simulação quanto de otimização, que operam métodos através de caixas-pretas, geralmente em forma de algoritmos de código fonte fechado. Essa condição, por um lado, pode facilitar e popularizar o processo, mas por outro, enviesa questões, como a validação dos resultados obtidos ou compreensões sobre os limites da eficiência dos métodos empregados no processo de projeto.

O conteúdo da investigação de métodos de otimização também é interdisciplinar e relaciona conhecimentos de campos da computação, matemática e algoritmos inspirados em processos biológicos, por exemplo. Para estimular a discussão e difusão dos métodos é necessário que as pesquisas reportem não apenas os resultados obtidos, mas também o conjunto de procedimentos aplicados e os códigos desenvolvidos, relatando suas limitações e potenciais.
O aumento na transparência dos métodos tecnocientíficos tem potencial de contribuição que vai além do próprio processo científico de verificação pela comunidade acadêmica, ao desmistificar um processo de mercantilização de soluções destas tecnologias. Parte da divulgação destes métodos pode recair em estratégias mais voltadas ao *marketing* que escritórios e empresas de *ponta* atribuem ao processo, do que por suas discussões abertas sobre a adequação entre natureza do problema e método. Além disso, replica-se a busca por soluções que respondem às realidades de países e centros de pesquisa localizados no norte global, sem uma reflexão e aplicação sobre como se discute a natureza dos problemas (como o de sustentabilidade) em lugares cujo contexto tecnocientífico, econômico e geográfico permanece sob lógicas diferentes de funcionamento.
Uma vez que o projeto orientado ao desempenho oferece um método que elege a melhor opção de projeto possível dentre centenas ou milhares de alternativas testadas, apoia-se nos dados, cálculos e gráficos como justificativa incontestável da adequabilidade da solução de projeto.

Outros espaços de solução, e possibilidades variadas que não foram declaradas, ficam, obviamente, fora deste conjunto, e poderiam adequar-se à soluções que interpretam os problemas a partir de outras epistemologias ou visões de mundo.
Um dos principais desdobramentos futuros possíveis da presente pesquisa é compreender melhor a relação entre a abertura das caixas-pretas e a produção de autonomia local em arquitetura e tecnologias, compreendendo, inclusive, observar quais outras possibilidades de solução de problemas estão disponíveis e que não fazem parte dos tipos de simulação e otimização que os *plugins*, softwares e sistemas computacionais oferecem atualmente.
Todo o conteúdo produzido na presente pesquisa, incluindo o código fonte do *plugin ArchOptimum*, está disponibilizado no endereço eletrônico <https://github.com/landimg/ArchOptimum>, objetivando servir de apoio ao ensino e pesquisas correlatas ao tema. Por fim, com o apoio da metodologia empregada na presente pesquisa, desde a revisão bibliográfica aos experimentos realizados, obteve-se resultados que colaboram com a abertura de métodos mais eficientes, verificáveis e reproduzíveis, tratando-se de otimização em arquitetura ■

> REFERÊNCIAS

Attia, S., Hamdy, M., O'Brien, W. e Carlucci, S. (2013). Assessing gaps and needs for integrating building performance optimization tools in net zero energy buildings design. *Energy and Buildings*, *60*, pp. 110-124.

Costa, A., Nannicini, G., Schroepfer, T. e Wortmann, T. (2015). Black-box optimization of lighting simulation in architectural design [pp. 27-39]. Em M. A. Cardin, D. Hastings, P. Jackson, D. Krob, P. Ch. Lui e G. Schmitt (Eds.), *Complex systems design & management Asia*. Berlim: Springer.

Digiandomenico, D. S. (2019). *Otimização de projeto orientado ao desempenho em arquitetura*. [Arquivo PDF- Dissertação de Mestrado]. São Carlos: Instituto de Arquitetura e Urbanismo/Universidade de São Paulo. DOI: 10.11606/D.102.2019.tde-09092019-095255

Haymaker, J., Bernal, M., Marshall, M. T., Okhoya, V., Szilasi, A., Rezaee, R., ... e Welle, B. (2018). Design space construction: a framework to support collaborative, parametric decision making. *Journal of Information Technology in Construction (ITcon)*, *23*(8), pp. 157-178.

Holmström, K. (2008). An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization. *Journal of Global Optimization*, *41*(3), pp. 447-464.

Landim, G. (2019). *Programação para Arquitetura: linguagens visuais e textuais em Projeto Orientado ao Desempenho*. [Arquivo PDF - Dissertação de Mestrado]. São Carlos: Instituto de Arquitetura e Urbanismo/ Universidade de São Paulo. DOI: 10.11606/D.102.2019.tde-09092019-100632

Landim, G., Digiandomenico, D., Amaro, J., Pratschke, A., Tramontano, M. e Toledo, C. (2017). Architectural Optimization and Open Source Development: Nesting and Genetic Algorithms [pp. 340-349]. [Arquivo PDF]. Em *Proceedings of the 39th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*. Cambridge: MIT. DOI: 10.52842/conf.acadia.2017.340

Li, S., Liu, L., e Peng, C. (2020). A Review of Performance-Oriented Architectural Design and Optimization in the Context of Sustainability: Dividends and Challenges. [Arquivo PDF]. *Sustainability*, *12*(4), 1427, pp. 1-36. MDPI AG. DOI: 10.3390/su12041427

Machairas, V., Tsangrassoulis, A. e Axarli, K. (2014). Algorithms for optimization of building design: A review. *Renewable and sustainable energy reviews*, *31*, pp. 101-112.

Perkins+Will. (julho de 2017). Design Space Construction Tutoria. [Em linha]. *Gitbook*. Disponível em https://www.gitbook.com/book/bernalm/design-space-construction/details

Rios, L. M. (2009). *Algorithms for derivative-free optimization*. Urbana-Champaign: University of Illinois.

Shi, X. (2010). Performance-based and performance-driven architectural design and optimization. *Frontiers of Architecture and Civil Engineering in China*, *4*(4), pp. 512-518.

Shi, X. e Yang, W. (2013). Performance-driven architectural design and optimization technique from a perspective of architects. *Automation in Construction*, *32*, pp. 125-135.

Touloupaki, E. e Theodosiou, T. (2017). Performance simulation integrated in parametric 3D modeling as a method for early stage design optimization—A review. *Energies*, *10*(5), p. 637.

Wortmann, T. e Nannicini, G. (2017). Introduction to architectural design optimization [pp. 279-278]. Em A. Karakitsiou, A. Migdalas, S. Th. Rassia e P. M. Pardalos (Eds.), *City Networks*. Berlim: Springer.

Wortmann, T. e Schroepfer, T. (2019, April). From optimization to performance-informed design [pp. 261-268]. Em S. Rockcastle, T. Rakha, C. Cerezo Dávila, D. Papanikolaou e T. Zakula (Eds.), *Proceedings of the Symposium on Simulation for Architecture and Urban Design*. Atlanta: SimAUD.